# Table of Contents

Welcome to knowledge management...............................................................1  
What is Splunk knowledge?.......................................................................1  
Why manage Splunk knowledge?...........................................................3  
Prerequisites for knowledge management..............................................4  

Get started with knowledge objects.......................................................7  
Manage knowledge objects through Settings pages..........................7  
Monitor and organize knowledge objects..........................................9  
The sequence of search-time operations...........................................10  
Give knowledge objects of the same type unique names....................22  
Develop naming conventions for knowledge objects.......................24  
Understand and use the Common Information Model Add-on............26  
Manage knowledge object permissions............................................27  
Manage orphaned knowledge objects..............................................34  
Disable or delete knowledge objects..............................................41  
About Splunk regular expressions....................................................44  

Fields and field extractions......................................................................51  
About fields............................................................................................51  
Use default fields...................................................................................58  
When Splunk software extracts fields..............................................65  
About regular expressions with field extractions...........................67  

Use the field extractor in Splunk Web....................................................69  
Build field extractions with the field extractor...............................69  
Field Extractor: Select Sample step.................................................73  
Field Extractor: Select Method step.................................................78  
Field Extractor: Select Fields step....................................................79  
Field Extractor: Rename Fields step..................................................86  
Field Extractor: Validate step............................................................89  
Field Extractor: Save step.................................................................90  

Use the settings pages for field extractions in Splunk Web.............92  
Use the Field extractions page...........................................................92  
Use the Field transformations page.................................................99  

Use the configuration files to configure field extractions..................106  
Configure custom fields at search time..........................................106  
Configure inline extractions...........................................................108
# Table of Contents

**Use the configuration files to configure field extractions**
- Configure advanced extractions with field transforms.......................... 110
- Configure automatic key-value field extraction................................... 119
- Example inline field extraction configurations.................................. 122
- Example transform field extraction configurations.......................... 124
- Configure extractions of multivalue fields with fields.conf................. 128

**Calculated fields**................................................................................. 131
- About calculated fields........................................................................ 131
- Create calculated fields with Splunk Web.......................................... 133
- Configure calculated fields with props.conf...................................... 134

**Event types**.......................................................................................... 136
- About event types................................................................................. 136
- Define event types in Splunk Web....................................................... 139
- About event type priorities.................................................................. 144
- Automatically find and build event types............................................ 145
- Configure event types in eventtypes.conf.......................................... 149
- Configure event type templates......................................................... 152

**Transactions**.......................................................................................... 153
- About transactions................................................................................ 153
- Search for transactions......................................................................... 155
- Configure transaction types................................................................. 158

**Use lookups in Splunk Web**.................................................................. 162
- About lookups....................................................................................... 162
- Define a CSV lookup in Splunk Web..................................................... 166
- Define an external lookup in Splunk Web............................................ 171
- Define a KV Store lookup in Splunk Web............................................ 176
- Define a geospatial lookup in Splunk Web........................................... 181
- Define a time-based lookup in Splunk Web.......................................... 187
- Define an automatic lookup in Splunk Web......................................... 188
- Lookup example in Splunk Web............................................................ 189

**Use the configuration files to configure lookups**................................. 193
- Introduction to lookup configuration.................................................... 193
- Configure CSV lookups........................................................................ 194
- Configure external lookups.................................................................. 201
Table of Contents

Use the configuration files to configure lookups
  ConfigureKV Store lookups .............................................................. 205
  Configure geospatial lookups ............................................................ 210
  Add field matching rules to your lookup configuration ..................... 217
  Configure a time-based lookup ....................................................... 218
  Make your lookup automatic ............................................................ 221

Workflow actions ................................................................................. 226
  About workflow actions in Splunk Web ............................................. 226
  Set up a GET workflow action ............................................................ 228
  Set up a POST workflow action ......................................................... 231
  Set up a search workflow action.......................................................... 235
  Control workflow action appearance in field and event menus ............ 237
  Use special parameters in workflow actions ....................................... 238

Tags ........................................................................................................ 240
  About tags and aliases ....................................................................... 240
  Tag field-value pairs in Search ........................................................... 242
  Define and manage tags in Search ....................................................... 245
  Tag the host field .............................................................................. 249
  Tag event types .................................................................................. 251

Field aliases ........................................................................................ 252
  Create field aliases in Splunk Web ...................................................... 252
  Configure field aliases with props.conf .............................................. 253

Search macros ...................................................................................... 255
  Use search macros in searches ........................................................... 255
  Define search macros in Settings ......................................................... 257
  Search macro examples ...................................................................... 259

Manage and explore datasets .............................................................. 263
  Dataset types and usage ..................................................................... 263
  Manage datasets .................................................................................. 264
  Explore a dataset .................................................................................. 270

Create and edit table datasets ............................................................ 276
  Table datasets and the Splunk Datasets Add-on ..................................... 276
  Manage table datasets ........................................................................ 278
# Table of Contents

**Create and edit table datasets**
- Define initial data for a new table dataset ................................................. 282
- Use the Table Editor .................................................................................. 288
- Dataset extension ....................................................................................... 296
- Accelerate tables ....................................................................................... 299

**Build a data model** .................................................................................... 304
- About data models ..................................................................................... 304
- Manage data models .................................................................................. 315
- Design data models ................................................................................... 327

**Define data model dataset fields** ............................................................... 337
- Define dataset fields ................................................................................ 337
- Add an auto-extracted field ..................................................................... 342
- Add an eval expression field .................................................................... 344
- Add a lookup field .................................................................................... 347
- Add a regular expression field ................................................................ 351
- Add a geo IP field .................................................................................... 354

**Use data summaries to accelerate searches** ............................................ 356
- Overview of summary-based search acceleration .................................... 356
- Manage report acceleration ..................................................................... 362
- Accelerate data models .......................................................................... 384
- Use summary indexing for increased reporting efficiency ....................... 402
- Manage summary index gaps ................................................................... 411
- Configure summary indexes .................................................................. 415
- Configure batch mode search ................................................................. 422
Welcome to knowledge management

What is Splunk knowledge?

Splunk software provides a powerful search and analysis engine that helps you to see both the details and the larger patterns in your IT data. When you use Splunk software you do more than look at individual entries in your log files; you leverage the information they hold collectively to find out more about your IT environment.

Splunk software extracts different kinds of knowledge from your IT data (events, fields, timestamps, and so on) to help you harness that information in a better, smarter, more focused way. Some of this information is extracted at index time, as Splunk software indexes your IT data. But the bulk of this information is created at "search time," both by Splunk software and its users. Unlike databases or schema-based analytical tools that decide what information to pull out or analyze beforehand, Splunk software enables you to dynamically extract knowledge from raw data as you need it.

As your organization uses Splunk software, additional categories of Splunk software knowledge objects are created, including event types, tags, lookups, field extractions, workflow actions, and saved searches.

You can think of Splunk software knowledge as a multitool that you use to discover and analyze various aspects of your IT data. For example, event types enable you to quickly and easily classify and group together similar events; you can then use them to perform analytical searches on precisely-defined subgroups of events.

The Knowledge Manager manual shows you how to maintain sets of knowledge objects for your organization through Splunk Web and configuration files, and it demonstrates ways that you can use Splunk knowledge to solve your organization's real-world problems.

Splunk software knowledge is grouped into five categories:

- **Data interpretation: Fields and field extractions** - Fields and field extractions make up the first order of Splunk software knowledge. The fields that Splunk software automatically extracts from your IT data help bring meaning to your raw data, clarifying what can at first glance seem incomprehensible. The fields that you extract manually expand and
improve upon this layer of meaning.

- **Data classification: Event types and transactions** - You use event types and transactions to group together interesting sets of similar events. Event types group together sets of events discovered through searches, while transactions are collections of conceptually-related events that span time.

- **Data enrichment: Lookups and workflow actions** - Lookups and workflow actions are categories of knowledge objects that extend the usefulness of your data in various ways. Field lookups enable you to add fields to your data from external data sources such as static tables (CSV files) or Python-based commands. Workflow actions enable interactions between fields in your data and other applications or web resources, such as a WHOIS lookup on a field containing an IP address.

- **Data normalization: Tags and aliases** - Tags and aliases are used to manage and normalize sets of field information. You can use tags and aliases to group sets of related field values together, and to give extracted fields tags that reflect different aspects of their identity. For example, you can group events from set of hosts in a particular location (such as a building or city) together--just give each host the same tag. Or maybe you have two different sources using different field names to refer to same data--you can normalize your data by using aliases (by aliasing `clientip` to `ipaddress`, for example).

- **Data models** - Data models are representations of one or more datasets, and they drive the Pivot tool, enabling Pivot users to quickly generate useful tables, complex visualizations, and robust reports without needing to interact with the Splunk software search language. Data models are designed by knowledge managers who fully understand the format and semantics of their indexed data. A typical data model makes use of other knowledge object types discussed in this manual, including lookups, transactions, search-time field extractions, and calculated fields.

The *Knowledge Manager* manual includes information about the following topic:

- **Summary-based report and data model acceleration** - When searches and pivots are slow to complete use Splunk software to speed things up. This chapter discusses report acceleration (for searches), data model acceleration (for pivots) and summary indexing (for special case searches).

For information on why you should manage Splunk knowledge, see Why manage Splunk knowledge?.
Knowledge managers should have a basic understanding of data input setup, event processing, and indexing concepts. For more information, see Prerequisites for knowledge management.

**Why manage Splunk knowledge?**

If you have to maintain a fairly large number of knowledge objects across your Splunk deployment, you know that management of that knowledge is important. This is especially true of organizations that have a large number of Splunk users, and even more so if you have several teams of users working with Splunk software. This is simply because a greater proliferation of users leads to a greater proliferation of additional Splunk knowledge.

When you leave a situation like this unchecked, your users may find themselves sorting through large sets of objects with misleading or conflicting names, struggling to find and use objects that have unevenly applied app assignments and permissions, and wasting precious time creating objects such as reports and field extractions that already exist elsewhere in the system.

Splunk knowledge managers provide centralized oversight of Splunk software knowledge. The benefits that knowledge managers can provide include:

- **Oversight of knowledge object creation and usage across teams, departments, and deployments.** If you have a large Splunk deployment spread across several teams of users, you'll eventually find teams "reinventing the wheel" by designing objects that were already developed by other teams. Knowledge managers can mitigate these situations by monitoring object creation and ensuring that useful "general purpose" objects are shared on a global basis across deployments.

  For more information, see Monitor and organize knowledge objects.

- **Normalization of event data.** To put it plainly: knowledge objects proliferate. Although Splunk software is based on data indexes, not databases, the basic principles of normalization still apply. It's easy for any robust, well-used Splunk implementation to end up with a dozen tags that all have been to the same field, but as these redundant knowledge objects stack up, the end result is confusion and inefficiency on the part of its users. We'll provide you with some tips about normalizing your knowledge object libraries by applying uniform naming standards and using the Splunk Common Information Model.
For more information, see Develop naming conventions for knowledge objects.

• **Management of knowledge objects through configuration files.** Some aspects of knowledge object setup are best managed through configuration files. This manual will show Splunk Enterprise knowledge managers how to work with knowledge objects in this way.

See Create and maintain search-time field extractions through index files as an example of how you can manage Splunk knowledge through configuration files.

• **Creation of data models for Pivot users.** Splunk software offers the Pivot tool for users who want to quickly create tables, charts, and dashboards without having to write search strings that can sometimes be long and complicated. The Pivot tool is driven by **data models**—without a data model Pivot has nothing to report on. Data models are designed by Splunk knowledge managers: people who understand the format and semantics of their indexed data, and who are familiar with the Splunk search language.

See About data models for a conceptual overview of data model architecture and usage.

• **Manage setup and usage of summary-based search and pivot acceleration tools.** Large volumes of data can result in slow performance for Splunk software, whether you're launching a search, running a report, or trying to use Pivot. To speed things up the knowledge manager can make use of **report acceleration**, data model acceleration, and **summary indexing** to help ensure that the teams in your deployment can get results quickly and efficiently. This manual shows you how to provide centralized oversight of these acceleration strategies so you can ensure that they are being used responsibly and effectively.

For more information, see Overview of summary-based search and pivot acceleration.

**Prerequisites for knowledge management**

Most knowledge management tasks are centered around search time event manipulation. In other words, a typical knowledge manager usually doesn't focus their attention on work that takes place before events are indexed, such as
setting up data inputs, adjusting event processing activities, correcting default field extraction issues, creating and maintaining indexes, setting up forwarding and receiving, and so on.

However, we do recommend that all knowledge managers have a good understanding of these concepts. A solid grounding in these subjects enables knowledge managers to better plan out their approach towards management of knowledge objects for their deployment...and it helps them troubleshoot issues that will inevitably come up over time.

Here are some topics that knowledge managers should be familiar with, with links to get you started:

- **Inherit a Splunk Enterprise deployment**: If you have inherited a Splunk Enterprise deployment, you can find more information on your deployment's network characteristics, data sources, user population, and knowledge objects in the Introduction in the *Inherited Deployment* manual.

- **Working with Splunk apps**: If your deployment uses more than one Splunk app, you should get some background on how they're organized and how app object management works within multi-app deployments. See What's an app?, App architecture and object ownership, and Manage app objects in the *Admin* manual.

- **Configuration file management**: Where are the configuration files? How are they organized? How do configuration files take precedence over each other? See About configuration files and Configuration file precedence in the *Admin* manual.

- **Indexing incoming data**: What is an index and how does it work? What is the difference between "index time" and "search time" and why is this distinction significant? Start with About indexes and indexers in the Managing Indexers and Clusters manual and read the rest of the chapter. Pay special attention to Index time vs search time.

- **Getting event data into your Splunk deployment**: It's important to have at least a baseline understanding of Splunk data inputs. Check out What Splunk can index and read the other topics in the *Getting Data In* manual as necessary.

- **Understand your forwarding and receiving setup**: If your Splunk deployment utilizes forwarders and receivers, it's a good idea to get a handle on how they've been implemented, as this can affect your
knowledge management strategy. Get an overview of the subject at About forwarding and receiving in the Forwarding Data manual.

• **Understand event processing:** It's a good idea to get a good grounding in the steps that Splunk software goes through to "parse" data before it indexes it. This knowledge can help you troubleshoot problems with your event data and recognize "index time" event processing issues. Start with Overview of event processing in the Getting Data In manual and read the entire chapter.

• **Default field extraction:** Most field extraction takes place at search time, with the exception of certain default fields, which get extracted at index-time. As a knowledge manager, most of the time you'll concern yourself with search-time field extraction, but it's a good idea to know how default field extraction can be managed when it's absolutely necessary to do so. This can help you troubleshoot issues with the host, source, and sourcetype fields that Splunk software applies to each event. Start with About default fields in the Getting Data In manual.

• **Managing users and roles:** Knowledge managers typically do not directly set up users and roles. However, it's a good idea to understand how they're set up within your deployment, as this directly affects your efforts to share and promote knowledge objects between groups of users. For more information, start with About users and roles in the Admin manual, and read the rest of the chapter as necessary.
Get started with knowledge objects

Manage knowledge objects through Settings pages

As your organization uses Splunk software, people add knowledge to the base set of event data indexed within it. You and your colleagues might:

- Save and schedule searches.
- Add tags to fields.
- Define event types and transactions that group together sets of events.
- Create lookups and workflow actions.

The process of creating knowledge objects starts slowly, but it can become complicated as people use Splunk software for longer periods. It is easy to reach a point where users are creating searches that already exist, adding unnecessary tags, designing redundant event types, and so on. These issues may not be significant if your user base is small. But if they accumulate over time, they can cause unnecessary confusion and repetition of effort.

This chapter discusses how knowledge managers can use the Knowledge pages in Settings to control the knowledge objects in their Splunk deployment. Settings can give an attentive knowledge manager insight into what knowledge objects people are creating, who is creating them, and (to some degree) how people are using them.

With Settings, you can easily:

- Create knowledge objects when you need to, either "from scratch" or through object cloning.
- Review knowledge objects as others create them, in order to reduce redundancy and ensure that people are following naming standards.
- Delete unwanted or poorly-defined knowledge objects before they develop downstream dependencies.
- Ensure that knowledge objects worth sharing beyond a particular working group, role, or app are made available to other groups, roles, and users of other apps.

Note: This chapter assumes that you have an admin role or a role with an equivalent permission set.

This chapter contains topics that will explain how to:
• Keep your knowledge object collections normalized and orderly.
• Develop naming conventions for your knowledge objects that will make them easier to understand and use.
• Use the Common Information Model Add-on to normalize your event data.
• Manage your knowledge object permissions. Make a knowledge object available to users of a specific app, users with a specific role, or users of all apps ("global" permissions).
• Disable or delete knowledge objects. Understand the restrictions on deleting knowledge objects, and know the risks of deleting knowledge objects that have downstream dependencies.

Managing knowledge using configuration files instead of Settings

In previous releases, Splunk Enterprise users edited configuration files directly to add, update, or delete knowledge objects. Now they can use the Knowledge pages in Settings, which provide a graphical interface for updating those configuration files.

Note: Splunk Cloud users must use the Splunk Web Knowledge pages in Settings to maintain knowledge objects.

Splunk recommends that Splunk Enterprise administrators learn how to modify configuration files. Understanding configuration files is beneficial for the following reasons:

• Some Splunk Web features make more sense if you understand how things work at the configuration file level. This is especially true for the Field extractions and Field transformations pages in Splunk Web.
• Managing certain knowledge object types requires changes to configuration files.
• Bulk deletion of obsolete, redundant, or improperly-defined knowledge objects is only possible with configuration files.
• You might find that you prefer to work directly with configuration files. For example, if you are a long-time Splunk Enterprise administrator who is already familiar with the configuration file system, you might already be familiar with managing Splunk knowledge using configuration files. Other users rely on the level of granularity and control that configuration files can provide.

The Knowledge Manager manual includes instructions for handling various knowledge object types via configuration files. For more information, see the documentation of those types.
For general information about configuration files in Splunk Enterprise, see the following topics in the Admin manual:

- About configuration files
- Configuration file precedence

The Admin Manual also contains a configuration file reference, which includes `.spec` and `.example` files for all the configuration files in Splunk Enterprise.

**Monitor and organize knowledge objects**

As a knowledge manager, you should periodically check up on the knowledge object collections in your Splunk deployment. You should be on the lookout for knowledge objects that:

- Fail to adhere to naming standards
- Are duplicates/redundant
- Are worthy of being shared with wider audiences
- Should be disabled or deleted due to obsolescence or poor design

Regular inspection of the knowledge objects in your system will help you detect anomalies that could become problems later on.

**Note:** This topic assumes that as a knowledge manager you have an *admin* role or a role with an equivalent permission set.

**Example - Keeping tags straight**

Most healthy Splunk deployments end up with a lot of *tags*, which are used to perform searches on clusters of field-value pairings. Over time, however, it’s easy to end up with tags that have similar names but which produce surprisingly dissimilar results. This can lead to considerable confusion and frustration.

Here’s a procedure you can follow for curating tags. It can easily be adapted for other types of knowledge objects handled through Splunk Web.

1. Go to **Settings > Tags > List by tag name**.
2. Look for tags with similar or duplicate names that belong to the same app (or which have been promoted to global availability for all users). For example, you might find a set of tags like `authentication` and `authentications` in the same app, where one tag is linked to an entirely
different set of field-value pairs than the other. Alternatively, you may encounter tags with identical names except for the use of capital letters, as in crash and Crash. Tags are case-sensitive, so Splunk software sees them as two separate knowledge objects. Keep in mind that you may find legitimate tag duplications if you have the App context set to All, where tags belonging to different apps have the same name. This is often permissible--after all, an authentication tag for the Windows app will have to be associated with an entirely different set of field-value pairs than an authentication for the UNIX app, for example.

3. Try to disable or delete the duplicate or obsolete tags you find, if your permissions enable you to do so. **However, be aware that there may be objects dependent on it that will be affected.** If the tag is used in reports, dashboard searches, other event types, or transactions, those objects will cease to function once the tag is removed or disabled. This can also happen if the object belongs to one app context, and you attempt to move it to another app context. For more information, see Disable or delete knowledge objects.

4. If you create a replacement tag with a new, more unique name, ensure that it is connected to the same field-value pairs as the tag that you are replacing.

**Using naming conventions to head off object nomenclature issues**

If you set up naming conventions for your knowledge objects early in your Splunk deployment you can avoid some of the thornier object naming issues. For more information, see Develop naming conventions for knowledge objects.

**The sequence of search-time operations**

When you run a search, the Splunk software runs several operations to derive various knowledge objects and apply them to the events returned by the search. These knowledge objects include extracted fields, calculated fields, lookup fields, field aliases, tags, and event types.

The Splunk software performs these operations in a specific sequence. This can cause problems if you configure something at the top of the process order with a definition that references the result of a configuration that is farther down in the process order.

**Search-time operations order example**
Consider calculated fields. Calculated field operations are in the middle of the search-time operation sequence. The Splunk software performs several other operations ahead of them, and it performs several more operations after them. Calculated fields derive new fields by running the values of fields that already exist in an event through an `eval` formula. This means that a calculated field formula cannot include fields in its formula that are added to your events by operations that follow it in the search-time operation sequence.

For example, when you design an `eval` expression for a calculated field, you can include extracted fields in the expression, because field extractions are processed at the start of the search-time operation sequence. By the time the Splunk software processes calculated fields, the field extractions exist and the calculated field operation can complete correctly.

However, an `eval` expression for a calculated field should never include fields that are added through a lookup operation. The Splunk software always performs calculated field operations ahead of lookup operations. This means that fields added through lookups at search time are unavailable when the Splunk software processes calculated fields. You will get an error message if your calculated field `eval` expression includes fields that are added through lookups.

**Search-time operation sequence**

The following table presents the search-time operation sequence as a list. After the list you can find more information about each operation in the sequence.

Each operation can have configurations that reference fields derived by operations that precede them in the sequence. However, those same configurations cannot contain fields that are derived by operations that follow them in the sequence.

All but one of these operations can be configured through Splunk Web, though some configuration options are only available by making manual `.conf` edits. You should make all manual file-based operation configurations on the search-head tier.

**Note:** This list does not include index-time operations, such as default and indexed field extraction. Index-time operations precede all search-time operations. See Index-time versus search time in *Managing Indexers and Clusters of Indexers*.

<table>
<thead>
<tr>
<th>Search-time</th>
<th>Operation</th>
<th>Can be</th>
<th>Location of file configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>operation order</td>
<td>name</td>
<td>configured via Splunk Web?</td>
<td>EXTRACT-&lt;class&gt; in a props.conf stanza</td>
</tr>
<tr>
<td>-----------------</td>
<td>------</td>
<td>-----------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>First</td>
<td>Inline field extraction (no field transform)</td>
<td>Yes</td>
<td>REPORT-&lt;class&gt; in a props.conf stanza.</td>
</tr>
<tr>
<td>Second</td>
<td>Field extraction that uses a field transform</td>
<td>Yes</td>
<td>props.conf stanzas, where KV_MODE is set to a valid value other than none. If no KV_MODE value is specified for a stanza, it is set to auto by default.</td>
</tr>
<tr>
<td>Third</td>
<td>Automatic key-value field extraction</td>
<td>No</td>
<td>FIELDALIAS-&lt;class&gt; in a props.conf stanza</td>
</tr>
<tr>
<td>Fourth</td>
<td>Field aliasing</td>
<td>Yes</td>
<td>EVAL-&lt;fieldname&gt; in a props.conf stanza</td>
</tr>
<tr>
<td>Fifth</td>
<td>Calculated fields</td>
<td>Yes</td>
<td>LOOKUP-&lt;class&gt; in a props.conf stanza.</td>
</tr>
<tr>
<td>Sixth</td>
<td>Lookups</td>
<td>Yes</td>
<td>eventtypes.conf stanza</td>
</tr>
<tr>
<td>Seventh</td>
<td>Event types</td>
<td>Yes</td>
<td>tags.conf stanza</td>
</tr>
<tr>
<td>Eighth</td>
<td>Tags</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

**Inline field extractions**

Inline field extractions are explicit field extractions that do not include a field transform reference. An explicit field extraction is a field extraction that is configured to extract a specific field or set of fields.

Each inline field extraction configuration is specific to events belonging to a particular host, source, or source type.

This operation does not include automatic key-value field extractions. Automatic key-value field extractions are their own operation category.

**Splunk Web management**
Create and manage inline field extractions in Settings. Navigate to **Settings > Fields > Field extractions**. You can also use the field extractor utility to design inline field extractions.

**Configuration information**

Create `EXTRACT-<class>` configurations within `props.conf` stanzas.

**Restrictions**

The Splunk software processes all inline field extractions belonging to a specific host, source, or source type in lexicographical order according to their `<class>` value. This means that you cannot reference a field extracted by `EXTRACT-aaa` in the field extraction definition for `EXTRACT-ZZZ`, but you can reference a field extracted by `EXTRACT-aaa` in the field extraction definition for `EXTRACT-ddd`. See Lexicographical processing of field extraction configurations.

Because they are at the top of the search-time operation sequence, inline field extraction configurations cannot reference fields that are derived and added to events by any other search-time operation.

**For more information**

In this manual:

- Build field extractions with the field extractor - The field extractor enables you to create inline field extractions in Splunk Web. It does not require that you understand how to write regular expressions.
- Use the Field Extractions page - For creating inline field extractions in Splunk Web through the Field Extractions page in Settings.
- Create and maintain search-time field extractions through configuration files - For configuring inline field extractions in `props.conf`.

**Field extraction that uses a field transform**

Field extraction configurations that reference a field transform are always processed by the Splunk software after it processes inline field extractions. Like inline field extractions, each transform-referencing field extraction is explicitly configured to extract a specific field or set of fields.

Each transform-referencing field extraction configuration is specific to events belonging to a particular host, source, or source type.
This operation does not include automatic key-value field extractions. Automatic key-value field extractions are their own operation category.

**Splunk Web management**

You can create and manage field extractions that use field transforms in Settings. Navigate to Settings > Fields and set the field extraction up using the Field Extractions and Field Transformations pages.

**Configuration information**

Create REPORT-<class> configurations within props.conf stanzas. The REPORT-<class> configurations include a reference to an additional configuration in transforms.conf.

**Restrictions**

The Splunk software processes all inline field extractions belonging to a specific host, source, or source type in lexicographical order according to their <class> value. This means that you cannot reference a field extracted by EXTRACT-aaa in the field extraction definition for EXTRACT-ZZZ, but you can reference a field extracted by EXTRACT-aaa in the field extraction definition for EXTRACT-ddd. See Lexicographical processing of field extraction configurations.

Transform-referencing field extraction configurations can reference fields that are extracted through inline field extraction operations. They cannot reference fields that are derived and added to events by automatic key-value field extractions and other operations that take place later in the search-time operation sequence.

**For more information**

In this manual:

- Use the field transformations page - For creating the transforms.conf part of a transform-referencing search-time field extraction.
- Use the field extractions page - For creating the props.conf part of a transform-referencing search-time field extraction.
- Create and maintain search-time field extractions through configuration files - For configuring transform-referencing field extractions in transforms.conf and props.conf.
**Automatic key-value field extraction**

A field extraction configuration that uses the `KV_MODE` attribute to automatically extract fields for events associated with a specific host, source, or source type.

Automatic key-value field extraction is not explicit, in that you cannot configure it to find a specific field or set of fields. It looks for any key=value patterns in events that it can find and extracts them as field/value pairs. It can be configured to extract fields from structured data formats like JSON, CSV, and table-formatted events.

Automatic key-value extraction always takes place after explicit field extraction methods (inline field extraction and transform--referencing field extraction).

**Splunk Web management**

There currently is not a way to configure automatic key-value field extractions in Splunk Web.

**Configuration information**

Set up automatic key-value field extractions for a specific host, source, or source type by finding (or creating) the appropriate stanza in `props.conf` and setting the `KV_MODE` attribute to `auto`, `auto_escaped`, `multi`, `json`, or `xml`.

When `KV_MODE` is not set for a `props.conf` stanza, that stanza has `KV_MODE=auto` by default. You have to set `KV_MODE=none` to disable automatic key-value field extraction for a specific host, source, or source type. When automatic key-value field extraction is disabled, explicit field extraction still takes place.

When `KV_MODE` is set to `auto` or `auto_escaped`, automatic JSON field extraction can take place alongside other automatic key/value field extractions. If you need to disable JSON field extraction without changing the `KV_MODE` value from `auto`, add `AUTO_KV_JSON=false` to the stanza. When not set, `AUTO_KV_JSON` defaults to `true`.

**Restrictions**

The Splunk software processes automatic key-value field extractions in the order that it finds them in events.

**For more information**
In this manual:

- Create and maintain search-time field extractions through configuration files - For configuring automatic key-value field extractions in `.conf` files.

**Field aliasing**

Field aliasing is the application of field alias configurations, which enable you to reference a single field in a search by multiple alternate names, or aliases.

Each field alias configuration is specific to events belonging to a particular host, source, or source type.

**Splunk Web management**

Create and manage field aliases in Settings. Navigate to **Settings > Fields > Field aliases**.

**Configuration information**

Create `FIELDALIAS-<class>` configurations in `props.conf` stanzas.

**Restrictions**

The Splunk software processes field aliases belonging to a specific host, source, or source type in lexicographical order. See Lexicographical processing of field extraction configurations.

You can create aliases for fields that are extracted at index time or search time. You cannot create aliases for fields that are added to events by search-time operations that follow the field aliasing process, like lookups and calculated fields.

**For more information**

- Create field aliases in Splunk Web
- Configure field aliases with `props.conf`

**Calculated fields**

Configurations that create one or more fields through the calculation of `eval` expressions and add those fields to events. The `eval` expression can use values of fields that are already present in the event due to index-time or search-time
field extraction processes.

Each calculated field configuration is specific to events belonging to a particular host, source, or source type.

**Splunk Web management**

You can create and manage calculated fields in Settings. Navigate to *Settings > Fields > Calculated fields*.

**Configuration information**

Create calculated fields by adding `EVAL-<fieldname>` configurations to `props.conf` stanzas.

**Restrictions**

All `EVAL-<fieldname>` configurations within a single `props.conf` stanza are processed in parallel, rather than in any particular sequence. This means you can’t "chain" calculated field expressions, where the evaluation of one calculated field is used in the expression for another calculated field.

Calculated fields can reference all types of field extractions as well as field aliases. They cannot reference lookups, event types, or tags.

**For more information**

In this manual:

- About calculated fields
- Create calculated fields with Splunk Web
- Configure calculated fields with `props.conf`

**Lookups**

Configurations that add fields from lookup tables to events when the lookup table fields are matched with one or more fields already present in those events. There are four distinct types of lookup configurations: CSV lookups, external lookups, KV Store lookups, and geospatial lookups.

Each lookup configuration is specific to events belonging to a particular host, source, or source type.
Splunk Web management

Create and manage your lookups in Settings. Navigate to Settings > Lookups.

Configuration information

Define lookups that automatically add fields to events in search results by creating a LOOKUP-<class> configuration in props.conf. Each LOOKUP-<class> includes a reference to a [<lookup_name>] stanza in transforms.conf.

Restrictions

The Splunk software processes lookups belonging to a specific host, source, or source type in lexicographical order. See Lexicographical processing of field extraction configurations.

Lookup configurations can reference fields that are added to events by field extractions, field aliases, and calculated fields. They cannot reference event types and tags.

For more information

In this manual:

• About lookups

Event types

Configurations that add event type field/value pairs to events that match the search strings that define the event types.

Splunk Web management

After you run a search, save it as an event type. You can also define and maintain event types in Settings. Navigate to Settings > Event types.

Configuration information

Configure event types in eventtypes.conf stanzas.

Restrictions
The Splunk software processes event types first by priority score and then by lexicographical order. So it processes all event types with a **Priority** of 1 first, and applies them to events in lexicographical order. Then it processes event types with a **Priority** of 2, and so on.

Search strings that define event types cannot reference tags. Event types are always processed and added to events before tags.

**For more information**

In this manual:

- Define event types in Splunk Web
- Automatically find and build event types
- Configure event types directly in eventtypes.conf

**Tags**

Configurations that add tags to specific field/value pairs in events.

**Splunk Web management**

You can add tags directly to field/value pairs in search results. You can also define and maintain tags in Settings. Navigate to **Settings > Tags**.

**Configuration information**

Configure tags in **tags.conf** stanzas.

**Restrictions**

The Splunk software applies tags to field/value pairs in events in lexicographical order, first by the field value, and then by the field name. See Lexicographical processing of field extraction configurations.

You can apply tags to any field/value pair in an event, whether it is extracted at index time, extracted at search time, or added through some other method, such as an event type, lookup, or calculated field.

**For more information**

In this manual:
• Tag field value pairs in Search
• Define and manage tags in Settings

**Lexicographical processing of knowledge object configurations**

The Splunk software processes the following knowledge objects in lexicographical order, according to the host, source, or source type they belong to:

• Inline field extractions
• Field extractions that use a field transform
• Field aliases
• Event types (after they are sorted according to priority)
• Lookups

The Splunk software also processes tags in lexicographical order, but they are not associated with a specific host, source, or source type.

Lexicographical order sorts items based on the values used to encode the items in computer memory. In Splunk software, this is almost always UTF-8 encoding, which is a superset of ASCII.

• Numbers are sorted before letters. Numbers are sorted based on the first digit. For example, the numbers 10, 9, 70, 100 are sorted lexicographically as 10, 100, 70, 9.
• Uppercase letters are sorted before lowercase letters.
• Symbols are not standard. Some symbols are sorted before numeric values. Other symbols are sorted before or after letters.

Splunk software also uses lexicographical ordering to determine configuration file precedence among app directories. See Configuration file precedence in the *Admin Manual*.

**Lexicographical order**

Lexicographical order sorts items based on the values used to encode the items in computer memory. In Splunk software, this is almost always UTF-8 encoding, which is a superset of ASCII.

• Numbers are sorted before letters. Numbers are sorted based on the first digit. For example, the numbers 10, 9, 70, 100 are sorted lexicographically as 10, 100, 70, 9.
• Uppercase letters are sorted before lowercase letters.
• Symbols are not standard. Some symbols are sorted before numeric values. Other symbols are sorted before or after letters.

**Example**

For example, the Splunk software extracts inline field extractions to a specific host, source, or source type in ASCII sort order. This means that when it processes inline field extractions belonging to the `access_combined_wcookies` source type, it processes an extraction called `REPORT-BBB` before `REPORT-ZZZ`, then process `REPORT-ZZZ` before `REPORT-aaa`, and so on.

This means that you cannot reference a field extracted by `REPORT-aaa` in the field extraction definition for `REPORT-BBB`.

For example, this configuration won't work because the `first_ten` field is extracted after the `first_two` field, due to field extraction process ordering (aaa < ZZZ).

```
[splunkd]
EXTRACT-aaa = ^(?<first_ten>.{10})
EXTRACT-ZZZ = (?<first_two>.{2}) in first_ten
```

This configuration will work because the `first_ten` field is extracted before the `first_two` field, due to field extraction process ordering (ZZZ > mmm).

```
[mongodb]
EXTRACT-ZZZ = ^(?<first_ten>.{10})
EXTRACT-mmm = (?<first_two>.{2}) in first_ten
```

Here is a search you can use to verify these configuration issues.

```
index=_internal (sourcetype=splunkd OR sourcetype=mongodb) | stats values(first_ten) values(first_two) by sourcetype
```

**More information about process order within a single props.conf file**

The *Admin Manual* contains several topics about configuration file administration. One of these topics, Attribute precedence within a single props.conf file, may be of particular interest to those interested in knowledge object processing order. It discusses the following topics.
• Precedence between sets of stanzas affecting the same host, source, or source type.
• Overriding the default lexicographical order in props.conf.
• Precedence for events with multiple attribute assignments.

Give knowledge objects of the same type unique names

When you create knowledge objects that are processed at search time, it is best if all knowledge objects of a single type have unique names. This helps you avoid name collision issues that can prevent your configurations from being applied at search time.

For example, to avoid name collision problems, you should not have two inline field extraction configurations that have the same <class> value in your Splunk implementation. However, you can have an inline field extraction, a transform field extraction, and a lookup that share the same name, because they belong to different knowledge object types.

You can avoid these problems with knowledge object naming conventions. See Develop naming conventions for knowledge objects.

Configurations sharing a host, source, or source type

When two or more configurations of a particular knowledge object type share the same props.conf stanza, they share the host, source, or source type identified for the stanza. If each of these configurations has the same name, then the last configuration listed in the stanza overrides the others.

For example, say you have two lookup configurations named LOOKUP-table in a props.conf stanza that is associated with the sendmail source type:

```
[sendmail]
LOOKUP-table = logs_per_day host OUTPUTNEW average_logs AS logs_per_day
LOOKUP-table = location host OUTPUTNEW building AS location
```

In this case, the last LOOKUP-table configuration in that stanza overrides the one that precedes it. The Splunk software adds the location field to your matching events, but does not add the logs_per_day field to any of them.

When you name your lookup LOOKUP-table, you are saying this is the lookup that achieves some purpose or action described by "table." In this example, these
lookups achieve different goals. One lookup determines something about logs per day, and the other lookup has something to do with location. Rename them.

```
[sendemail]
LOOKUP-table = logs_per_day host OUTPUTNEW average_logs AS logs_per_day
LOOKUP-location = location host OUTPUTNEW building AS location
```

Now you have two different configurations that do not collide.

**Configurations belonging to different hosts, sources or source types**

You can also run into name collision issues when the configurations involved do not share a specific host, source, or source type.

For example, if you have lookups with different hosts, sources, or source types that share the same name, you can end up with a situation where only one of them seems to work at any given time. If you know what you are doing you might set this up on purpose, but in most cases it is inconvenient.

Here are two lookup configurations named `LOOKUP-splk_host`. They are in separate `props.conf` stanzas.

```
[host::machine_name]
LOOKUP-splk_host = splk_global_lookup search_name OUTPUTNEW global_code

[sendmail]
LOOKUP-splk_host = splk_searcher_lookup search_name OUTPUTNEW search_code
```

Any events that overlap between these two lookups are only affected by one of them.

- Events that match the host get the host lookup.
- Events that match the source type get the source type lookup.
- Events that match both get the host lookup.

For more information about this, see Configuration file precedence in the Admin Manual.

**Configurations belonging to different apps**

When you have configurations that belong to the same knowledge object type and share the same name, but belong to different apps, you can also run into
naming collisions. In this case, the configurations are applied in reverse lexicographical order of the app directories.

For example, say you have two field alias configurations.

The first configuration is in etc/apps/splk_global_lookup_host/local/props.conf:

```
[host::*]
FIELDALIAS-sshd = sshd1_code AS global_sshd1_code
```

The second configuration is in etc/apps/splk_searcher_lookup_host/local/props.conf:

```
[host::*]
FIELDALIAS-sshd = sshd1_code AS search_sshd1_code
```

In this case, the `search_sshd1_code` alias would be applied to events that match both configurations, because the app directory `splk_searcher_lookup_host` comes up first in the reverse lexicographical order. To avoid this, you might change the name of the first field alias configuration to `FIELDALIAS-global_sshd`.

**Lexicographical order**

Lexicographical order sorts items based on the values used to encode the items in computer memory. In Splunk software, this is almost always UTF-8 encoding, which is a superset of ASCII.

- Numbers are sorted before letters. Numbers are sorted based on the first digit. For example, the numbers 10, 9, 70, 100 are sorted lexicographically as 10, 100, 70, 9.
- Uppercase letters are sorted before lowercase letters.
- Symbols are not standard. Some symbols are sorted before numeric values. Other symbols are sorted before or after letters.

**Develop naming conventions for knowledge objects**

As a best practice, develop naming conventions for your knowledge objects when it makes sense to do so. If the naming conventions you develop are followed consistently by all of the Splunk users in your organization, you will find that they become easier to use and that their purpose is much easier to discern at a glance.
You can develop naming conventions for just about every kind of knowledge object in your Splunk deployment. Naming conventions can help with object organization, but they can also help users differentiate between groups of reports, event types, and tags that have similar uses. And they can help identify a variety of things about the object that may not even be in the object definition, such as what teams or locations use the object, what technology it involves, and what it is designed to do.

Early development of naming conventions for your Splunk deployment will help you avoid confusion and chaos later on down the road.

**Example - Set up a naming convention for reports**

You work in the systems engineering group of your company, and as the knowledge manager for your Splunk deployment, it is your job to define a naming convention for the reports produced by your team.

You develop a naming convention that combines:

- **Group**: Corresponds to the working group(s) of the user saving the search.
- **Search type**: Indicates the type of search (alert, report, summary-index-populating).
- **Platform**: Corresponds to the platform subjected to the search.
- **Category**: Corresponds to the concern areas for the prevailing platforms.
- **Time interval**: The interval over which the search runs (or on which the search runs, if it is a scheduled search).
- **Description**: A meaningful description of the context and intent of the search, limited to one or two words if possible. Ensures the search name is unique.

<table>
<thead>
<tr>
<th>Group</th>
<th>Search type</th>
<th>Platform</th>
<th>Category</th>
<th>Time interval</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEG</td>
<td>Alert</td>
<td>Windows</td>
<td>Disk</td>
<td>&lt;arbitrary&gt;</td>
<td>&lt;arbitrary&gt;</td>
</tr>
<tr>
<td>NEG</td>
<td>Report</td>
<td>iSeries</td>
<td>Exchange</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPS</td>
<td>Summary</td>
<td>Network</td>
<td>SQL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOC</td>
<td></td>
<td></td>
<td>Event log</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Jobs</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Subsystems</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Services</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Possible reports using this naming convention:

- SEG_Alert_Windows_Eventlog_15m_Failures
- SEG_Report_iSeries_Jobs_12hr_Failed_Batch
- NOC_Summary_Network_Security_24hr_Top_src_ip

**Understand and use the Common Information Model Add-on**

The Common Information Model Add-on is based on the idea that you can break down most log files into two components:

- fields
- event category tags

With these two components, a knowledge manager can normalize log files at search time so that they follow a similar schema. The Common Information Model details the standard fields and event category tags that Splunk software uses when it processes most IT data.

In the past, the Common Information Model was represented here as a set of tables that you could use to normalize your data by ensuring that they were using the same field names and event tags for equivalent events from different sources or vendors.

Now, the Common Information Model is delivered as an add-on that implements the CIM tables as **data models**. You can use these data models in two ways:

- Initially, you can use them to test whether your fields and tags have been normalized correctly.
- After you have verified that your data is normalized, you can use the models to generate reports and dashboard panels via Pivot.

You can download the Common Information Model Add-on from Splunkbase here. For a more in-depth overview of the CIM add-on, see the Common Information Model Add-on product documentation.
Manage knowledge object permissions

This topic assumes that as a knowledge manager you have an "admin" role or a role with an equivalent permission set.

As a Knowledge Manager, you can set knowledge object permissions to restrict or expand access to the variety of knowledge objects in your Splunk deployment.

In some cases you'll determine that certain specialized knowledge objects should only be used by people in a particular role, within a specific app. And in others you'll move to the other side of the scale and make universally useful knowledge objects globally available to all users in all apps. As with all aspects of knowledge management you'll want to carefully consider the implications of these access restrictions and expansions.

When a Splunk user first creates a new report, event type, transaction, or similar knowledge object, it is only available to that user. To make that object available to more people, Splunk Web provides the following options, which you can take advantage of if your permissions enable you to do so. You can:

- Make the knowledge object available globally to users of all apps (also referred to as "promoting" an object).
- Make the knowledge object available to all users of an app.
- Restrict (or expand) access to global or app-specific objects by user or role.
- Set read/write permissions at the app level for roles, to enable users to share or delete objects they do not own.

By default, only users with a power or admin role can share and promote knowledge objects. This makes you and your fellow knowledge managers gatekeepers with approval capability over the sharing of new knowledge objects.

For more information about extending the ability to set permissions to other roles, see the subtopic "Enabling roles other than Admin and Power to set permissions and share objects," below.

How do permissions affect knowledge object usage?

To illustrate how these choices can affect usage of a knowledge object, imagine that Bob, a user of a (fictional) Network Security app with an admin-level "Firewall Manager" role, creates a new event type named firewallbreach, which finds events that indicate firewall breaches. Here's a series of
permissions-related issues that could come up, and the actions and results that would follow:

<table>
<thead>
<tr>
<th>Issue</th>
<th>Action</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>When Bob first creates firewallbreach, it is only available to him. Other users cannot see it or work with it. Bob decides he wants to share it with his fellow Network Security app users.</td>
<td>Bob updates the permissions of the firewallbreach event type so that it is available to all users of the Network Security app, regardless of role. He also sets up the new event type so that all Network Security users can edit its definition.</td>
<td>Anyone using Splunk Enterprise in the Network Security app context can see, work with, and edit the firewallbreach event type. Users of other apps in the same Splunk deployment have no idea it exists.</td>
</tr>
<tr>
<td>A bit later on, Mary, the knowledge manager, realizes that only users in the Firewall Manager role should have the ability to edit or update the firewallbreach event type.</td>
<td>Mary restricts the ability to edit the event type to the Firewall Manager role.</td>
<td>Users of the Network Security app can use the firewallbreach event type in transactions, searches, dashboards, and so on, but now the only people that can edit the knowledge object are those with the Firewall Manager role and people with admin-level permissions (such as the knowledge manager). Splunk users in other app contexts remain blissfully ignorant of the event type.</td>
</tr>
<tr>
<td>At some point a few people who have grown used to</td>
<td>They make their case to the knowledge manager, who</td>
<td>Now, everyone that uses this Splunk</td>
</tr>
</tbody>
</table>
using the very handy firewallbreach event type in the Network Security app decide they’d like to use it in the context of the Windows app as well. promptly promotes the firewallbreach event type to global availability. deployment can use the firewallbreach event type, no matter what app context they happen to be in. But the ability to update the event type definition is still confined to admin-level users and users with the Firewall Manager role.

Permissions - Getting started

To change the permissions for a knowledge object, follow these steps:

1. In Splunk Web, navigate to the page for the type of knowledge object that you want to update permissions for.
2. Find the knowledge object that you created (use the filtering fields at the top of the page if necessary) and open its permissions dialog.
   ♦ In some cases you will need to click a Permissions link to do this. In other cases you need to make a menu selection such as Edit > Edit Permissions or Manage > Edit Permissions.
   ♦ If you are on a listing page you can also expand the object row and click Edit for Permissions.
3. On the Permissions page for the knowledge object in question, perform the actions in the following subsections depending on how you'd like to change the object's permissions.
4. Click Save to save your changes.
Make an object available to users of all apps

To make an object globally available to users of all apps in your Splunk deployment:

1. Navigate to the Permissions page for the knowledge object (following the instructions above).
2. For Display for, select All apps.
3. In the Permissions section, for Everyone, select a permission of either Read or Write.

<table>
<thead>
<tr>
<th>Option</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>Lets users see and use the object, but not update its definition. When users only have Read permission for a particular report, they can see it in the top level navigation and they can run it. But they can't update the search string, change its time range, or save their changes.</td>
</tr>
<tr>
<td>Write</td>
<td>Lets users view, use, and update the defining details of an object as necessary.</td>
</tr>
</tbody>
</table>

If neither Read or Write is selected then users cannot see or use the knowledge object.
4. Save the permission change.

Make an object available to all users of its app

All knowledge objects are associated with an app. When you create a new knowledge object, it is associated with the app context that you are in at the time. In other words, if you are using the Search & Reporting app when you create the
object, the object will be listed in Settings with Search & Reporting as its App column value. This means that if you restrict its sharing permissions to the app level it will only be available to users of the Search & Reporting app.

When you create a new object, you are given the option of keeping it private, sharing it with users of the app that you’re currently using, or sharing it globally with all users. Opt to make the app available to "this app only" to restrict its usage to users of that app, when they are in that app context.

If you have write permissions for an object that already exists, you can change its permissions so that it is only available to users of its app by following these steps.

1. Navigate to the Permissions page for the knowledge object (following the instructions in "Permissions - Getting Started," above).
2. For Display for, select App.
3. In the Permissions section, for Everyone, select a permission of either Read or Write.

<table>
<thead>
<tr>
<th>Option</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>Lets users see and use the object, but not update its definition. When users only have Read permission for a particular report, they can see it in the top level navigation and they can run it. But they can't update the search string, change its time range, or save their changes.</td>
</tr>
<tr>
<td>Write</td>
<td>Lets users view, use, and update the defining details of an object as necessary.</td>
</tr>
</tbody>
</table>

If neither Read or Write is selected then users cannot see or use the knowledge object.

4. Save the permission change.

**Moving or cloning a knowledge object**

You may run into situations where you want users of an app to be able to access a particular knowledge object that belongs to a different app, but you do not want to share that object globally with all apps. There are two ways you can do this: by cloning the object, or by moving it.

<table>
<thead>
<tr>
<th>Option</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Make a copy of a knowledge object. The copy has all of the same settings as the original object, which you can keep or modify. You can keep it in the same app as the object you’re cloning, or you can put it in another app.</td>
</tr>
</tbody>
</table>
a new app. If you add the cloned object to the same app as the original, give it a different name. You can keep the original name if you add the object to an app that doesn't have a knowledge object of the same type with that name. You can clone any object, even if your role does not have write permissions for it.

Move an existing knowledge object to another app. Removes the object from its current app and places it in an app that you determine. Once there, you can set its permissions so that it is private, globally available, or only available to users of that app. The ability to move an app is connected to the same permissions that determine whether you can delete an app. You can only move a knowledge object if you have created that object and have write permissions for the app to which it belongs.

Switching the app context of a knowledge object by moving it can have downstream consequences for objects that have been associated with it. See Disable or delete knowledge objects.

You can find the Clone and Move controls on the Settings pages for various knowledge object types. To clone or move an object, find the object in its list and click Clone or Move.

Restrict knowledge object access by app and role

You can use this method to lock down various knowledge objects from alteration by specific roles. You can arrange things so users in a particular role can use the knowledge object but not update it--or you can set it up so those users cannot see the object at all. In the latter case, the object will not show up for them in Splunk Web, and they will not find any results when they search on it.

If you want restrict the ability to see or update a knowledge object by role, simply navigate to the Permissions page for the object. If you want members of a role to:

- Be able to use the object and update its definition, give that role Read and Write access.
- Be able to use the object but be unable to update it, give that role Read access only (and make sure that Write is unchecked for the Everyone role).
- Be unable to see or use the knowledge object at all, leave Read and Write unchecked for that role (and unchecked for the Everyone role as well).
About role-based user access in *Securing Splunk Enterprise*.

**Enable a role other than Admin and Power to set permissions and share objects**

By default, only the Power and Admin roles can set permissions for knowledge objects. Follow these steps to give another role the ability to set knowledge object permissions.

The ability to set permissions for knowledge objects is controlled at the app level. It is not connected to a role capability like other actions such as scheduling searches or changing default input settings. You have to give roles write permissions to an app to enable people with those roles to manage the permissions of knowledge objects created in the context of that app.

**Steps**

1. From the Splunk Home page, select any app in the Apps Panel to open the app.
2. Click on the Applications menu in the Splunk bar, and select **Manage Apps**.
3. Find the app that you want to adjust permissions for and open its **Permissions** settings.
4. On the Permissions page for the app, give the role **Read** and **Write** permissions.
5. Click **Save** to save your changes.

Users whose roles have write permissions to an app can also delete knowledge objects that are associated with that app. For more information, see Disable or delete knowledge objects.

**Set permissions for categories of knowledge objects**

You can set role-based permissions for specific knowledge object types by making changes to the **default.meta** file. For example, you can give all user roles the ability to set permissions for all saved searches in a specific app.

See Set permissions for objects in a Splunk App on the *Splunk Developer Portal*. 
About deleting users who own knowledge objects

If you delete a user from your Splunk deployment, the objects that user owns become orphaned. Orphaned objects can have serious implications. For example, when a scheduled report or alert becomes orphaned, it ceases to run on its schedule. When this happens, your team can miss important alerts, actions that are tied to affected scheduled reports will cease to function, and any dashboard panels that are associated with affected scheduled reports will stop showing search results.

To prevent this from happening, reassign knowledge objects to another user. For more information see Manage orphaned knowledge objects.

Manage orphaned knowledge objects

When a knowledge object owner leaves a department or company and their Splunk account is deactivated, the knowledge objects that they owned remain in the system. These are orphaned knowledge objects. Knowledge objects without valid owners can cause problems. For example, searches that refer to orphaned lookup definitions may not work.

Orphaned scheduled reports can be a particular problem. The search scheduler cannot run a scheduled report on behalf of a nonexistent owner. This happens because the scheduled report is no longer associated with a role. Without that role association, the search scheduler has no way of knowing what search quotas and concurrent search configurations the report is limited by. As a result, it will not run the scheduled report on its schedule at all. This can result in broken dashboards and embedded searches, data collection gaps in summary indexes, and more.

Orphaned knowledge objects also present a security concern, whether they are scheduled or not. There are a variety of reasons that you should not allow knowledge objects to operate on behalf of owners who are no longer in your system.

The Splunk software provides several methods of detecting orphaned knowledge objects. Once you have found orphaned knowledge objects, you have several options for resolving their orphaned status.
Find orphaned knowledge objects

There are several ways that you can find out whether or not you have orphaned knowledge objects in your Splunk implementation. Most of these detection methods specialize in orphaned scheduled reports, because they tend to cause the most problems for users.

The Reassign Knowledge Objects page in Settings is the only orphaned knowledge object detection method that can find all orphaned knowledge object types. It can only find orphaned knowledge objects that have been shared at the app or global levels.

These detection methods have no way of knowing when people are removed using a third-party user authentication system.

*Review orphaned scheduled search notifications*

By default your Splunk implementation runs a search to find orphaned scheduled reports on a daily schedule. When it finds orphaned scheduled reports it creates a notification message. If you open that message you can click a link to see a list of the orphaned reports in the Orphaned Scheduled Searches, Reports, and Alerts dashboard.

*Steps*

1. Click **Messages** when the UI indicates there are messages there.

2. If you find a message indicating that the Splunk software has found orphaned scheduled searches, click the message link to run a search that displays the orphaned scheduled searches.

*Look at the Orphaned Scheduled Searches, Reports, and Alerts dashboard and report*

The Orphaned Scheduled Searches, Reports, and Alerts dashboard is delivered with your Splunk implementation. When you access it, the dashboard loads with the results of the Orphaned Searches, Reports and Alerts report, which is designed to return the names of any orphaned scheduled reports in your system.
You can run the Orphaned Searches, Reports And Alerts report directly from the Reports listing page to get the same results.

**Run the Monitoring Console health check**

If you have the Monitoring Console configured for your Splunk instance, you can use its health check feature to detect orphaned scheduled searches, reports, and alerts. It will tell you how many of these knowledge objects exist in your system. You have to run a drilldown search to see a list that identifies the orphaned searches by name.

**Prerequisites**

- Access and customize health check in the *Admin Manual*.

**Steps**

1. Open the Monitoring Console by selecting **Settings > Monitoring Console**.
2. Select the **Health Check** tab.
3. Click the **Start** button in the upper right corner to run the health check. By default, the health check will search for orphaned scheduled searches, reports, and alerts. If the health check finds orphaned searches, the Monitoring Console marks the Orphaned Scheduled Searches row with an Error notification moves it to the top of the health check list, alongside other rows with Error notifications.
4. Click the Orphaned Scheduled Searches row to launch a drill down search. This search displays the names of the orphaned scheduled searches.

**Use the Reassign Knowledge Objects page in Settings**

The Reassign Knowledge Objects page in Settings is the only orphaned knowledge object detection method that detects all knowledge objects (not just searches, reports, and alerts). However, it can only find knowledge objects that
have been shared to the app or global levels.

**Steps**

1. Select **Settings > All configurations**.
2. Click **Reassign Knowledge Objects**.
3. Click **Orphaned** to filter out non-orphaned objects from the list.

At this point, the list should only contain orphaned objects that have been shared. Now you can determine what you want to do with the items in that list.

**Reassign one or more shared knowledge objects to a new owner**

Use the Reassign Knowledge Objects page in Settings to reassign a knowledge object to a new owner. The Reassign Knowledge Objects page can reassign both owned and orphaned knowledge objects. It is designed to work with all Splunk deployments, including those that use **search head clustering (SHC)**.

The Reassign Knowledge Objects page cannot reassign knowledge objects that are both orphaned and privately shared. See Reassign private orphaned knowledge objects.

Knowledge object ownership changes can have side effects such as giving saved searches access to previously inaccessible data or making previously available knowledge objects unavailable. Review your knowledge objects before you reassign them.

Only users with the Admin role can reassign knowledge objects to new owners.

**Find the knowledge object or objects you want to reassign**

First, you need to use the filtering options on the Reassign Knowledge Objects page to help you find the knowledge object or objects that you want to reassign.

**Steps**

1. Select **Settings > All configurations**.
2. Click **Reassign Knowledge Objects**.
3. Find the object or objects that you want to reassign.

<table>
<thead>
<tr>
<th>Objects to find</th>
<th>Step to follow</th>
</tr>
</thead>
</table>

37
<table>
<thead>
<tr>
<th>Objects belonging to an owner with an active account.</th>
<th>Click <strong>Filter by Owner</strong> and select the owner name from the dropdown.</th>
</tr>
</thead>
<tbody>
<tr>
<td>All shared, orphaned objects.</td>
<td>Click <strong>Orphaned</strong>.</td>
</tr>
<tr>
<td>Objects belonging to a specific knowledge object type.</td>
<td>Make a selection from the <strong>Object type</strong> dropdown.</td>
</tr>
<tr>
<td>Objects belonging to a specific app.</td>
<td>Make a selection from the <strong>App</strong> dropdown. You can optionally switch <strong>All Objects</strong> to <strong>Objects created in the app</strong> to filter out objects created in apps other than the app you have selected.</td>
</tr>
<tr>
<td>Objects that include a particular text string.</td>
<td>Enter the string into the <strong>filter</strong> field and click <strong>Return</strong>.</td>
</tr>
</tbody>
</table>

Your next steps depend on how many knowledge objects you want to reassign to a different owner.

**Reassign a single knowledge object to another owner**

If you are using the Reassign Knowledge Objects page to reassign an individual object to another owner, follow these steps.

**Prerequisites**

Find the knowledge object you want to reassign before starting this task.

**Steps**

1. For the knowledge object that you want to reassign, click **Reassign** in the **Action** column.
2. Click **Select an owner** and select the name of the person that you want to reassign the knowledge object to.
3. Click **Save** to save your changes.

**Bulk-reassign multiple knowledge objects to another owner**

If you are using the Reassign Knowledge Objects page to reassign multiple objects to another owner, follow these steps.
Prerequisites

Find the knowledge objects you want to reassign before starting this task.

Steps

1. Select the checkboxes next to the objects that you want to reassign. If you want to reassign all objects in the list to the same owner, click the checkbox at the top of the checkbox column. You can reassign up to 100 objects in one bulk reassignment action.
2. Click Edit Selected Knowledge Objects and select Reassign.
3. (Optional) Remove knowledge objects that you have accidentally selected by clicking the X symbols next to their names.
4. Click Select an owner and select the name of the person that you want to bulk-reassign the selected knowledge objects to.
5. Click Save to save your changes.

Reassign unshared, orphaned knowledge objects

If you want to reassign orphaned knowledge objects that had a Private sharing status when they were orphaned, you cannot reassign them through the UI, or through REST API calls. There are two ways to reassign unshared, orphaned knowledge objects. You can temporarily recreate the invalid owner, or you can copy and paste the knowledge object stanza between the configuration files of the invalid and valid owners.

Temporarily recreate the invalid owner

The easiest solution for this is to temporarily recreate the invalid owner account, reassign the knowledge object, and then deactivate the invalid owner account.

Prerequisites

See About users and roles in the Admin Manual to learn how to add and remove users from your Splunk implementation.

Steps

1. Add the invalid knowledge object owner as a new user in your Splunk deployment.
2. Use the Reassign Knowledge Objects page to assign the knowledge object to a different active owner.
3. Deactivate the invalid owner account.
Perform a knowledge object stanza copy and paste operation between two .conf files

If you cannot reactivate invalid owner accounts, you can transfer ownership of unshared and orphaned knowledge objects by performing a .conf file stanza cut and paste operation. You cut the knowledge object stanza out of a .conf file belonging to the invalid owner and paste it into the corresponding .conf file of a valid owner.

Prerequisites

To use this method you must meet the following requirements:

- You must be using either Splunk Enterprise or Splunk Light.
- You must have filesystem access.
- You cannot be running SHC on your Splunk deployment.
- You must be able to restart your Splunk deployment.

Steps

1. In the filesystem of your Splunk deployment, open the the .conf file for the invalid owner at 
   etc/users/<name_of_invalid_user>/search/local/<name_of_conf_file>.
2. Locate the stanza for the orphaned knowledge object and cut it out.
3. Save your changes to the file and close it.
4. Open the the corresponding .conf file for the new owner at 
   etc/users/<name_of_valid_user>/search/local/<name_of_conf_file>.
5. Copy the knowledge object stanza that you just cut to the .conf file for the valid owner.
6. Save your changes to the file and close it.
7. Restart your Splunk deployment so the changes take effect.

About resolving orphaned scheduled searches

The action you take to resolve an orphaned scheduled search, report, or alert depends on what you want it to do going forward.

<table>
<thead>
<tr>
<th>Objective</th>
<th>Step to follow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keep the search running on its schedule</td>
<td>Reassign the search to a new owner.</td>
</tr>
<tr>
<td>Let the search run on an ad-hoc basis.</td>
<td>Remove the schedule for the search from its definition in Settings &gt; Searches, reports, and alerts.</td>
</tr>
</tbody>
</table>
search has been shared with other users of an app, users of that app can run it. This can be important if it is used in a dashboard, for example.

However, you may need to ensure that other users do not schedule it again. You can do this by limiting the number of roles that have edit access to the search.

Alternatively, you can reassign it to a new owner as an unscheduled search.

<table>
<thead>
<tr>
<th>Keep the search from running again under any circumstances.</th>
<th>Disable the search, or delete it.</th>
</tr>
</thead>
</table>

**Turn off notifications of orphaned searches**

By default, Splunk software notifies you about orphaned searches. If you would rather not receive these notifications, open limits.conf, look for the [system_checks] stanza, and set orphan_searches to disabled.

**Disable or delete knowledge objects**

Your ability to delete knowledge objects in Splunk Web depends on a set of factors:

- **You cannot delete default knowledge objects that were delivered with Splunk software (or with an app).**
  If the knowledge object definition resides in a default directory, it can’t be removed through Splunk Web. It can be disabled by clicking Disable for the object in Settings. Only objects that exist in an app’s local directory are eligible for deletion.

- **You can always delete knowledge objects that you have created, and which haven’t been shared by you or someone with admin-level permissions.**
  Once you share a knowledge object you’ve created with other users, your ability to delete it is revoked, unless you have write permissions for the app to which they belong.

- **To delete any other knowledge object, your role must have write permissions for the app to which the knowledge object belongs and the knowledge object itself.**
  This applies to knowledge objects that are shared globally as well as those that are only shared within an app. All knowledge objects belong to
a specific app, no matter how they are shared.

App-level write permissions are usually only granted to users with admin-equivalent roles.

**Note:** If a role does not have write permissions for an app but does have write permissions for knowledge objects belonging to that app, it can disable those knowledge objects. Clicking **Disable** for a knowledge object has the same function as knowledge object deletion, with the exception that Splunk software does not remove disabled knowledge objects from the system. A role with write permissions for a disabled knowledge object can re-enable it at any time.

There are similar rules for data models. To enable a role to create data models and share them with others, the role must be given write access to an app. This means that users who can create and share data models can potentially also delete knowledge objects. For more information, see Manage data models.

**Grant a role write permissions for an app**

If your role has admin-level permissions, you can grant a role write permissions for an app in Splunk Web.

**Steps**

1. From the Splunk Home page, select any app in the Apps Panel to open the app.
2. Click on the Applications menu in the Splunk bar, and select **Manage Apps**.
3. Find the app that you want to adjust permissions for and open its **Permissions** settings.
4. Select **Write** for the roles that should be able to delete knowledge objects for the app.
5. Click **Save** to save your changes.

Users whose roles have write permissions to an app can also edit knowledge objects that are associated with that app. For more information, see Manage knowledge object permissions.

You can also manage role-based permissions for an app by updating its **local.meta** file. For more information see Setting access to manager consoles and apps in *Securing Splunk Enterprise*. 

42
Grant a role with app write permissions the ability to delete a knowledge object that belongs to that app

Once a role has write permissions for an app, users with that role can delete any knowledge object belonging to that app as long as they also have write permissions for those objects. Users can do this whether the knowledge object is shared just to the app, or globally to all apps. Even when knowledge objects are shared globally they belong to a specific app.

Prerequisites

- Your role has admin-level permissions.
- The role that you are setting object-level permissions for has the ability to write to the app that the object belongs to.

Steps

1. Navigate to the listing page for the knowledge object in Settings.
2. To ensure that you are viewing objects that belong to the app for which the role has write permissions, select the app name or All for the App context.
3. On the listing page, locate the knowledge object that the role needs to be able to delete and click its Permissions link.
4. On the permissions page for the knowledge object, select Write for the role.

If you have followed this procedure and the procedure that came before it you the role should be able to delete the knowledge object.

Deleting knowledge objects with downstream dependencies

You have to be careful about deleting knowledge objects with downstream dependencies, as this can have negative impacts.

For example, you could have a tag that looks like the duplicate of another, far more common tag. On the surface, it would seem to be harmless to delete the dup tag. But what you may not realize is that this duplicate tag also happens to be part of a search that a very popular event type is based upon. And that popular event type is used in two important reports--the first is the basis for a well-used dashboard panel, and the other is used to populate a summary index that is used by searches that run several other dashboard panels. So if you delete that tag, the event type breaks, and everything downstream of that event
This is why it is important to nip poorly named or defined knowledge objects in the bud before they become inadvertently hard-wired into the workings of your deployment. The only way to identify the downstream dependencies of a particular knowledge object is to search on it, find out where it is used, and then search on those things to see where they are used—it can take a bit of detective work. There is no "one click" way to bring up a list of knowledge object downstream dependencies at this point.

If you really feel that you have to delete a knowledge object, and you're not sure if you've tracked down and fixed all of its downstream dependencies, you could try disabling it first to see what impact that has. If nothing seems to go seriously awry after a day or so, delete it.

Deleting knowledge objects in configuration files

Note that in Splunk Web, you can only disable or delete one knowledge object at a time. If you need to remove large numbers of objects, the most efficient way to do it is by removing the knowledge object stanzas directly through the configuration files. Keep in mind that several versions of a particular configuration file can exist within your system. In most cases you should only edit the configuration files in $SPLUNK_HOME/etc/system/local/, to make local changes on a site-wide basis, or $SPLUNK_HOME/etc/apps/<App_name>/local/, if you need to make changes that apply only to a specific app.

Do not try to edit configuration files until you have read and understood the following topics in the Admin manual:

- About configuration files
- Configuration file precedence

About Splunk regular expressions

This primer helps you create valid regular expressions. For a discussion of regular expression syntax and usage, see an online resource such as www.regular-expressions.info or a manual on the subject.

Regular expressions match patterns of characters in text and are used for extracting default fields, recognizing binary file types, and automatic assignation of source types. You also use regular expressions when you define custom field
extractions, filter events, route data, and correlate searches. Search commands that use regular expressions include `rex` and `regex` and evaluation functions such as `match` and `replace`.

Splunk regular expressions are PCRE (Perl Compatible Regular Expressions) and use the PCRE C library.

**Regular expressions terminology and syntax**

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>literal</td>
<td>The exact text of characters to match using a regular expression.</td>
</tr>
<tr>
<td>regular expression</td>
<td>The metacharacters that define the pattern that Splunk software uses to match against the literal.</td>
</tr>
<tr>
<td>groups</td>
<td>Regular expressions allow groupings indicated by the type of bracket used to enclose the regular expression characters. Groups can define character classes, repetition matches, named capture groups, modular regular expressions, and more. You can apply quantifiers to and use alternation within enclosed groups.</td>
</tr>
<tr>
<td>character class</td>
<td>Characters enclosed in square brackets. Used to match a string. To set up a character class, define a range with a hyphen, such as <code>[A-Z]</code>, to match any uppercase letter. Begin the character class with a caret (^) to define a negative match, such as <code>[^A-Z]</code> to match any lowercase letter.</td>
</tr>
<tr>
<td>character type</td>
<td>Similar to a wildcard, character types represent specific literal matches. For example, a period . matches any character, \w matches words or alphanumeric characters including an underscore, and so on.</td>
</tr>
<tr>
<td>anchor</td>
<td>Character types that match text formatting positions, such as return (\r) and newline (\n).</td>
</tr>
<tr>
<td>alternation</td>
<td>Refers to supplying alternate match patterns in the regular expression. Use a vertical bar or pipe character (</td>
</tr>
<tr>
<td>quantifiers, or repetitions</td>
<td>Use ( *, +, ? ) to define how to match the groups to the literal pattern. For example, * matches 0 or more, +</td>
</tr>
</tbody>
</table>
matches 1 or more, and ? matches 0 or 1.

**back references**
Literal groups that you can recall for later use. To indicate a back reference to the value, specify a dollar symbol ($) and a number (not zero).

**lookarounds**
A way to define a group to determine the position in a string. This definition matches the regular expression in the group but gives up the match to keep the result. For example, use a lookaround to match \( x \) that is followed by \( y \) without matching \( y \).

**Character types**

Character types are short for literal matches.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
<th>Example</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>\w</td>
<td>Match a word character (a letter, number, or underscore character).</td>
<td>\w\w\w</td>
<td>Matches any three word characters.</td>
</tr>
<tr>
<td>\W</td>
<td>Match a non-word character.</td>
<td>\W\W\W</td>
<td>Matches any three non-word characters.</td>
</tr>
<tr>
<td>\d</td>
<td>Match a digit character.</td>
<td>\d\d\d-\d\d\d-\d\d\d\d</td>
<td>Matches a Social Security number, or a similar 3-2-4 number string.</td>
</tr>
<tr>
<td>\D</td>
<td>Match a non-digit character.</td>
<td>\D\D\D</td>
<td>Matches any three non-digit characters.</td>
</tr>
<tr>
<td>\s</td>
<td>Match a whitespace character.</td>
<td>\d\s\d</td>
<td>Matches a sequence of a digit, a whitespace, and then another digit.</td>
</tr>
<tr>
<td>\S</td>
<td>Match a non-whitespace character.</td>
<td>\d\S\d</td>
<td>Matches a sequence of a digit, a</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
<td>Example</td>
<td>Explanation</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>.</td>
<td>Match any character. Use sparingly.</td>
<td>\d\d.\d\d.\d\d</td>
<td>Matches a date string such as 12/31/14 or 01.01.15, but can also match 99A99B99.</td>
</tr>
</tbody>
</table>

**Groups, quantifiers, and alternation**

Regular expressions allow groupings indicated by the type of bracket used to enclose the regular expression characters. You can apply quantifiers ( *, +, ? ) to the enclosed group and use alternation within the group.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
<th>Example</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Match zero or more times.</td>
<td>\w*</td>
<td>Matches zero or more word characters.</td>
</tr>
<tr>
<td>+</td>
<td>Match one or more times.</td>
<td>\d+</td>
<td>Match at least one digit.</td>
</tr>
<tr>
<td>?</td>
<td>Match zero or one time.</td>
<td>\d\d\d-?\d\d-?\d\d\d</td>
<td>Matches a Social Security Number with or without dashes.</td>
</tr>
<tr>
<td>( )</td>
<td>Parentheses define match or capture groups, atomic groups, and lookarounds.</td>
<td>(H..)(o..)</td>
<td>When given the string Hello World, this matches Hel and o W.</td>
</tr>
<tr>
<td>[ ]</td>
<td>Square brackets define character classes.</td>
<td>[a-z0-9#]</td>
<td>Matches any character that is a through z, 0 through 9, or #.</td>
</tr>
</tbody>
</table>
Curly brackets define repetitions. \( \{ 3,5 \} \) Matches a string of 3 to 5 digits in length.

Angle brackets define named capture groups. Use the syntax \( (?P<var> ... ) \) to set up a named field extraction. \( (?P<ssn>\d\d\d-\d\d-\d\d\d\d) \) Pulls out a Social Security Number and assigns it to the \( ssn \) field.

Double brackets define Splunk-specific modular regular expressions. \( [[octet]] \) A validated 0-255 range integer.

**A simple example of groups, quantifiers, and alternation**

This example shows two ways to match either to or too.

The first regular expression uses the ? quantifier to match up to one more "o" after the first.

The second regular expression uses alternation to specify the pattern.

\( to(o)? \)  
\( (to|too) \)

**Capture groups in regular expressions**

A named capture group is a regular expression grouping that extracts a field value when regular expression matches an event. Capture groups include the name of the field. They are notated with angle brackets as follows:

matching text \( (?<field_name>capture \text{ pattern}) \) more matching text.

For example, you have this event text:

131.253.24.135 fail admin_user

Here are two regular expressions that use different syntax in their capturing groups to pull the same set of fields from that event.
- Expression A: (?<ip>\d+\.\d+\.\d+\.\d+) (?<result>\w+) (?<user>.*)
- Expression B: (?<ip>\S+) (?<result>\S+) (?<user>\S+)

In Expression A, the pattern-matching characters used for the first capture group (ip) are specific. \d means "digit" and + means "one or more." So \d+ means "one or more digits." \. refers to a period.

The capture group for ip wants to match one or more digits, followed by a period, followed by one or more digits, followed by a period, followed by one or more digits, followed by a period, followed by one or more digits. This describes the syntax for an ip address.

The second capture group in Expression A for the result field has the pattern \w+, which means "one or more alphanumeric characters." The third capture group in Expression A for the user field has the pattern .*, which means "match everything that's left."

Expression B uses a common technique called negative matching. With negative matching, the regular expression does not try to define which text to match. Instead it defines what the text is not. In this Expression B, the values that should be extracted from the sample event are "not space" characters (\S). It uses the + to specify "one or more" of the "not space" characters.

So Expression B says:

1. Pull out the first string of not-space characters for the ip field value.
2. Ignore the following space.
3. Then pull out the second string of not-space characters for the result field value.
4. Ignore the second space.
5. Pull out the third string of not-space characters for the user field value."

**Non-capturing group matching**

Use the syntax (?: ... ) to create groups that are matched but which are not captured. Note that here you do not need to include a field name in angle brackets. The colon character after the ? character is what identifies it as a non-capturing group.

For example, (?:Foo|Bar) matches either Foo or Bar, but neither string is captured.
Modular regular expressions

Modular regular expressions refer to small chunks of regular expressions that are defined to be used in longer regular expression definitions. Modular regular expressions are defined in transforms.conf.

For example, you can define an integer and then use that regular expression definition to define a float.

```
[int]
# matches an integer or a hex number
REGEX = 0x[a-fA-F0-9]+|\d+

[float]
# matches a float (or an int)
REGEX = \d+\.\d+|[\[\[int\]\]]
```

In the regular expression for [float], the modular regular expression for an integer or hex number match is invoked with double square brackets, `[[int]]`.

You can also use the modular regular expression in field extractions.

```
[octet]
# this would match only numbers from 0-255 (one octet in an ip)
REGEX = (?:2(?:5\[0-5]|[0-4][0-9])|[0-1][0-9][0-9]|\[0-9][0-9]?)

[ipv4]
# matches a valid IPv4 optionally followed by :port_num the 
# octets in the ip would also be validated 0-255 range 
# Extracts: ip, port
REGEX = (?<ip>\[[octet]\]|\:.\[[octet]\]\((3)\):\[[int;port]\])?
```

The [octet] regular expression uses two nested non-capturing groups to do its work. See the subsection in this topic on non-capturing group matching.
Fields and field extractions

About fields

Fields appear in event data as searchable name-value pairings such as user_name=fred or ip_address=192.168.1.1. Fields are the building blocks of Splunk searches, reports, and data models. When you run a search on your event data, Splunk software looks for fields in that data.

Look at the following example search.

status=404

This search finds events with status fields that have a value of 404. When you run this search, Splunk Enterprise does not look for events with any other status value. It also does not look for events containing other fields that share 404 as a value. As a result, this search returns a set of results that are more focused than you get if you used 404 in the search string.

Fields often appear in events as key=value pairs such as user_name=Fred. But in many events, field values appear in fixed, delimited positions without identifying keys. For example, you might have events where the user_name value always appears by itself after the timestamp and the user_id value.

Nov 15 09:32:22 00224 johnz
Nov 15 09:39:12 01671 dmehta
Nov 15 09:45:23 00043 sting
Nov 15 10:02:54 00676 lscott
Splunk Enterprise can identify these fields using a custom field extraction.

About field extraction

As Splunk software processes events, it extracts fields from them. This process is called field extraction.

Automatically-extracted fields

Splunk software automatically extracts host, source, and sourcetype values, timestamps, and several other default fields when it indexes incoming events.
It also extracts fields that appear in your event data as key=value pairs. This process of recognizing and extracting k/v pairs is called field discovery. You can disable field discovery to improve search performance.

When fields appear in events without their keys, Splunk software uses pattern-matching rules called regular expressions to extract those fields as complete k/v pairs. With a properly-configured regular expression, Splunk Enterprise can extract user_id=johnz from the previous sample event. Splunk Enterprise comes with several field extraction configurations that use regular expressions to identify and extract fields from event data.

For more information about field discovery and an example of automatic field extraction, see When Splunk Enterprise extracts fields.

For more information on how Splunk Enterprise uses regular expressions to extract fields, see About Splunk regular expressions.

**To get all of the fields in your data, create custom field extractions**

To use the power of Splunk search, create additional field extractions. Custom field extractions allow you to capture and track information that is important to your needs, but which is not automatically discovered and extracted by Splunk software. Any field extraction configuration you provide must include a regular expression that specifies how to find the field that you want to extract.

All field extractions, including custom field extractions, are tied to a specific source, sourcetype, or host value. For example, if you create an ip field extraction, you might tie the extraction configuration for ip to sourcetype=access_combined.

Custom field extractions should take place at search time, but in certain rare circumstances you can arrange for some custom field extractions to take place at index time. See When Splunk Enterprise extracts fields.

**Before you create custom field extractions, get to know your data**

Before you begin to create field extractions, ensure that you are familiar with the formats and patterns of the event data associated with the source, sourcetype, or host that you are working with. One way is to investigate the predominant event patterns in your data with the Patterns tab. See Identify event patterns with the Patterns tab in the Search Manual.
Here are two events from the same source type, an apache server web access log.

131.253.24.135 - - [03/Jun/2014:20:49:53 -0700] "GET /wp-content/themes/aurora/style.css HTTP/1.1" 200 7464 "http://www.splunk.com/download" "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.0; Trident/5.0; Trident/5.0)?


While these events contain different strings and characters, they are formatted in a consistent manner. They both present values for fields such as clientIP, status, bytes, method, and so on in a reliable order.

Reliable means that the method value is always followed by the URI value, the URI value is always followed by the status value, the status value is always followed by the bytes value, and so on. When your events have consistent and reliable formats, you can create a field extraction that accurately captures multiple field values from them.

For contrast, look at this set of Cisco ASA firewall log events:

<table>
<thead>
<tr>
<th>No</th>
<th>Date</th>
<th>Time</th>
<th>IP Address</th>
<th>Log Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Jul 15</td>
<td>20:10:27</td>
<td>10.11.36.31</td>
<td>%ASA-6-113003: AAA group policy for user AmorAubrey is being set to Acme_techoutbound</td>
</tr>
<tr>
<td>2</td>
<td>Jul 15</td>
<td>20:12:42</td>
<td>10.11.36.11</td>
<td>%ASA-7-710006: IGMP request discarded from 10.11.36.36 to outside:87.194.216.51</td>
</tr>
<tr>
<td>3</td>
<td>Jul 15</td>
<td>20:13:52</td>
<td>10.11.36.28</td>
<td>%ASA-6-302014: Teardown TCP connection 517934 for Outside:128.241.220.82/1561 to Inside:10.123.124.28/8443 duration 0:05:02 bytes 297 Tunnel has been torn down (AMOSORTILEGIO)</td>
</tr>
<tr>
<td>4</td>
<td>Apr 19</td>
<td>11:24:32</td>
<td>PROD-MFS-002</td>
<td>%ASA-4-106103: access-list fmVPN-1300 denied udp for user 'sdewilde7' outside/12.130.60.4(137) -&gt; insidel/10.157.200.154(137) hit-cnt 1 first hit [0x286364c7, 0x0]</td>
</tr>
</tbody>
</table>

While these events contain field values that are always space-delimited, they do not share a reliable format like the preceding two events. In order, these events represent:

1. A group policy change
2. An IGMP request
3. A TCP connection
4. A firewall access denial for a request from a specific IP

Because these events differ so widely, it is difficult to create a single field
extraction that can apply to each of these event patterns and extract relevant field values.

In situations like this, where a specific host, source type, or source contains multiple event patterns, you may want to define field extractions that match each pattern, rather than designing a single extraction that can apply to all of the patterns. Inspect the events to identify text that is common and reliable for each pattern.

**Using required text in field extractions**

In the last four events, the string of numbers that follows %ASA-#- have specific meanings. You can find their definitions in the Cisco documentation. When you have unique event identifiers like these in your data, specify them as required text in your field extraction. Required text strings limit the events that can match the regular expression in your field extraction.

Specifying required text is optional, but it offers multiple benefits. Because required text reduces the set of events that it scans, it improves field extraction efficiency and decreases the number of false-positive field extractions.

The field extractor utility enables you to highlight text in a sample event and specify that it is required text.

**Methods of custom field extraction**

As a knowledge manager you oversee the set of custom field extractions created by users of your Splunk deployment, and you might define specialized groups of custom field extractions yourself. The ways that you can do this include:

- The field extractor utility, which generates regular expressions for your field extractions.
- Adding field extractions through pages in Settings. You must provide a regular expression.
- Manual addition of field extraction configurations at the .conf file level. Provides the most flexibility for field extraction.

The field extraction methods that are available to Splunk users are described in the following sections. All of these methods enable you to create search-time field extractions. To create an index-time field extraction, choose the third option: Configure field extractions directly in configuration files.
Let the field extractor build extractions for you

The field extractor utility leads you step-by-step through the field extraction design process. It provides two methods of field extraction: regular expressions and delimiter-based field extraction. The regular expression method is useful for extracting fields from unstructured event data, where events may follow a variety of different event patterns. It is also helpful if you are unfamiliar with regular expression syntax and usage, because it generates regular expressions and lets you validate them.

The delimiter-based field extraction method is suited to structured event data. Structured event data comes from sources like SQL databases and CSV files, and produces events where all fields are separated by a common delimiter, such as commas, spaces, or pipe characters. Regular expressions usually are not necessary for structured data events from a common source.

With the regular expression method of the field extractor you can:

- Set up a field extraction by selecting a sample event and highlighting fields to extract from that event.
- Create individual extractions that capture multiple fields.
- Improve extraction accuracy by detecting and removing false positive matches.
- Validate extraction results by using search filters to ensure specific values are being extracted.
- Specify that fields only be extracted from events that have a specific string of required text.
- Review stats tables of the field values discovered by your extraction.
- Manually configure regular expression for the field expression yourself.

With the delimiter method of the field extractor you can:

- Identify a delimiter to extract all of the fields in an event.
- Rename specific fields as appropriate.
- Validate extraction results.

The field extractor can only build search time field extractions that are associated with specific sources or source types in your data (no hosts).

For more information about using the field extractor, see Build field extractions with the field extractor.
Define field extractions with the Field extractions and Field transformations pages

You can use the Field extractions and Field transformations pages in Settings to define and maintain complex extracted fields in Splunk Web.

This method of field extraction creation lets you create a wider range of field extractions than you can generate with the field extractor utility. It requires that you have the following knowledge.

• Understand how to design regular expressions.
• Have a basic understanding of how field extractions are configured in props.conf and transforms.conf.

If you create a custom field extraction that extracts its fields from _raw and does not require a field transform, use the field extractor utility. The field extractor can generate regular expressions, and it can give you feedback about the accuracy of your field extractions as you define them.

Use the Field Extractions page to create basic field extractions, or use it in conjunction with the Field Transformations page to define field extraction configurations that can do the following things.

• Reuse the same regular expression across multiple sources, source types, or hosts.
• Apply multiple regular expressions to the same source, source type, or host.
• Use a regular expression to extract fields from the values of another field.

The Field extractions and Field transformations pages define only search time field extractions.

See the following topics in this manual:

• Use the Field extractions page in Splunk Web
• Use the Field transformations page in Splunk Web.

Configure field extractions directly in configuration files

To get complete control over your field extractions, add the configurations directly into props.conf and transforms.conf. This method lets you create field extractions with capabilities that extend beyond what you can create with Splunk Web methods such as the field extractor utility or the Settings pages. For
example, with the configuration files, you can set up:

- Delimiter-based field extractions.
- Extractions for multivalue fields.
- Extractions of fields with names that begin with numbers or underscores. This action is typically not allowed unless key cleaning is disabled.
- Formatting of extracted fields.

See Create and maintain search-time extractions through configuration files.

You can create **index-time** field extractions only by configuring them in props.conf and transforms.conf. Adding to the default set of indexed fields can result in search performance and indexing problems. But if you must create additional index-time field extractions, see Create custom fields at index time in the Getting Data In manual.

Create custom calculated fields and multivalue fields

Two kinds of custom fields can be persistently configured with the help of .conf files: calculated fields and multivalue fields.

**Multivalue fields** can appear multiple times in a single event, each time with a different value. To configure custom multivalue fields, make changes to fields.conf as well as to props.conf. See Configure multivalue fields.

**Calculated fields** provide values that are calculated from the values of other fields present in the event, with the help of eval expressions. Configure them in props.conf. See About calculated fields.

**Build field extractions into search strings**

The following search commands facilitate the search-time extraction of fields in different ways:

- rex
- extract
- multikv
- spath
- xmlkv
- xpath
- kvform

See Extract fields with search commands in the Search Manual. Alternatively you can look up each of these commands in the Search Reference.
Field extractions facilitated by search commands apply only to the results returned by the searches in which you use these commands. You cannot use these search commands to create reusable extractions that persist after the search is completed. For that, use the field extractor utility, configure extractions with the Settings pages, or set up configurations directly in the .conf files.

Use default fields

Fields are searchable name-value pairs in event data. When you search, you’re matching search terms against segments of your event data; you can search more precisely by using fields. Fields are extracted from event data at either index time or search time. The fields that are extracted automatically at index time are known as default fields.

Default fields serve a number of purposes. For example, the default field index identifies the index in which the event is located. The default field linecount describes the number of lines the event contains, and timestamp specifies the time at which the event occurred. Splunk software uses the values in some of the fields, particularly sourcetype, when indexing the data, in order to create events properly. After the data has been indexed, you can use the default fields in your searches.

For more information on using default fields in search commands, see About the search language in the Search Manual. For information on configuring default fields, see About default fields in the Getting Data In manual.

<table>
<thead>
<tr>
<th>Type of field</th>
<th>List of fields</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal fields</td>
<td>_raw, _time, _indextime, _cd, _bkt</td>
<td>Contain general information about events.</td>
</tr>
<tr>
<td>Default fields</td>
<td>host, index, linecount, punct, source, sourcetype, splunk_server, timestamp</td>
<td>These are fields that contain information about where an event originated, in which index it's located, what type it is, how many lines it contains, and when it occurred. These fields are indexed and added to the Fields menu by default.</td>
</tr>
<tr>
<td>Default datetime fields</td>
<td>date_hour, date_mday, date_minute, date_month</td>
<td>These are fields that provide additional searchable granularity to event timestamps.</td>
</tr>
</tbody>
</table>
Note: Only events that have timestamp information in them as generated by their respective systems will have date_* fields. If an event has a date_* field, it represents the value of time/date directly from the event itself. If you have specified any timezone conversions or changed the value of the time/date at indexing or input time (for example, by setting the timestamp to be the time at index or input time), these fields will not represent that.

A field can have more than one value. See Manipulate and evaluate fields with multiple values.

You can extract non-default fields with Splunk Web or by using extracting search commands. See About fields.

You might also want to change the name of a field, or group it with other similar fields. This is easily done with tags or aliases for the fields and field values. See Tag field value pairs in Search.

This topic discusses the internal and other default fields that Splunk software automatically adds when you index data.

**Internal fields**

Fields that begin with an underscore are internal fields.

Do not override internal fields unless you are absolutely sure you know what you are doing.

__raw__

The __raw__ field contains the original raw data of an event. The search command uses the data in __raw__ when performing searches and data extraction.

You cannot always search directly on values of __raw__, but you can filter on __raw__ with commands like regex or sort.

**Example:** Return sendmail events that contain an IP address that starts with 10.

```
eventtype=sendmail | regex _raw=*_10.\d\d\d\d.\d\d\d\d/\d\d\d\d*
```

59
_time

The _time field contains an event's timestamp expressed in Unix time. This field is used to create the event timeline in Splunk Web.

**Note:** The _time field is stored internally in UTC format. It is translated to human-readable Unix time format when Splunk software renders the search results (the very last step of search time event processing).

**Example:** Search all sources of type mail for mail addressed to the user strawsky@bigcompany.com. Then sort the search results by timestamp.

```
sourcetype=mail to=strawsky@bigcompany.com | sort _time
```

_indextime

The _indextime field contains the time that an event was indexed, expressed in Unix time. You might use this field to focus on or filter out events that were indexed within a specific range of time. Because _indextime is a hidden field, it will not be displayed in search results unless renamed or used with an eval.

_cd

The _cd field provides an address for an event within the index. It is composed of two numbers, a short number and a long number. The short number indicates the specific index bucket that the event resides in. The long number is an index bucket offset. It provides the exact location of the event within its bucket. Because _cd is a hidden field, it will not be displayed in search results unless renamed or used with an eval. Because _cd is used for internal reference only, we do not recommend that you set up searches that involve it.

_bkt

The _bkt field contains the id of the bucket that an event is stored in. Because _bkt is a hidden field, it will not be displayed in search results unless renamed or used with an eval.

Other default fields

host

The host field contains the originating hostname or IP address of the network device that generated the event. Use the host field to narrow searches by
specifying a host value that events must match. You can use wildcards to specify multiple hosts with a single expression (Example: host=corp*).

You can use host to filter results in data-generating commands, or as an argument in data-processing commands.

**Example 1:** Search for events on all corp servers for accesses by the user strawsky. It then reports the 20 most recent events.

```
host=corp* eventtype=access user=strawsky | head 20
```

**Example 2:** Search for events containing the term 404, and are from any host that starts with 192.

```
404 | regex host=*192.\d\d\d.\d\d\d.\d\d\d\d\d\d\d\d\d\d\d\d\d\d\d\d\d
```

**index**

The index field contains the name of the index in which a given event is indexed. Specify an index to use in your searches by using: index="name_of_index". By default, all events are indexed in the main index.

**Example:** Search the myweb index for events that have the .php extension.

```
index="myweb" * .php
```

**linecount**

The linecount field contains the number of lines an event contains. This is the number of lines an event contains before it is indexed. Use linecount to search for events that match a certain number of lines, or as an argument in data-processing commands. To specify a matching range, use a greater-than and less-than expression (Example: linecount>10 linecount<20).

**Example:** Search corp1 for events that contain 40 and have 40 lines, and omit events that contain 400.

```
40 linecount=40 host=corp1 NOT 400
```

**punct**

The punct field contains a punctuation pattern that is extracted from an event. The punctuation pattern is unique to types of events. Use punct to filter events during a search or as a field argument in data-processing commands.
You can use wildcards in the punct field to search for multiple punctuation patterns that share some common characters that you know you want to search for. You must use quotation marks when defining a punctuation pattern in the punct field.

**Example 1:** Search for all punctuation patterns that start and end with :

```
punct=":*:"
```

**Example 2:** Search the php_error.log for php error events that have the punctuation pattern `[--_:]__:___:____/-.//.///"`.

```
source="/var/www/log/php_error.log"
punct="[--_:]__:___:____"/-.//.///"
```

**source**

The source field contains the name of the file, stream, or other input from which the event originates. Use source to filter events during a search, or as an argument in a data-processing command. You can use wildcards to specify multiple sources with a single expression (Example: `source=*php.log*`).

You can use source to filter results in data-generating commands, or as an argument in data-processing commands.

**Example:** Search for events from the source /var/www/log/php_error.log.

```
source="/var/www/log/php_error.log"
```

**sourcetype**

The sourcetype field specifies the format of the data input from which the event originates, such as access_combined or cisco_syslog. Use sourcetype to filter events during a search, or as an argument in a data-processing command. You can use wildcards to specify multiple sources with a single expression (Example: `sourcetype=access*`).

**Example:** Search for all events that are of the source type access log.

```
sourcetype=access_log
```
**splunk_server**

The `splunk_server` field contains the name of the Splunk server containing the event. Useful in a distributed Splunk environment.

**Example:** Restrict a search to the main index on a server named `remote`.

```
splunk_server=remote index=main 404
```

**timestamp**

The `timestamp` field contains an event's timestamp value. You can configure the method that is used to extract timestamps. You can use `timestamp` as a search command argument to filter your search.

For example, you can add `timestamp=none` to your search to filter your search results to include only events that have no recognizable timestamp value.

**Example:** Return the number of events in your data that have no recognizable timestamp.

```
timestamp=none | stats count(_raw) as count
```

**Default datetime fields**

You can use datetime fields to filter events during a search or as a field argument in data-processing commands.

If you are located in a different timezone from the Splunk server, time-based searches use the timestamp of the event as specified on the server where the event was indexed. The datetime values are the literal values parsed from the event when it is indexed, regardless of its timezone. So, a string such as `05:22:21` will be parsed into indexed fields: `date_hour::5` `date_minute::22` `date_second::21`.

**date_hour**

The `date_hour` field contains the value of the hour in which an event occurred (range: 0-23). This value is extracted from the event's timestamp (the value in `_time`).

**Example:** Search for events with the string `apache` that occurred between 10pm and 12am on the current day.
apache (date_hour >= 22 AND date_hour <= 24)

**date_mday**

The *date_mday* field contains the value of the day of the month on which an event occurred (range: 1-31). This value is extracted from the event's timestamp (the value in `_time`).

**Example:** Search for events containing the string `apache` that occurred between the 1st and 15th day of the current month.

apache (date_mday >= 1 AND date_mday <= 15)

**date_minute**

The *date_minute* field contains the value of the minute in which an event occurred (range: 0-59). This value is extracted from the event's timestamp (the value in `_time`).

**Example:** Search for events containing the string `apache` that occurred between the 15th and 20th minute of the current hour.

apache (date_minute >= 15 AND date_minute <= 20)

**date_month**

The *date_month* field contains the value of the month in which an event occurred. This value is extracted from the event's timestamp (the value in `_time`).

**Example:** Search for events with the string `apache` that occurred in January.

apache date_month=1

**date_second**

The *date_second* field contains the value of the seconds portion of an event's timestamp (range: 0-59). This value is extracted from the event's timestamp (the value in `_time`).

**Example:** Search for events containing the string `apache` that occurred between the 1st and 15th second of the current minute.

apache (date_second >= 1 AND date_second <= 15)
date_wday

The `date_wday` field contains the day of the week on which an event occurred (Sunday, Monday, etc.). The date is extracted from the event's timestamp (the value in `_time`) and determines what day of the week that date translates to. This day of the week value is then placed in the `date_wday` field.

**Example:** Search for events containing the string `apache` that occurred on Sunday.

```
apache date_wday="sunday"
```

date_year

The `date_year` field contains the value of the year in which an event occurred. This value is extracted from the event's timestamp (the value in `_time`).

**Example:** Search for events containing the string `apache` that occurred in 2008.

```
apache date_year=2008
```

date_zone

The `date_zone` field contains the value of time for the local timezone of an event, expressed as hours in Unix Time. This value is extracted from the event's timestamp (the value in `_time`). Use `date_zone` to offset an event's timezone by specifying an offset in minutes (range: -720 to 720).

**Example:** Search for events containing the string `apache` that occurred in the current timezone (local).

```
apache date_zone=local
```

When Splunk software extracts fields

Fields are extracted at *index time* and again at *search time*. After you run a search, fields extracted for that search are listed in the fields sidebar.

Field extraction at index time

At index time, Splunk software extracts a small set of default fields for each event, including `host`, `source`, and `sourcetype`. Default fields are common to all
events. See Use default fields.

Splunk software can also extract custom **indexed fields** at index time. These are fields that you have explicitly configured for index-time extraction.

**Caution:** Do not add custom fields to the set of default fields that Splunk software extracts and indexes at **index time**. Adding to this list of fields can slow indexing performance and search times, because each indexed field increases the size of the searchable index. Indexed fields are also less flexible, because whenever you make changes to your set of indexed fields, you must re-index your entire dataset. See Index time versus search time in the *Managing Indexers and Clusters* manual.

**Field extraction at search time**

At search time, Splunk software can extract additional fields, depending on its **Search Mode** setting and whether that setting enables **field discovery** given the type of search being run.

When field discovery is enabled, Splunk software:

- Identifies and extracts the first 50 fields that it finds in the event data that match obvious `key=value` pairs. This 50 field limit is a default that you can modify by editing the `[kv]` stanza in `limits.conf`, if you have Splunk Enterprise.
- Extracts any field explicitly mentioned in the search that it might otherwise have found though automatic extraction, but is not among the first 50 fields identified.
- Performs custom field extractions that you have defined, either through the Field Extractor, the Extracted Fields page in Settings, configuration file edits, or search commands such as `rex`.

When field discovery is disabled, Splunk software extracts:

- Any field explicitly mentioned in the search.
- The default and indexed fields mentioned above.
- Any custom field extraction that has the `CAN_OPTIMIZE` parameter set to `true` in `transforms.conf`.

Splunk software discovers fields other than default fields and fields explicitly mentioned in the search string only when you:

- Run a **non-transforming** search in the *Smart* search mode.
• Run any search in the *Verbose* search mode.

See Set search mode to adjust your search experience in the *Search Manual.*

For an explanation of search time and index time, see Index time versus search time in the *Managing Indexers and Clusters* manual.

**Example of automatic field extraction**

This is an example of how Splunk software automatically extracts fields without user help, as opposed to custom field extractions, which follow event-extraction rules that you define.

Say you search on `sourcetype`, a default field that Splunk software extracts for every event at index time. If your search is

```
sourcetype=veeblefetzer
```

for the past 24 hours, Splunk software returns every event with a sourcetype of `veeblefetzer` in that time range. From this set of events, Splunk software extracts the first 50 fields that it can identify on its own. And it performs extractions of custom fields, based on configuration files. All of these fields appear in the fields sidebar when the search is complete.

Now, if a name/value combination like `userlogin=fail` appears for the first time 25,000 events into the search, and `userlogin` isn't among the set of custom fields that you've preconfigured, it likely is not among the first 50 fields that Splunk software finds on its own.

However, if you change your search to

```
sourcetype=veeblefetzer userlogin=*  
```

then Splunk software finds and returns all events including both the `userlogin` field and a `sourcetype` value of `veeblefetzer`. It will be available in the field sidebar along with the other fields extracted for this search.

**About regular expressions with field extractions**

Inline and transform *field extractions* require regular expressions with the names of the fields that they extract.
In **inline field extractions**, the regular expression is in `props.conf`. You have one regular expression per field extraction configuration.

In **transform extractions**, the regular expression is separated from the field extraction configuration. The regular expression is in `transforms.conf` while the field extraction is in `props.conf`. This means that you can apply one regular expression to multiple field extraction configurations, or multiple regular expressions to one field extraction configuration.

**Regular expressions**

When you set up field extractions through configuration files, you must provide the regular expression. You can design them so that they extract two or more fields from the events that match them. You can test your regular expression by using the `rex` search command.

The capturing groups in your regular expression must identify field names that contain alpha-numeric characters or an underscore.

You can use the **field extractor** to generate field-extracting regular expressions. For information on the field extractor, see Build field extractions with the field extractor.

**Proper field name syntax**

Field names must conform to the field name syntax rules.

- Valid characters for field names are `a-z`, `A-Z`, `0-9`, `.`, `;`, and `_`.
- Field names cannot begin with `0-9` or `_`. Leading underscores are reserved for Splunk Enterprise internal variables.

Splunk software applies key cleaning to fields that are extracted at search time. When key cleaning is enabled, Splunk Enterprise removes all leading underscores and `0-9` characters from extracted fields. Key cleaning is enabled by default.

You can disable key cleaning for a search-time field extraction by configuring it as an advanced **REPORT-** extraction type, including the setting `CLEAN_KEYS=false` in the referenced field transform stanza. See Create advanced search-time field extractions with field transforms.

You cannot turn off key cleaning for inline **EXTRACT-** (`props.conf` only) field extraction configurations. See Configure inline extractions with `props.conf`. 
Use the field extractor in Splunk Web

Build field extractions with the field extractor

Use the **field extractor** utility to create new fields. The field extractor provides two field extraction methods: regular expression and delimiters.

The regular expression method works best with unstructured event data. You select a sample event and highlight one or more fields to extract from that event, and the field extractor generates a regular expression that matches similar events in your dataset and extracts the fields from them. The regular expression method provides several tools for testing and refining the accuracy of the regular expression. It also allows you to manually edit the regular expression.

The delimiters method is designed for structured event data: data from files with headers, where all of the fields in the events are separated by a common delimiter, such as a comma or space. You select a sample event, identify the delimiter, and then rename the fields that the field extractor finds. data that resides in a file that has headers and fields separated by specific characters

Overview of the field extractor

To help you create a new field, the field extractor takes you through a set of steps. The field extractor workflow diverges at the Select Method step, where you select the field extraction method that you want to use.

This table gives you an overview of the required steps. For detailed information about a step, click the link in the **Step Title** column.

<table>
<thead>
<tr>
<th>Step Title</th>
<th>Description</th>
<th>Field Extraction</th>
</tr>
</thead>
</table>

69
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select sample</td>
<td>Select the <strong>source type</strong> or <strong>source</strong> that is tied to the events that have the field (or fields) that you want to extract. Then choose a sample event that has that field (or fields).</td>
</tr>
<tr>
<td>Select method</td>
<td>Select a field extraction method. You can have the field extractor generate a field-extracting regular expression, or you can employ delimiter-based field extraction. The choice you make depends on whether you are trying to extract fields from unstructured or structured event data.</td>
</tr>
</tbody>
</table>
| Select fields | Highlight one or more field values in the event to identify them as fields. The field extractor generates a regular expression that matches the event and extracts the field. Optionally, you can:  
- Provide additional sample events to improve extraction accuracy.  
- Identify required text to focus the field extraction on events that contain this text.  
- Examine field extraction results.  
- Update the underlying regular expression manually. |
| Rename fields | Identify the delimiter that separates all of the fields in the event, and then rename one or more of those fields. |
| Validate fields |  
- Examine the field extraction results.  
- Identify incorrectly extracted fields as counterexamples to improve the accuracy of the field extraction. |
| Save | Name your new field extraction, set its permissions, and save it. |

### Access the field extractor

There are several ways to access the field extractor utility. The access method you use can determine which step of the field extractor workflow you start at.

All users can access the field extractor after running a search that returns events. You have three post-search entry points to the field extractor:
• Bottom of the fields sidebar
• All Fields dialog box
• Any event in the search results

You can also enter the field extractor:

• from the Field Extractions page in Settings.
• when you add data with a fixed source type.
• from the Splunk Web Home page (if you have Admin role privileges).

**Access the field extractor from the bottom of the fields sidebar**

When you use this method to access the field extractor it runs only against the set of events returned by the search that you have run. To get the full set of source types in your Splunk deployment, go to the Field Extractions page in Settings.

1. Run a search that returns events.
2. Scroll down to the bottom of the fields sidebar and click **Extract New Fields**.
   The field extractor starts you at the Select Sample step.

**Access the field extractor from the All Fields dialog box**

When you use this method to access the field extractor you can only extract fields from the data that has been returned by your search. To get the full set of source types in your Splunk deployment, go to the Field Extractions page in Settings.

1. Run a search that returns events.
2. At the top of the fields sidebar, click **All Fields**.
3. In the All Fields dialog box, click **Extract new fields**.
   The field extractor starts you at the Select Sample step.
Access the field extractor from a specific event

Use this method to select an event in your search results, and create a field extraction that:

- Extracts one or more fields found in that event.
- Is tied to the source type of that event.

When you use this method to access the field extractor, the field extractor runs against the set of events returned by the search that you have run.

1. Run a search that returns events.
2. Find an event that you want to extract fields from, and click the arrow symbol to the left of the timestamp to open it.
3. Click Event Actions, and select Extract Fields.
   The field extractor starts you at the Select Method step, in a new browser tab. You have already defined the source type and sample event.

Access the field extractor through the Field Extractions page in Settings

This entry method is available to all users.

1. Select Settings > Fields > Field extractions.
2. Click the Open field extractor button.
   The field extractor starts you at the Select Sample step.

Access the field extractor through the Home page

This entry method is available only to users whose roles have the edit_monitor capability, such as Admin.

On the Home page, click the extract fields link under the Add Data icon.

   The field extractor starts you at the Select Sample step.
Access the field extractor after you add data

This entry method is available only to users whose roles have the edit_monitor capability, such as Admin.

After you add data to Splunk Enterprise, use the field extractor to extract fields from that data, as long as it has a fixed source type.

For example: You add a file named vendors.csv to your Splunk deployment and give it the custom source type vendors. After you save this input, you can enter the field extractor and extract fields from the events associated with the vendors source type.

Another example: You create a monitor input for the /var/log directory and select Automatic for the source type, meaning that Splunk software automatically determines the source type values of the data from that input on an event by event basis. When you save this input you do not get a prompt to extract fields from this new data input, because the events indexed from that directory can have a variety of source type values.

1. Enter the Add Data page. See "How do you want to add data?" in the Getting Data In manual.
2. Define a data input with a fixed source type. This can be an existing source type or a custom source type that you define. See "View and set source types for event data" in the Getting Data In manual.
3. Save the new data input. Note: Wait 30 seconds before going to the next step. This gives the Splunk software some time to index the data and get it ready for field extraction.
4. In the "File has been uploaded successfully" dialog box, click Extract Fields.

The field extractor starts you at the Select Sample step.

Field Extractor: Select Sample step

In the Select Sample field extractor step, you do two things:

- First you identify a data type for your field extraction. Your data type selection brings up a list of events that have the selected source or source type value.
• Then you select an event from the list that has the field or fields that you want to extract.

The field extractor bypasses the Select Sample step when you enter the field extractor from a specific event in your search results. When you do this, the field extractor starts you off at the Select Method step.

**Select a data type and a sample event**

**Note:** The field extractor bypasses the first step of this procedure (select a data type) if you choose your source type before you enter the field extractor.

The Splunk field extractor is limited to twenty lines on a sample event.

This happens when you enter the field extractor:

• After you run a search where a specific source type is identified in the search string and then click the **Extract New Fields** link in the fields sidebar or the All Fields dialog box.
• After you run a search that returns a set of events that all have the same source type, and then click the **Extract New Fields** link in the fields sidebar or All Fields dialog box.
• After you add a data input with a fixed source type.

**Steps**

1. Select a **Data Type** for your field extraction.

   Each field extraction is associated with a specific **source type** or **source** value. If you have entered the field extractor after running a search, the sets of sources and source types that you can choose from are limited to those discovered in the results returned by that search. To see all of the source and source type sets in your Splunk deployment, go to the Field Extractions page in Settings.

   If you select **sourcetype** the **Source Type** list appears. Choose a source type there. If you do not see the source type that you would like to use, try specifying the source type that you want to use in that search and rerunning it.

   If you select **source** the **Source Name** field appears. Enter a source value there.

   This screenshot is an example of the source type listing you see when you enter the field extractor from the Field Extractions page in Settings.
If you run a search and then enter the field extractor by clicking Extract New Fields at the bottom of the fields sidebar, your Source Type list options may be reduced. This is because the list only shows source types that appear in the data returned by the search.

After you provide a source type or source, the Events tab appears. If events exist that have the source or source type that you provided, they are listed in this tab.

2. In the event list, select a sample event that has one or more values that you want to extract as fields. Sample events are limited to twenty lines. The selected event appears just above the Events tab.

When field extractions already exist for the source type or source that you have chosen, they are surrounded by colored outlines in the selected event and the events in the event list. Mouse over a
circled value to see the name of the field.

**Note:** When two or more field extractions overlap in the event that you select, only one of them is highlighted. A red triangle warning icon appears next to the **Existing fields** button when the field extractor detects overlapping fields. See "Use the Fields sidebar to control existing field extraction highlighting"

3. Click **Next** to go to the Select Method step.

**Use the Fields sidebar to control existing field extraction highlighting**

This is an optional action that you can perform on every field extractor step except Save.

The source or source type that you select may already be associated with search-time field extractions. When this is the case, the field extractor highlights the extracted field values in the sample events with colored outlines.

The field extractor highlighting functionality cannot display highlighting for overlapping field values. When two or more extracted fields share event text, it can only display highlighting for one of those fields at a time.

For example, if the field extractor extracts a **phone_number** value of *(555) 789-1234* and an **area_code** value of *555* from the same bit of text in an event, it can display highlighting for the **phone_number** value or the **area_code** value, but not both at once.

When two or more existing field extractions overlap, the field extractor automatically disables highlighting for all of the fields. If you select a sample event with overlapping field extractions, the field extractor displays a red triangle warning indicator next to the **Existing fields** button.

**Note:** This warning does not appear when you use the Field sidebar to manually turn off highlighting for extracted fields that do not overlap with other fields.

The **Existing fields** button opens the Fields sidebar. Use the Fields sidebar to:

- Determine which existing field extractions are highlighted in the sample events.
• Turn off highlighting for an existing field extraction, if you want to define a new field extraction that overlaps with it.
• Determine whether an existing field extraction is accurately extracting field values.

Steps

1. Click **Existing fields** in the upper right of the screen.
   The **Fields** sidebar opens. Existing field extractions for your selected source or source type appear in a table.

   It is possible for a field to appear multiple times with different **Pattern Name** values.
   If there are no existing field extractions, the table does not appear.

2. (Optional) Click **open** for an extraction to see detail information about it.
   A page opens in a new tab. This page displays the regular expression that extracts the field. It also provides examples of events that the field extraction matches and values that the regular expression extracts.
   If the field extraction matches a different **event pattern** than the one you want to extract the field from, you can create a new extraction with the same name as long as it has a unique **Pattern Name**. You define the pattern name for your field extraction at the Save step.

3. (Optional) Use the **Highlighted** checkboxes to manage highlighting of extracted fields in sample events.
   Uncheck a **Highlighted** checkbox to turn off highlighting for a field and vice versa.
   When two or more field extractions overlap with each other, only one of the field extractions can have highlighting enabled at any given time. To make an unavailable field extraction available again, deselect the field extraction that overlaps with it. If you then select the other extraction, the extraction that you just deselected becomes unavailable.
If you want to create a new field extraction that overlaps with an existing field extraction, you must first deselect the existing extraction. See the documentation of the Select Fields step for more information.

4. Close the sidebar by clicking the X in the corner or by clicking outside of the sidebar.

**Field Extractor: Select Method step**

In the Select Method step of the field extractor you can choose a field extraction method that fits the data you are working with.

The step displays your **Source** or **Source type** and your sample event. At the bottom of the step you see two field extraction methods: **Regular expression** and **Delimiter**.

**Steps**

1. Click the field extraction method that is appropriate for your data.
   - Click **Regular Expression** if the event that you have selected is derived from unstructured data such as a system log. The field extractor can attempt to generate a regular expression that matches similar events and extracts your fields.
   - Click **Delimiters** if the fields in your selected event are:
     - cleanly separated by a common delimiter, such as a space, a comma, or a pipe character.
     - consistent across multiple events (each value is in the same place from event to event).
This is commonly the case with structured, table-based data such as `.csv` files or events indexed from a database. Here is an example of an event that uses a comma delimiter to separate out its fields. Its source is a `.csv` file from the USGS Earthquakes website which provides data on earthquakes that have occurred around the world over a 30 day period.

```
```

You can see that there is a missing field where two commas appear next to each other.

In cases where your fields are separated by delimiters but are not consistent across multiple events, you should use the **Regular Expression** method in conjunction with required text. Here’s an example of two events that use a cleanly separated comma delimiter but whose fields are not consistent:

- `indexer.splunk.com,jesse,pwcheck.fail`
- `Indexer.splunk.com,usercheck,greg`

The second field extraction would include `jesse` and `usercheck`, even through those are values for two different fields. So this set of events is not a good candidate for delimiter-based field extraction.

2. Click **Next** to go on to the next step. If you have chosen the **Regular Expression** method, you go on to the Select fields step. If you have chosen the **Delimiters** method, you go on to the Rename fields step.

### Field Extractor: Select Fields step

The Select Fields step of the field extractor is for regular-expression-based field extractions only.

In the Select Fields step of the field extractor, highlight values in the sample event that you want the field extractor to extract as fields.

To improve the accuracy of your field extraction, you can optionally:

- Preview the results returned by the regular expression.
- Identify additional sample events to expand the range of the regular expression.
- Identify a string of required text to focus the field extraction on events that contain this text.
- Manually edit the regular expression.
Identify one or more field values

Define at least one field extraction for your chosen source or source type.

1. In the sample event, highlight a value that you want to extract as a field.
   A dialog box with fields appears underneath the highlighted value.
   **Note:** The field extractor identifies existing field extractions in the
   sample event with colored outlines. If the text that you want to
   select overlaps with an existing field extraction, you must turn off its
   highlighting before you can select the overlapping text. You can
   turn off highlighting for a previously-extracted field using the
   Existing Fields sidebar. See "Use the Fields sidebar to control
   existing field extraction highlighting" in the Select Sample step.

2. Enter a name for the **Field Name** field.
   Field names must start with a letter and contain only letters,
   numbers, and underscores.

3. Click **Add Extraction** to save the extraction.
   When you add your first field extraction, the field extractor
   generates a regular expression that matches events like the event
   that you have selected and attempts to extract the field that you
   have defined from those events.
   The field extractor also displays a Preview section under the
   sample event. This section displays the list of events that match
   your chosen source or source type, and indicates which of those
   events match the regular expression that the field extractor has
   generated. The field extractor identifies the extracted field with
   colored highlighting. Previously extracted fields for the selected
   source or source type are indicated by a colored outline.

4. (Optional) Preview the results of the field extraction to see whether or not
   the field is being extracted correctly.
   This can help you determine whether you need to take steps to
   improve your field extraction by adding sample events or identifying
   required text.
   See "Preview the results of the field extraction".

5. (Optional) Repeat steps 1 through 4 until you identify all the values that
   you want to extract.
   The field extractor gives each extracted value a different highlight
   color.
   As you select more fields in an event for extraction there is a
   greater chance that the field extractor will be unable to generate a
   regular expression that can reliably extract all of the fields. You can
   improve the reliability of multifield extractions by adding sample
   events and identifying required text. You can also improve the
regular expression by editing it manually.

6. (Optional) Remove or rename field extractions in the sample event by clicking on them and selecting an action of Remove or Rename.

7. Click Next to go to the Validate Fields step.

**Preview the results of the field extraction**

This action is optional for the Select Fields and Validate Fields steps.

The Preview section appears after you add your first field extraction. It displays a list of the events that match your chosen source or source type. It also displays tabs for each field that you are trying to extract from the sample event.

The event list has features that you can use to inspect the accuracy of the field extraction. The list displays all of the events in the sample for the source type, by default.

- Use the left-most column to identify which events match the regular expression and which events do not.
- If the regular expression matches a small percentage of the sample events, toggle the view to Matches to remove the nonmatching events from the list. You can also select Non-Matches to see only the events that fail to match the regular expression.
- Click a field tab to value distribution statistics for a field. Each field tab displays a bar chart showing the count of each value found for the field in the event sample, organized from highest to lowest.

• Click a value in the chart to filter the field listing table on that value. For example, in the status chart, a click on the 503 value causes the field extractor to return to the main Preview field list view, with the filter set to status=503. It only lists events with that status value.
You may find that the generated field extraction is not correctly matching events. Or you may discover that it is extracting the wrong field values. When this happens, there are steps that you can take to improve the field extraction.

You can:

- Add sample events to extend the range of the regular expression. This can help it to match more events.
- Identify required text to create extractions that match specific event patterns. This reduces the set of events that are matched by the regular expression.
- Submit incorrectly extracted field values as counterexamples in the Validate Fields step.
- Remove fields from an extraction that involves multiple fields, when the extraction fails. You can create additional field extractions for those removed fields.

**Add sample events to expand the range of the regular expression**

This action is optional for the Select Fields step.

When you select a set of fields in your sample event you may find that events with those fields are not matched. This happens when the regular expression generated by the field extractor matches events with patterns similar to your sample event, but misses others that have slightly different patterns.

Try to expand the range of the regular expression by adding one of the missed events as an additional sample event. After you highlight the missed fields, the field extractor attempts to generate a new field extraction that encompasses both event patterns.

1. In the field listing table, click an event that is not matched by the regular expression but which has values for all of the fields that you are extracting from your first sample event.
   Additional sample events have the greatest chance of improving the accuracy of the field extraction when their format or pattern closely matches that of the original sample event.
   The sample event you select appears under the original sample event.
2. In the additional sample event, highlight the value for a field that you are extracting from the first sample event.
3. Select the correct **Field Name**.
You see names only for fields that you identified in the first sample event.

4. Click **Add Extraction**.
   - The field extractor attempts to expand the range of the regular expression so that it can find the field value in both event patterns. It matches the new regular expression against the event sample and displays the results in the event table.

5. (Optional) If you are extracting multiple fields, repeat steps 2 through 4 for each field.
   - You do not need to highlight all of the fields that are highlighted in the first sample event. For example, you may find that a more reliable field extraction results when the additional sample event only highlights one of the two fields highlighted in the original sample event.

6. (Optional) Add additional sample events.

7. (Optional) Remove sample events by clicking the gray "X" next to the event.

The field extractor sometimes cannot build a regular expression that matches the sample events as well as the original sample event. You can address the situation by using one of these methods:

- **Remove some of the fields you are trying to extract, if you are extracting multiple fields.** This action can result in a field extraction that works across all of your selected events. The first field values you should remove are those that are embedded within longer text strings. You can set up separate field extractions for the fields that you remove.

- **Define a separate field extraction for each event pattern that contains the field values that you want to extract, using required text to set the extractions apart.** For information about required text, see the next topic.
Identify required text to create extractions that match specific event patterns

This action is optional for the Select Fields step.

Sometimes a source type contains different kinds of events that contain the same field or fields that you want to extract. It can be difficult to design a single field extraction that matches multiple event patterns. One way to deal with this is to define a different field extraction for each event pattern.

You can focus the extraction to specific event patterns with required text. Required text behaves like a search filter. It is a string of text that must be present in the event for Splunk software to match it with the extraction.

For example, you might have event patterns for the access_combined source type that are differentiated by the strings action=addtocart, action=changequantity, action=purchase, and action=remove. You can create four extractions, one for each string, that each extract the same fields, but which have a different string for required text.

You can also use required text to make sure that a value is extracted only from specific events.

There are two limits to required text definition:

- You can define only one string of required text for a single field extraction.
- You cannot apply a required text string to a string of text that you highlighted as an extracted field value, nor can you do the reverse.

**Procedure**

1. In the sample event, highlight the text you want to require.
2. Select **Require**.
3. Click **Add Required Text** to add the required text to the field extraction.
4. (Optional) Remove required text in the sample event by clicking it and selecting **Remove Required Text**.
This example shows a field extraction that extracts fields named \texttt{http-method}\textcolor{green}{} (green) and \texttt{status}\textcolor{yellow}{} (yellow) and which has \texttt{action=purchase} defined as required text. In the field listing table, the first two events do not match the extraction, because they do not have the required text. The third event matches the regular expression and has the required text. It has highlighting that shows the extracted fields.

The filter feature is a useful tool for setting up and testing required text.

**Manually edit the regular expression**

This action is optional for the Select Fields and Validate steps.

You can manually edit the regular expression. However, doing this takes you out of the field extractor workflow. When you save your changes to the field extraction, the field extractor takes you to the final Save step.

1. Click \textbf{Show Regular Expression}.
2. Click \textbf{Edit the Regular Expression}.
   
   Click the \textbf{Back} button at the top left of the page if you want to abandon manual regular expression editing and return to the field extractor workflow. You can only go back if you have not yet tried to preview a regular expression change.
3. Edit the regular expression.
4. Click \textbf{Preview} to match your edited extraction against the sample events.
   
   The \textbf{Back} button disappears. The \textbf{Preview} button is grayed out until you make more edits to the field extraction.
Use the Filter, Sample, and Matches, and Non-Matches controls to help you assess the quality of your regular expression. Repeat steps 3 and 4 until the regular expression is matching events and extracting fields appropriately.

5. Click **Save** to save your new field extraction. The field extractor sends you to the Save step. When you enter the Save step, click **Back** to continue editing the regular expression. The **Back** button disappears after you enter a name for the extraction or make permissions choices.


**Field Extractor: Rename Fields step**

The Rename Fields step of the field extractor is for delimiter-based field extractions only. If you are extracting fields using a regular expression, see the topics for the Select Fields and Validate steps.

In the Rename Fields step you:

- **Identify the delimiter that separates the fields in your sample event**, such as a space, comma, tab, pipe, or another character or character combination. The field extractor breaks the event out into fields based on your delimiter choice
- **Rename one or more the fields that you want to extract from these events.**
- **Optionally preview the results of the delimiter-based field extraction.** This can help you validate the extraction and determine which fields to rename.
Identify a delimiter and rename one or more fields

Identify a delimiter. Rename at least one field.

1. Under Rename Fields, select one of the available Delimiter options or provide one of your own.
   The field extractor replaces the sample event with a display of the fields it finds in the event, using the delimiter that you select. It gives each field a color and a temporary name (field1, field2, field3 and so on).
   If you select Space, Comma, Tab, or Pipe, the field extractor breaks the event up into fields based on that delimiter. For example, a string like 2015-06-01T14:07:50:170Z|Jones|Alex|555-922-1212|324 Bowie Street|Alexandria, Va would get broken up into six separate fields if you choose Pipe as its delimiter.
   If the delimiter is not one of those four options, select Other, and enter the delimiter character or characters in the provided field. Then click the Return key to have the field extractor break up your event into fields based on that delimiter.
   The field extractor also creates a Preview area below the field display that previews how the delimiter-based field extraction works for other events in the dataset represented by your source or source type selection. See "Preview the results of the field extraction."

2. (Optional) Review the contents of the Preview section to determine the accuracy of the delimiter-based extraction and identify fields that should be renamed.
   This can help you make decisions about which fields to rename.
3. Click on a field that you want to rename.
   
   A Field Name field appears. Enter the correct field name.
You must select and rename at least one field to move on to the Save step.

4. Click Rename Field to rename the field.
   The field extractor replaces the field temporary name with the name you have provided throughout the page.

5. (Optional) Repeat steps 3 and 4 for all additional fields you choose to rename from the event.
   **Note:** You do not have to rename every field discovered by the field extractor.

6. Click Next to go to the Save step.

**Preview the results of the field extraction**

These actions are optional for the Rename Fields step.

After the field extractor applies delimiter-based field extraction to your sample event, the lower part of the page becomes a Preview section. You can go to the Preview section to preview the results of this extraction against the dataset represented by your chosen source or source type.

The Preview section has features that you can use to inspect the accuracy of the field extraction and identify fields that you may want to rename. It consists of a table that shows the events broken out into fields according to your delimiter choice. It also provides informational tabs for each field that the field extractor discovers.

1. (Optional) Change the sample size of the preview dataset to see statistics for a wider range of events.
   The preview section displays results for the **First 1,000 events** in the dataset by default. You can change the preview set to be the first 10,000 events or the events from the last five minutes, 24 hours, or 30 days.

2. (Optional) Review the first column to see if any events failed to match the pattern of the selected event.
The first column of the Preview event listing table displays a green check mark for events that match the pattern and a red "X" for events that do not match. If you have events that do not match, it means that those events may have more or fewer fields than your sample event, and you may want to try using a different delimiter or investigate why your chosen delimiter is only working for some events in your event set. You can quickly find rare matching or non-matching events by using the Matches and Non-Matches filters.

3. (Optional) Click a field tab to see information about it.

Each field information tab provides a value distribution for the field, organized from most to least common. It is based on the selected event sample. If the default sample of 1,000 isn't providing values that you expected to see, try changing it to a larger sample.

Field Extractor: Validate step

The Validate step of the field extractor is for regular-expression-based field extractions only.

Validate your field extraction in the Validate step of the field extractor. The field extractor provides the following validation methods:

- **Review the event list table to see which events match or fail to match the field extraction.** See "Preview the results of the field extraction".
- **Report incorrect extractions to the field extractor by providing counterexamples.** In response, the field extractor attempts to improve the accuracy of the regular expression.
- **Manually edit the regular expression.** See "Manually edit the regular expression".

When you are done validating your field extractions, click **Save** to save the extraction.

Provide counterexample feedback

This is an optional action for the Validate step.

If you find events that contain incorrectly extracted fields, submit those events as counterexample feedback.
1. Find an event with a field value that has been incorrectly extracted.
   The highlighted text is not a correct value for the field that the
   highlighter represents.
2. Click the gray "X" next to the incorrect field value.
   The field extractor displays the counterexample event above the
   table, marking the incorrect value with red strikethrough. It also
   updates the regular expression and its preview results.

3. If a counterexample does not help, remove it by clicking the blue "X" to the
   left of the counterexample event.

Field Extractor: Save step

In the Save step of the field extractor you define the name of the new field
extraction definition, set its permissions, and save the extraction.

1. Give the field extraction definition a name if it does not have one, or verify
   that the name that the field extractor provides is correct.
   If you created your field extraction definition with the regular
   expression mode, the Name will consist of a comma-separated list
   of the fields extracted by the definition. You can change this name.
   If you created your field extraction definition with the delims mode,
   Name will be blank. You must provide a name to save the field
   extraction definition.
   Note: The extraction name cannot include spaces.
2. (Optional) Change the Permissions of the field extraction to either App or
   All apps and update the role-based read/write permissions.
   You can only change field extraction permissions if your role
   includes the capability that allows you to do so.
   The field extraction is set to Owner, meaning that it only extracts
   fields in searches run by the person who created the extraction.
Set Permissions to App to make this extraction available only to users of the app that the field extraction belongs to. Set Permissions to All apps to enable all users of all apps to benefit from this field extraction when they run searches. When you change the app permissions to App or All apps you can set read and write permissions per role. See "Manage knowledge object permissions," in this manual.

Note: For delimiter-based field extractions, you will need to move the transforms.conf stanzas manually in order to change the field extraction permissions. You do not need to move props.conf stanzas. See App architecture and object ownership.

3. Click Finish to save the extraction.

You can manage the field extractions that you create. They are listed on the Field Extractions page in Settings. See Use the Field extractions page, in this manual.
Use the settings pages for field extractions in Splunk Web

Use the Field extractions page

Use the Field extractions page in Settings to manage search-time field extractions. There are three methods by which you can add search-time field extractions. You can:

- Use the field extractor to create extractions. This method is relatively easy and does not require you to understand how regular expressions work.
- Make direct edits to props.conf. You need Splunk Enterprise to use this method.
- Add new field extractions with the Field extractions page.

The Field extractions page enables you to:

- Review the overall set of search-time extractions that you have created or which your permissions enable you to see, for all Apps in your Splunk deployment.
- Create new search-time field extractions.
- Change permissions for field extractions. Field extractions created through the field extractor and the Field extractions page are initially only available to their creators until they are shared with others.
- Delete field extractions, if your app-level permissions enable you to do so, and if they are not default extractions that were delivered with the product. Default knowledge objects cannot be deleted. For more information about deleting knowledge objects, see Disable or delete knowledge objects.

If you have additional write permissions for a particular search-time field extraction, the Field extractions page also enables you to:

- Update its regular expression, if it is an inline field extraction.
- Add or delete named extractions that have been defined in transforms.conf or the Field transactions page in Splunk Web, if it uses transactions.

Note: You cannot manage index-time field extractions in Splunk Web. We do not recommend that you change your set of index-time field extractions, but if you need to, you have to modify your props.conf and transforms.conf configuration.
files manually. For more information about index-time field extraction configuration, see "Configure index-time field extractions" in the *Getting Data In Manual*.

Navigate to the Field extractions page by selecting **Settings > Fields > Field extractions**.

**Review search-time field extractions in Splunk Web**

To better understand how the Field extractions page displays your field extraction, it helps to understand how field extractions are set up in your *props.conf* and *transforms.conf* files.

Field extractions can be set up entirely in *props.conf*, in which case they are identified on the Field extractions page as inline field extractions. Some field extractions include a *transforms.conf* component, and these types of field extractions are called **transform field extractions**. To create or edit that component of the field extraction via Splunk Web, use the Field Transforms page in Splunk Web.

For more information about transforms and the Field Transforms page, see use the field transformations page.

For more information about field extraction setup directly in the *props.conf* and *transforms.conf* files see Create and maintain search-time field extractions through configuration files.

**Name column**

The **Name** column in the Field extractions page displays the overall name (or "class") of the field extraction. The field extraction format is:

```
<spec> : [EXTRACT-<class> | REPORT-<class>]
```

  - `<spec>` can be:
    - `<sourcetype>`, the source type of an event.
    - `host::<host>`, where `<host>` is the host for an event.
    - `source::<source>`, where `<source>` is the source for an event.

**EXTRACT-<class>** field extractions are extractions that are only defined in *props.conf*. They are created automatically by field extractions made through IFX and certain search commands. If you have Splunk Enterprise, you can also add them by making direct updates to the *props.conf* file. This kind of extraction
is always associated with a field-extracting regular expression. On the Field extractions page, this regex appears in the Extraction/Transform column.

REPORT-<class> field extractions reference field transform stanzas in transforms.conf. This is where their field-extracting regular expressions are located. On the Field extractions page, the referenced field transform stanza is indicated in the Extraction/Transform column.

You can work with transforms in Splunk Web through the Field Transformations page. See Use the Field Transformations page in Splunk Web.

**Type column**

There are two field extraction types: *inline* and *transforms.conf*.

- *Inline* extractions always have EXTRACT-<class> configurations. They are entirely defined within props.conf; they do not reference external field transforms.
- *Uses transform* extractions always have REPORT-<class> name configurations. As such they reference field transforms in transforms.conf. You can define field transforms directly in transforms.conf or via Splunk Web using the Field transformations page.

**Extraction Transform column**

In the Extraction/Transform column, Splunk Web displays different things depending on the field extraction Type.

- For *inline* extraction types, Splunk Web displays the regular expression that Splunk software uses to extract the field. The named group (or groups) within the regex show you what field(s) it extracts.

You can use regular expressions with inline field extractions to apply your inline field extraction to several sourcetypes. For example, you could have multiple sourcetypes named *foo_apache_access*, *bar_apache_access*, *baz_apache_access*, *quux_apache-access*. You can apply your field extraction to these sourcetypes by using the following as your sourcetype: *(?:\?)(\?)*_apache_access

For a primer on regular expression syntax and usage, see Regular-Expressions.info. You can test your regex by using it in a search with the rex search command.
• In the case of _Uses transform_ extraction types, Splunk Web displays the name of the _transforms.conf_ field transform stanza (or stanzas) that the field extraction is linked to through _props.conf_. A field extraction can reference multiple field transforms if you want to apply more than one field-extracting regex to the same source, source type, or host. This can be necessary in cases where the field or fields that you want to extract appear in two or more very different event patterns.

For example, the Expression column could display two values for a _Uses transform_ extraction: _access-extractions_ and _ip-extractions_. These may appear in _props.conf_ as:

```plaintext
[access_combined]
REPORT-access = access-extractions, ip-extractions
```

In this example, _access-extractions_ and _ip-extractions_ are both names of field transform stanzas in _transforms.conf_. To work with those field transforms through Splunk Web, go to the Field transforms page.

**Add new field extractions in Splunk Web**

Use Splunk Web to create new field extractions.

**Prerequisites**

• Regular expressions and field name syntax for information about field-extracting regular expressions.
• About default fields (host, source, source type, and more) for information about hosts, sources, and sourcetypes.

**Steps**

1. Select **Settings > Fields**.
2. Click **Field extractions** to go to the field extractions page.
3. Click **New** to create a new field extraction.
4. Define a **Destination app** context for the field extraction. By default it will be the app context you are currently in.
5. Give the field extraction a **Name**, using underscores for spaces between words. In _props.conf_ this is the `<class>` value for an EXTRACT or REPORT field extraction type. **Note:** `<class>` values do not have to follow field name syntax restrictions (see note below). You can use characters other than a-z, A-Z, and 0-9, and spaces are allowed. In addition `<class>` values are not subject to key cleaning.
6. Define the sourcetype, source, or host to which the extraction applies.
   Select <sourcetype>, <source>, or <host> and enter the value. This maps to the
   <spec> value in props.conf.
7. Define the extraction type.
   If you select <Uses transform>, enter the transform(s) involved in the
   <Extraction/Transform> field, separated by commas. The transform
   can then be created or updated with the Field transforms page.
   If you select <Inline>, enter the regular expression used to extract the
   field (or fields) in the <Extraction/Transform> field. For a primer on
   regular expression syntax and usage, see
   Regular-Expressions.info. You can test your regex by using it in a
   search with the rex search command. Splunk also maintains a list
   of useful third-party tools for writing and testing regular
   expressions.

**Example - Add a new error code field**

This shows how you would define an extraction for a new <err_code> field. The
field can be identified by the occurrence of <device_id>= followed by a word within
brackets and a text string terminating with a colon. The field should be extracted
from events related to the <testlog> source type.

In <props.conf> this extraction would look like:

```
[testlog]
EXTRACT-errors = device_id=\[w+\](?<err_code>[^:]+)
```

Here’s how you would set that up through the Add new field extractions page:
Note: You can find a version of this example in Create and maintain search-time field extractions, which shows you how to set up field extractions using the props.conf file.

Create a field from a subtoken

You may run into problems if you are extracting a field value that is a subtoken—a part of a larger token. Tokens are chunks of event data that have been run through event processing prior to being indexed. During event processing, events are broken up into segments, and each segment created is a token. You will need access to the .conf files in order to create a field from a subtoken. If you create a field from a subtoken in Splunk UI, your field extraction will show up but you will be unable to use it in search. For more information, see create a field from a subtoken.

Update existing field extractions

To edit an existing field extraction, click its name in the Name column.

This takes you to a details page for that field extraction. In the Extraction/Transform field what you can do depends on the type of extraction that you are working with.

- If the field extraction is an inline extraction, you can edit the regular expression it uses to extract fields.
- If the field extraction uses one or more transforms, you can update the transform or transforms involved (put them in a comma-separated list if there is more than one.) The transforms can then be created or updated via the Field transforms page.

The field extraction above uses three transforms: wel-message, wel-eq-kv, and wel-col-kv. To find out how these transforms are set up, go to Settings > Fields > Field Transformations or use transforms.conf.
Note: Transform field extractions must include at least one valid `transforms.conf` field extraction stanza name.

**Update field extraction permissions**

When a field extraction is created through an inline method (such as IFX or a search command) it is initially only available to its creator. To make it so that other users can use the field extraction, you need to update its permissions.

**Steps**

1. Select **Settings > Fields**.
2. Click **Field extractions** to go to the field extractions page.
3. Find the locate field extraction on the Field extractions page and click on its **Permissions**.

This opens the standard permission management page used in Splunk Web for knowledge objects. On this page, you can set up role-based permissions for the field extraction, and determine whether it is available to users of one specific App, or globally to users of all Apps. For more information about managing permissions with Splunk Web, see Manage knowledge object permissions.

**Delete field extractions in Splunk Web**

You can delete field extractions if your permissions enable you to do so. You will not be able to delete default field extractions (extractions delivered with the product and stored in the "default" directory of an app).

1. Navigate to **Settings > Fields > Field extractions**.
2. Click **Delete** for the field extraction you want to remove.

Note: Take care when deleting objects that have downstream dependencies. For example, if your field extraction is used in a search that in turn is the basis for an event type that is used by five other saved searches (two of which are the foundation of dashboard panels), all of those other knowledge objects will be negatively impacted by the removal of that extraction from the system. For more information about deleting knowledge objects, see Disable or delete knowledge objects.

**Configure field extractions with .conf files**

Inline and transform field extractions can be configured using .conf files. See
Configure custom fields at search time.

**Use the Field transformations page**

The Field transformations page in Settings lets you manage transform field extractions, which reside in transforms.conf. Field transforms can be created either through direct edits to transforms.conf or by addition through the Field transformations page.

Every field transform has at least one field extraction component.

The Field transformations page enables you to:

- Review the overall set of field transforms that you have created or which your permissions enable you to see, for all Apps in your Splunk deployment.
- Create new search-time field transforms. For more information about situations that call for the use of field transforms, see "When to use the Field transformations page," below.
- Update permissions for field transforms. Field transforms created through the Field transformations page are initially only available to their creators until they are shared with others. You can only update field transform permissions if you own the transform, or if your role’s permissions enable you to do so.
- Delete field transforms, if your app-level permissions enable you to do so, and if they are not default field transforms that were delivered with the product. Default knowledge objects cannot be deleted. For more information about deleting knowledge objects, see Disable or delete knowledge objects in this manual.

If you have "write" permissions for a particular field transform, the Field transformations page enables you to:

- Update its regular expression and change the key the regular expression applies to.
- Define or update the field transform format.

Navigate to the Field transformations page by selecting Settings > Fields > Field transformations.
Why set up a field transform for a field extraction?

While you can define most search-time field extractions entirely within props.conf or the Field extractions page in Splunk Web, some advanced search-time field extractions require a transforms.conf component called a field transform. These search-time field extractions are called **transform field extractions** and can be defined and managed through the Field transforms page.

Use a search-time field extractions with a field transform component when you need to:

- **Reuse the same field-extracting regular expression across multiple sources, source types, or hosts** (in other words, configure one field transform that is referenced by multiple field extractions). If you find yourself using the same regex to extract fields for different sources, source types, and hosts, you may want to set it up as a transform. Then, if you find that you need to update the regex, you only have to do so once, even though it is used by more than one field extraction.
- **Apply more than one field-extracting regular expression to the same source, source type, or host** (in other words, apply multiple field transforms to the same field extraction). This is sometimes necessary in cases where the field or fields that you want to extract from a particular source/source type/host appear in two or more very different event patterns.
- **Use a regular expression to extract fields from the values of another field** (also referred to as a "source key"). For example, you might pull a string out of a url field value, and have that be a value of a new field.

You can do more things with search-time field transforms (such as setting up delimiter based field extractions and configuring extractions for multi-value fields) if you configure them directly within transforms.conf. See the section on field transform setup in Configure advanced extractions with field transforms.

**Note:** All index-time field extractions are coupled with one or more field transforms. You cannot manage index-time field extractions in Splunk Web, however—you have to use the props.conf and transforms.conf configuration files. We don't recommend that you change your set of index-time field extractions under normal circumstances, but if you find that you must do so, see Create custom fields at index-time in the *Getting Data In* manual.
Review and update search-time field transforms in Splunk Web

To better understand how the Field transformations page in Splunk Web displays your field transforms, it helps to understand how search-time field extractions are set up in your props.conf and transforms.conf files.

A typical field transform looks like this in transforms.conf:

```
[banner]
REGEX = /js/(?<license_type>[^/]*)/(?<version>[^/]*)/login/(?<login>[^/]*)
SOURCE_KEY = uri
```

This transform matches its regex against uri field values, and extracts three fields as named groups: license_type, version, and login.

In props.conf, that transform is matched to the source .../banner_access_log* like so:

```
[source::.../banner_access_log*]
REPORT-banner = banner
```

This means the regex is only matched to uri fields in events coming from the .../banner_access_log source. But you can match it to other sources, sourcetypes, and hosts if necessary.

**Note:** By default, transforms are matched to a SOURCE_KEY value of _raw, in which case their regexes are applied to the entire event, not just fields within that event.

**The Name column**

The Name column of the Field transformations page displays the names of the search-time field transforms that your permissions enable you to see. These names are the actual stanza names for field transforms in transforms.conf. The transform example presented above would appear in the list of transforms as banner.

Click on a transform name to see the detail information for that particular transform.
**Reviewing and editing transform details**

The details page for a field transform enables you to view and update its regular expression, key, and event format. Here’s the details page for the `banner` transform that we described at the start of this subtopic:

If you have the permissions to do so, you can edit the regex, key, and event format. Keep in mind that these edits can affect multiple field extractions defined in `props.conf` and the Field extractions page, if the transform has been applied to more than one source, sourcetype, or host.

**Create a new field transform**

**Prerequisites**

- Regular expressions and field name syntax for information about field-extracting regular expressions.
- About default fields (host, source, source type, and more) for information about hosts, sources, and sourcetypes.
- Configure custom fields at search time for information on different types of field extraction.

**Steps**

1. Select **Settings > Fields** to navigate to the Fields manager page.
2. Select **Field transformations > New** to navigate to the Fields transformations page.
3. Identify the **Destination app** for the field transform, if it is not the app you are currently in.

4. Give the field transform a **Name**.
   This equates to the stanza name for the transform on `transforms.conf`. When you save this transform this is the name that appears in the **Name** column on the Field transformations page.

5. Enter a **Regular expression** for the transform.
   See Regular expressions and field name syntax.

6. (Optional) Define a **Key** for the transform.
   This corresponds to the `SOURCE_KEY` option in `transforms.conf`. By default it is set to `_raw`, which means the regular expression is applied to entire events.
   To have the regular expression be applied to values of a specific field, replace `_raw` with the name of that field. You can only use fields that are present when the field transform is executed.

7. (Optional) Specify the **Event format**.
   This corresponds to the `FORMAT` option in `transforms.conf`. You use `$n` to indicate groups captured by the regular expression. For example, if the regular expression you've designed captures two groups, you could have a Format set up like this: `$1::$2`, where the first group is the field name, and the second group is the field value.
   Or you could set Format up as `username::$1 userid::$2`, which means the regular expression extracts the values for the `username` and `userid` fields. The Format field defaults to `<transform_stanza_name>::$1`.

8. (Optional) Select **Create multvalue fields** if the same field can be extracted from your events more than once.
   This causes Splunk software to extract the field as a single multivalue field.

9. (Optional) Select **Automatically clean field names** to ensure that the extracted fields have valid names.
   Leading underscore characters and 0-9 numerical characters are removed from field names, and characters other than those falling within the a-z, A-Z, and 0-9 ranges in field names are replaced with underscores.

**Example - Extract both field names and their corresponding field values from an event**

You can use the **Event format** attribute in conjunction with a properly designed regular expression to set up a field transform that extracts both a field name and its corresponding field value from each matching event.
Here's an example, using a transform that is delivered with Splunk software.

The \texttt{bracket-space} field transform has a regular expression that finds field name/value pairs within brackets in event data. It will reapply this regular expression until all of the matching field/value pairs in an event are extracted.

As we stated earlier in this topic, field transforms are always associated with a field extraction. On the Field Extractions page in Splunk Web, you can see that the \texttt{bracket-space} field transform is associated with the \texttt{osx-asl:REPORT-asl} extraction.

**Update field transform permissions**

When a field transform is first created, by default it is only available to its creator. To make it so that other users can use the field transform, you need to update its permissions. To do this, locate the field transform on the Field transformations page and select its Permissions link. This opens the standard permission management page used in Splunk Web for knowledge objects.

On this page you can set up role-based permissions for the field transform, and determine whether it is available to users of one specific App, or globally to users of all Apps. For more information about managing permissions with Splunk Web, see Manage knowledge object permissions.

**Delete field transforms**

On the Field transformations page in Splunk Web, you can delete field
transforms if your permissions enable you to do so.

Click **Delete** for the field extraction that you want to remove.

**Note:** Take care when deleting knowledge objects that have downstream dependencies. For example, if the field extracted by your field transform is used in a search that in turn is the basis for an event type that is used by five other reports (two of which are the foundation of dashboard panels), all of those other knowledge objects will be negatively impacted by the removal of that transform from the system. For more information about deleting knowledge objects, see [Disable or delete knowledge objects](#).
Use the configuration files to configure field extractions

Configure custom fields at search time

Use configuration files to configure custom fields at search time, to enrich your events with fields that are not discovered by available Splunk Web extraction methods. You can use .conf files such as transforms.conf and props.conf to add, maintain, and review libraries of custom field additions.

You can set up and manage search-time field extractions via Splunk Web. You cannot configure automatic key-value field extractions through Splunk Web. For more information on setting up field extractions through Splunk Web, see manage search-time field extractions.

You can locate props.conf and transforms.conf in $SPLUNK_HOME/etc/system/local/, or your own custom app directory in $SPLUNK_HOME/etc/apps/.

In general, you should try to extract your fields at search time rather than at index-time. There are relatively few cases where index-time extractions are better, and they can cause an increase in index size making your searches slower. See Configuring index-time field extractions.

Field extraction configurations must include a regular expression that specifies how to find the field that you want to extract.

See About fields.

Types of field extraction

There are three field extraction types: inline, transform, and automatic key-value.

<table>
<thead>
<tr>
<th>Field extraction type</th>
<th>Configuration location</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inline extractions</td>
<td>Inline extractions have EXTRACT-&lt;class&gt; configurations in props.conf stanzas.</td>
<td>Configure inline extractions</td>
</tr>
</tbody>
</table>
Transform extractions have `REPORT-<class>` name configurations that are defined in `props.conf` stanzas. Their `props.conf` configurations must reference field transform stanzas in `transforms.conf`.

Configure advanced extractions with field transforms.

Automatic key-value extractions are configured in `props.conf` stanzas where `KV_MODE` is set to a valid value other than `none`.

Configure automatic key-value field extraction.

### When to use inline or transform extractions

<table>
<thead>
<tr>
<th>Field extraction type</th>
<th>Situation</th>
<th>See</th>
</tr>
</thead>
</table>
| Inline extractions    | • You have one regular expression per field extraction configuration.  
                         • You have a simple setup with one regular expression, and you want to  
                           extract multiple fields.  
                         • You want to create a new field by configuring an extraction.  | Configure inline extractions with `props.conf`                      |
| Transform extractions | • To reuse the same field-extracting regular expression across multiple  
                           sources, source types, or hosts.  
                         • To apply more than one field-extracting regular expression to the  
                           same source, source type, or host.  
                         • To set up delimiter-based field extractions.  
                         • To configure extractions for multivalue fields.  
                         • To extract fields with names that begin with numbers or underscores.  
                         • To extract fields from the values of another field.  
                         • To manage the formatting of extracted fields, in cases where you  
                           are extracting multiple fields or are extracting both the field  
                           name and field value.  | Configure advanced extractions with field transforms                  |
Both of these configurations can be set up in the regular expression as well.

**Configure inline extractions**

Inline field extractions are field extractions that are configured within `props.conf`. You can have one regular expression per field extraction configuration. See About configuration files in the *Admin* manual.

Use inline field extractions when you:

- Have one regular expression per field extraction configuration
- Have a simple setup with one regular expression, and you want to extract multiple fields
- Want to create a new field by configuring an extraction

**Inline extractions and the search-time operations sequence**

*Search-time operations order*

Inline field extractions come first in the search-time operation sequence.

*Restrictions*

Because inline field extractions happen first in the search-time operation sequence, they cannot reference fields that are derived and added to events by other search-time operations.

*For more information*

For more information, see The sequence of search-time operations.

**Configure an inline search-time field extraction**

Inline search-time field extractions use the EXTRACT extraction configuration in `props.conf`. Each EXTRACT extraction stanza contains the regular expression to extract fields at search time, and other attributes that govern the way those fields are extracted.

**Prerequisites**

Review the following topics.
• About default fields (host, source, source type, and more) for information about hosts, sources, and sourcetypes.
• fields.conf for information about adding an entry to fields.conf.
• Regular expressions and field name syntax for information about field-extracting regular expressions.
• Create a field from a subtoken for information subtoken field extraction.
• Access to the props.conf located in $SPLUNK_HOME/etc/system/local/, or in your custom app directory in $SPLUNK_HOME/etc/apps/.

Caution: Do not edit files in $SPLUNK_HOME/etc/system/default/. A subsequent upgrade or migration will overwrite your configuration and cause Splunk software to fail.

Steps

1. Identify the source type, source, or host that provide the events that your field should be extracted from.
   All extraction configurations in props.conf are restricted to a specific source, source type, or host.
2. Configure a regular expression that identifies the field in the event.
3. Follow the format for the EXTRACT field extraction type to configure a field extraction stanza in props.conf that includes the host, source, or sourcetype for the event and the regular expression that you have configured.
4. If your field value is a subtoken, you must also add an entry to fields.conf.
5. Restart Splunk Enterprise.

EXTRACT field extraction configuration syntax

<spec> options

[<spec>]
EXTRACT-<class> = [<regular_expression>|<regular_expression> in <string>]

<spec>
Syntax: <source type>| host::<host> | source::<source> | rule::<rulename>| delayedrule::<rulename>

<table>
<thead>
<tr>
<th>&lt;spec&gt;</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;source type&gt;</td>
<td>Source type of an event</td>
</tr>
</tbody>
</table>
Before using `rule` or `delayedrule`, try generating a new source type based on the source seen by Splunk software.

**EXTRACT configuration attributes**

<table>
<thead>
<tr>
<th>EXTRACT-&lt;class&gt;</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;class&gt;</code></td>
<td>A unique literal string that identifies the namespace of the field you're extracting. <code>&lt;class&gt;</code> values do not have to follow field name syntax restrictions and are not subject to key cleaning.</td>
</tr>
<tr>
<td><code>&lt;regular_expression&gt;</code></td>
<td>Required to have named capturing groups. Each group represents a different extracted field. When the <code>&lt;regular_expression&gt;</code> matches an event, the named capturing groups and their values are added to the event.</td>
</tr>
<tr>
<td><code>&lt;regular_expression&gt; in &lt;source_field&gt;</code></td>
<td>Matches a regular expression against the values of a specific field. Otherwise it matches all raw event data.</td>
</tr>
<tr>
<td><code>&lt;regular_expression&gt; in &lt;string&gt;</code></td>
<td>When <code>&lt;string&gt;</code> is not a field name, change the regular expression to end with <code>[i]n &lt;string&gt;</code> to ensure that Splunk software does not match <code>&lt;string&gt;</code> to a field name.</td>
</tr>
</tbody>
</table>

**Configure advanced extractions with field transforms**

A transform extraction is made up of two components: a field transform configuration in `transforms.conf` and a `REPORT-<class>` field extraction configuration in `props.conf`. You can find `transforms.conf` and `props.conf` in `$SPLUNK_HOME/etc/system/local`. This section shows you how to configure field transforms in `transforms.conf`. For configuring a field transform in Splunk Web, see manage field transforms.
In transform extractions, the regular expression is in `transforms.conf` and the field extraction is in `props.conf`. You can apply one regular expression to multiple field extraction configurations, or have multiple regular expressions for one field extraction configuration. See configure custom fields at search time.

Field transforms contain a field-extracting regular expression and other attributes that govern the way the transform extracts fields. Field transforms are always created in conjunction with field extraction stanzas in `props.conf`.

**Transform extractions and the search-time operations sequence**

**Search-time operation order**

Extraction transforms are second in the search-time operations sequence and are processed after inline field extractions.

**Restrictions**

Splunk software processes all inline field extractions that belong to a specific host, source, or source type in ASCII sort order according to their `<class>` value. You cannot reference a field extracted by `EXTRACT-aaa` in the field extraction definition for `EXTRACT-ZZZ`, but you can reference a field extracted by `EXTRACT-aaa` in the field extraction definition for `EXTRACT-ddd`.

*For more information*

See The sequence of search-time operations.

**Configure a transform extraction**

Transform extractions use the REPORT extraction configuration in `props.conf`. Each REPORT extraction stanza references a field transform that is defined in `transforms.conf`. The field transform contains the regular expression that Splunk Enterprise uses to extract fields at search time, and other attributes that govern the way that the transform extracts those fields.

**Caution:** Do not edit files in `$SPLUNK_HOME/etc/system/default/`. An upgrade or migration will overwrite your configuration and cause Splunk software to break.

**Prerequisites**

Review the following topics.
• Configure custom fields at search time for information on different types of field extraction.
• Configure inline extractions for information on configuring inline extractions.
• About default fields (host, source, source type, and more) for information about hosts, sources, and sourcetypes.
• Regular expressions and field name syntax for information about field-extracting regular expressions.
• Field transform syntax for information on the format for transform definitions.
• Syntax for transform configuration for the syntax of transformation extractions.
• Access the props.conf and the transforms.conf files, located in $SPLUNK_HOME/etc/system/local/, or in your custom app directory in $SPLUNK_HOME/etc/apps/.

Steps

1. Identify the source type, source, or host that provides the events that your field is extracted from.
   Extraction configurations in props.conf are restricted to a specific source, source type, or host.
2. Configure a regular expression that identifies the field in the event.
   If your event lists field/value pairs or field values, configure a delimiter-based field extraction that does not require a regular expression.
3. Configure a field transform in transforms.conf that utilizes this regular expression or delimiter configuration.
   The transform can define a source key and event value formatting.
4. Follow the format for the REPORT field extraction type to configure a field extraction stanza in props.conf that uses the host, source, or source type identified earlier.
5. (Optional) You can configure additional field extraction stanzas for other hosts, sources, and source types that refer to the same field transform.
6. Restart your Splunk deployment for your changes to take effect.

Field transform syntax

There are two ways to use transforms. One for regex-based field extractions and one for delimiter-based field extractions. Use the following format when you define a search-time field transform in transforms.conf:

```
[<unique_transform_stanza_name>]
```
The `<unique_transform_stanza_name>` is required for all search-time transforms. `<unique_transform_stanza_name>` values are not required to follow field name syntax restrictions. See field name syntax. You can use characters other than a-z, A-Z, and 0-9, and spaces are allowed. They are not subject to key cleaning.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>REGEX</td>
<td>Empty string</td>
<td>Required unless you are setting up an ASCII-only delimiter-based field extraction. See DELIMS.</td>
</tr>
<tr>
<td>FORMAT</td>
<td>Empty string</td>
<td>Optional</td>
</tr>
<tr>
<td>MATCH_LIMIT</td>
<td>100000</td>
<td>Optional</td>
</tr>
<tr>
<td>DEPTH_LIMIT</td>
<td>1000</td>
<td>Optional</td>
</tr>
<tr>
<td>SOURCE_KEY</td>
<td>_raw</td>
<td>Optional</td>
</tr>
<tr>
<td>DELIMS</td>
<td>Empty string</td>
<td>Optional</td>
</tr>
<tr>
<td>FIELDS</td>
<td>Empty string</td>
<td>Optional</td>
</tr>
<tr>
<td>MV_ADD</td>
<td>False</td>
<td>Optional</td>
</tr>
<tr>
<td>CLEAN_KEYS</td>
<td>True</td>
<td>Optional</td>
</tr>
<tr>
<td>KEEP_EMPTY_VALS</td>
<td>False</td>
<td>Optional</td>
</tr>
<tr>
<td>CAN_OPTIMIZE</td>
<td>True</td>
<td>Optional</td>
</tr>
</tbody>
</table>

Field transform syntax descriptions

Click Expand to see additional information, such as details and configuration examples, about each attribute.
**REGEX**

A regular expression that operates on your data to extract fields.

**REGEX and the FORMAT field**

Name-capturing groups in the **REGEX** are extracted directly to fields. You do not have to specify **FORMAT** for simple field extraction cases.

If the **REGEX** extracts both the field name and its corresponding value, you can use the following special capturing groups to avoid specifying the mapping in **FORMAT**: `<_KEY_>`<string>, `<_VAL_>`<string>.

**Example of **REGEX** and **FORMAT****

<table>
<thead>
<tr>
<th>Using <strong>FORMAT</strong></th>
<th>Not using <strong>FORMAT</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>REGEX = ([a-z]+)=([a-z]+) is equivalent to</td>
<td>REGEX = (?&lt;KEY_1&gt;[a-z]+)=(?&lt;VAL_1&gt;[a-z]+)</td>
</tr>
<tr>
<td><strong>FORMAT</strong> = $1::2</td>
<td></td>
</tr>
</tbody>
</table>

**Example of using **REGEX** for **DELIMS**-like functionality**

Use **REGEX** for a non-ASCII delimiter.

**Invalid **DELIMS**

<table>
<thead>
<tr>
<th><strong>DELIMS</strong></th>
<th><strong>REGEX</strong></th>
<th><strong>FORMAT</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;</td>
<td></td>
<td>&quot;, &quot;?=&quot;</td>
</tr>
</tbody>
</table>

| **DELIMS** = "||" | **FIELDS** = ace, bubbles, cupcake | **REGEX** = ^(?<ace>[^|]+)[|](?<bubbles>[^|]+)[|](?<cupcake>[^|]+) |

---

**FORMAT**

Use **FORMAT** to specify the format of the field/value pair(s) that you are extracting. You do not need to specify the **FORMAT** if you have a simple **REGEX** with name-capturing groups.
Configuration

For search-time extractions, the pattern for the \texttt{FORMAT} field is as follows:

\begin{verbatim}
FORMAT = \langle field-name\rangle::\langle field-value\rangle(\langle field-name\rangle::\langle field-value\rangle)*
\end{verbatim}

where: field-name = \{<string>|$<extracting-group-number>\} field-value = \{<string>|$<extracting-group-number>\}

Restrictions

You cannot create concatenated fields with \texttt{FORMAT} at search time. This functionality is available only for index-time field transforms. To concatenate a set of regular expression extractions into a single field value, use the \texttt{FORMAT} attribute as an index-time extraction. For example, if you have the string 192(x)0(y)2(z)1 in your event data, you can extract it at index time as an ip address field value in the format 192.0.2.1. See Configure index-time field extractions in the \textit{Getting Data In} manual. Do not make extensive changes to your set of indexed fields as it can negatively impact indexing performance and search times.

Example of search-time \texttt{FORMAT} usage

1. FORMAT = firstfield::$1 secondfield::$2 thirdfield::other-value
2. FORMAT = $1::$2

If you configure \texttt{FORMAT} with a variable field name, the regular expression is repeatedly applied to the source event text to match and extract all field/value pairs.

\begin{verbatim}
MATCH_LIMIT
\end{verbatim}

Use \texttt{MATCH_LIMIT} to set an upper bound on how many times PCRE calls an internal function, match(). If set too low, PCRE may fail to correctly match a pattern.

Configuration

Limits the amount of resources that are spent by PCRE when running patterns that will not match. Defaults to 100000.
**DEPTH_LIMIT**

Use `DEPTH_LIMIT` to limit the depth of nested backtracking in an internal PCRE function, `match()`. If set too low, PCRE might fail to correctly match a pattern.

**Configuration**

Limits the amount of resources that are spent by PCRE when running patterns that will not match. Defaults to 1000.

---

**SOURCE_KEY**

Use `SOURCE_KEY` to extract values from another field. You can use any field that is available at the time of the execution of this field extraction.

**Configuration**

To configure `SOURCE_KEY`, identify the field to which the transform's `REGEX` is to be applied.

---

**DELIMS**

Use `DELIMS` in place of `REGEX` when dealing with ASCII-only delimiter-based field extractions, where field values or field/value pairs are separated by delimiters such as commas, colons, spaces, tab spaces, line breaks, and so on.

**Configuration**

Each ASCII character in the delimiter string is used as a delimiter to split the event. If the event contains full delimiter-separated field value pairs, you enter two sets of quoted delimiters for `DELIMS`. The first set of quoted delimiters separates the field value pairs. The second set of quoted delimiters separates the field name from its corresponding value.

If the events contain only delimiter-separated values (no field names), use one set of quoted delimiters to separate the values. Use the `FIELDS` attribute to apply field names to the extracted values. Alternatively, Splunk software reads even tokens as field names and odd tokens as field values.

**Restrictions**
Delimiters must be specified within double quotes (DELIMS="|;"'). Special escape sequences are \t (tab), \n (newline), \r (carriage return), \ (backslash) and \" (double quotes). If a value contains an embedded unescaped double quote character, such as "foo"bar", use REGEX, not DELIMS. Non-ASCII delimiters require the use of REGEX. See REGEX for examples of usage of DELIMS-like functionality.

Example

The following example of DELIMS usage applies to an event where field value pairs are separated by '|' symbols, and the field names are separated from their corresponding values by '=' symbols.

```
[pipe_eq]
DELIMS = "|", "=
```

**FIELDS**

Use in conjunction with DELIMS when you perform delimiter-based field extraction, and you only have field values to extract. Use FIELDS to provide field names for the extracted field values in list format according to the order in which the values are extracted.

If field names contain spaces or commas, use " ". To escape, use \.

Example

Following is an example of a delimiter-based extraction where three field values appear in an event. They are separated by a comma and a space.

```
[commalist]
DELIMS = ", 
FIELDS = field1, field2, field3
```

**MV_ADD**

Use MV_ADD for events that have multiple occurrences of the same field with different values, and you want to keep each value.

Configuration

When MV_ADD = true, Splunk software transforms fields that appear multiple times in an event with different values into multivalue fields. The field name
appears once. The multiple values for the field follow the = sign.

When \texttt{MV\_ADD = false}, Splunk software keeps the first value found for a field in an event, and discards every subsequent value found.

\textbf{CLEAN\_KEYS}

Controls whether the system strips leading underscores and 0-9 characters from the field names it extracts. Key cleaning is the practice of replacing any non-alphanumeric characters in field names with underscores, as well as the removal of leading underscores and 0-9 characters from field names.

\textbf{Configuration}

Add \texttt{CLEAN\_KEYS = false} to your transform to keep your field names intact with no removal of leading underscores or 0-9 characters.

\textbf{KEEP\_EMPTY\_VALS}

Controls whether Splunk software keeps field value pairs when the value is an empty string.

This option does not apply to field/value pairs that are generated by the Splunk software autoKV extraction (automatic field extraction) process. AutoKV ignores field/value pairs with empty values.

\textbf{CAN\_OPTIMIZE}

Controls whether Splunk software can disable the extraction.

Use \texttt{CAN\_OPTIMIZE} when you run searches under a search mode setting that disables field discovery to ensure that Splunk software discovers specific fields. Splunk software disables an extraction when none of the fields identified by the extraction are needed for the evaluation of a search.

\textbf{Syntax for a transform-referencing field extraction configuration}

To set up a search-time field extraction in \texttt{props.conf} that is associated with a field transform, use the \texttt{REPORT} field extraction class. Use the following format.
[<spec>]
REPORT-<class> = <unique_transform_stanza_name1>,
<unique_transform_stanza_name2>,...

<table>
<thead>
<tr>
<th>&lt;spec&gt;</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source type</td>
<td>Source type of an event</td>
</tr>
<tr>
<td>host::&lt;host&gt;</td>
<td>Host for an event</td>
</tr>
<tr>
<td>source::&lt;source&gt;</td>
<td>Source for an event</td>
</tr>
</tbody>
</table>

You can associate multiple field transform stanzas to a single field extraction by listing them after the initial <unique_transform_stanza_name>, separated by commas. See examples of transform extractions.

<table>
<thead>
<tr>
<th>REPORT-&lt;class&gt;</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;class&gt;</td>
<td>A unique literal string that identifies the namespace of the field you are extracting. &lt;class&gt; values do not have to follow field name syntax restrictions and are not subject to key cleaning.</td>
</tr>
<tr>
<td>&lt;unique_transform_stanza_name&gt;</td>
<td>Name of your field transform stanza from transforms.conf.</td>
</tr>
</tbody>
</table>

**Configure automatic key-value field extraction**

**Automatic key-value field extraction** is a search-time field extraction configuration that uses the KV_MODE attribute to automatically extract fields for events associated with a specific host, source, or source type. Configure automatic key-value field extractions by finding or creating the appropriate stanza in props.conf. You can find props.conf in $$SPLUNK_HOME/etc/system/local/ or your own custom app directory in $$SPLUNK_HOME/etc/apps/.

Automatic key-value field extraction is not explicit. You cannot configure it to find a specific field or set of fields. It looks for key-value patterns in events and extracts them as field/value pairs. You can configure it to extract fields from structured data formats like JSON, CSV, and from table-formatted events. Automatic key-value field extraction cannot be configured in Splunk Web, and cannot be used for index-time field extractions.
Automatic key-value field extraction and the sequence of search operations

Search-time operation order

Automatic key-value field extraction is third in the sequence of search operations, before field aliases and after transform extractions.

Restrictions

Splunk software processes automatic key-value field extractions in the order that it finds them in events.

For more information

See search time operations sequence.

Automatic key-value field extraction format

The following is the format for autoKV field extraction.

\[
\text{KV\_MODE} = [\text{none}|\text{auto}|\text{auto\_escaped}|\text{multi}|\text{json}|\text{xml}]
\]

<table>
<thead>
<tr>
<th>KV_MODE value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>Disables field extraction for the source, source type, or host identified by the stanza name. Use this setting to ensure that other regular expressions that you create are not overridden by automatic field/value extraction for a particular source, source type, or host. Use this setting to increase search performance by disabling extraction for common but nonessential fields. We have some field extraction examples at the end of this topic that demonstrate the disabling of field extraction in different circumstances.</td>
</tr>
<tr>
<td>auto</td>
<td>This is the default field extraction behavior if you do not include this attribute in your field extraction stanza. Extracts field/value pairs and separates them with equal signs.</td>
</tr>
<tr>
<td>auto_escaped</td>
<td>Extracts field/value pairs and separates them with equal signs, and ensures that Splunk Enterprise recognizes &quot; and \ as escaped sequences within quoted values. For example: field=&quot;value with &quot;nested&quot; quotes&quot;.</td>
</tr>
<tr>
<td>Multi</td>
<td>Invokes the <code>multikv</code> search command, which extracts field values from table-formatted events.</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>XML</td>
<td>Use this setting to use the field extraction stanza to extract fields from XML data. This mode does not extract non-XML data.</td>
</tr>
<tr>
<td>JSON</td>
<td>Use this setting to use the field extraction stanza to extract fields from JSON data. This mode does not extract non-JSON data.</td>
</tr>
</tbody>
</table>

**Note:** If you set `KV_MODE = json`, do not set `INDEXED_FIELDS = JSON` for the same source type, the json fields are extracted twice, at index time and at search time.

When `KV_MODE` is set to `auto` or `auto_escaped`, automatic JSON field extraction can take place alongside other automatic key/value field extractions. To disable JSON field extraction without changing the `KV_MODE` value from `auto`, add `AUTO_KV_JSON=false` to the stanza. When not set, `AUTO_KV_JSON` defaults to `true`.

### Disabling automatic extractions for specific sources, source types, or hosts

You can disable automatic search-time field extraction for specific sources, source types, or hosts in `props.conf`. Add `KV_MODE = none` for the appropriate ` [<spec> ]` in `props.conf`. When automatic key-value field extraction is disabled, explicit field extraction still takes place.

Custom field extractions set up manually via the configuration files or Splunk Web will still be processed for the affected source, source type, or host when `KV_MODE = none`.

```
[<spec>]
KV_MODE = none
<spec> can be:

- `<sourcetype>`, where `<sourcetype>` is the event source type.
- `host::<host>`, where `<host>` is the host for an event.
- `source::<source>`, where `<source>` is the source for an event.
```
Example inline field extraction configurations

The following are examples of inline field extraction, using props.conf.

Add an error code field

Create an error code field by configuring a field extraction in props.conf. The field is identified by the occurrence of device_id= followed by a word within brackets and a text string terminating with a colon. The field is extracted from the testlog source type.

In props.conf, add the following line:

```
[testlog]
EXTRACT-errors = device_id=[w+]\(?<err_code>[^:]+\)
```

Extract multiple fields by using one regular expression

The following is an example of a field extraction of five fields. A sample of the event data follows.

```
#%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet9/16, changed state to down
```

The stanza in props.conf for the extraction looks like this:

```
[syslog]
EXTRACT-port_flapping = Interface\s(?<interface>(?<media>[^d]+)(?<slot>[^d]+)/(?<port>[^d]+))\schanged\sstate\sto\s(?<port_status>up|down)
```

Five fields are extracted as named groups: interface, media, slot, port, and port_status.

Use extracted fields to report port flapping events.

1. Use tags to define event types in eventtypes.conf:

   ```
   [cisco_ios_port_down]
   search = "changed state to down"
   
   [cisco_ios_port_up]
   search = "changed state to up"
   ```

2. Create a report in savedsearches.conf that ties much of the above together to find port flapping and report on the results:
[port flapping]
search = eventtype=cisco_ios_port_down OR eventtype=cisco_ios_port_up starthoursago=3 | stats count by interface,host,port_status | sort -count

You can then use these fields with some event types to help you find port flapping events and report on them.

Create a field from a subtoken

You may run into problems if you are extracting a field value that is a subtoken—a part of a larger token. Tokens are chunks of event data that have been run through event processing prior to being indexed. During event processing, events are broken up into segments, and each segment created is a token.

Example

Tokens are never smaller than a complete word or number. For example, you may have the word foo123 in your event. If it has been run through event processing and indexing, it is a token, and it can be a value of a field. However, if your extraction pulls out the foo as a field value unto itself, you're extracting a subtoken. The problem is that while foo123 exists in the index, foo does not, which means that you'll likely get few results if you search on that subtoken, even though it may appear to be extracted correctly in your search results.

Because tokens cannot be smaller than individual words within strings, a field extraction of a subtoken (a part of a word) can cause problems because subtokens will not themselves be in the index, only the larger word of which they are a part.

1. (Optional) If your field value is a smaller part of a token, you must configure props.conf as explained here.
2. Add an entry to fields.conf.

[<fieldname>]
INDEXED = False
INDEXED_VALUE = False

♦ Fill in <fieldname> with the name of your field.
♦ For example, [url] if you've configured a field named "url."
♦ Set INDEXED and INDEXED_VALUE to false.
♦ This setting specifies that the value you're searching for is not a token in the index.
You do not need to add this entry to fields.conf for cases where you are extracting a field's value from the value of a default field (such as host, source, sourcetype, or timestamp) that is not indexed and therefore not tokenized.

For more information on the tokenization of event data, see About segmentation in the Getting Data In Manual.

Example transform field extraction configurations

These examples present transform field extraction use cases that require you to configure one or more field transform stanzas in transforms.conf and then reference them in a props.conf field extraction stanza.

Configure a field extraction that uses multiple field transforms

You can create transforms that pull field name/value pairs from events, and you can create a field extraction that references two or more field transforms.

Scenario

You have logs that contain multiple field name/field value pairs. While the fields vary from event to event, the pairs always appear in one of two formats.

The logs often come in this format:

[fieldName1=fieldValue1] [fieldName2=fieldValue2]

However, sometimes they are more complicated, logging multiple name/value pairs as a list where the format looks like:

[headerName=fieldName1] [headerValue=fieldValue1], [headerName=fieldName2] [headerValue=fieldValue2]

The list items are separated by commas, and each fieldName is matched with a corresponding fieldValue. In this scenario, you want to pull out the field names and values so that the search results are

fieldName1=fieldValue1
fieldName2=fieldValue2

Here’s an example of an HTTP request event that combines both of the above formats.

[method=GET] [IP=10.1.1.1] [headerName=Host]
[headerValue=www.example.com], [headerName=User-Agent]
You want to develop a single field extraction that would pull the following field/value pairs from that event.

```
method=GET
IP=10.1.1.1
Host=www.example.com
User-Agent=Mozilla
Connection=close
byteCount=255
```

**Solution**

You want to design two different regular expressions that are optimized for each format. One regular expression will identify events with the first format and pull out all of the matching field/value pairs. The other regular expression will identify events with the other format and pull out those field/value pairs.

Create two unique transforms in `transforms.conf`--one for each regex--and then connect them in the corresponding field extraction stanza in `props.conf`.

**Steps**

1. The first transform you add to `transforms.conf` catches the fairly conventional `[fieldName1=fieldValue1] [fieldName2=fieldValue2]` case.
   
   ```
   [myplaintransform]
   REGEX=\[(?!(?:headerName|headerValue))\[\]([^\s=]+)=([^\]]+)\]
   FORMAT=$1::$2
   ```

2. The second transform added to `transforms.conf` catches the slightly more complex `[headerName=fieldName1] [headerValue=fieldValue1], [headerName=fieldName2] [headerValue=fieldValue2] case:
   
   ```
   [mytransform]
   REGEX= \[headerName=(\w+)\], \s\[headerValue=([^\]]+)\]
   FORMAT= $1::$2
   ```

Both transforms use the `<fieldName>::<fieldValue> FORMAT` to match each field name in the event with its corresponding value. This setting in `FORMAT` enables Splunk Enterprise to keep matching the regular expression against a matching event until every matching field/value combination is extracted.

3. This field extraction stanza, created in `props.conf`, references both of the field transforms:
Besides using multiple field transforms, the field extraction stanza also sets KV_MODE=none. This disables automatic key-value field extraction for the identified source type while letting your manually defined extractions continue. This ensures that these new regular expressions are not overridden by automatic field extraction, and it also helps increase your search performance.

For more information on automatic key-value field extraction, see Automatic key-value field extraction for search-time data.

**Configure delimiter-based field extractions**

You can use the **DELIMS** attribute in field transforms to configure field extractions for events where field values or field/value pairs are separated by delimiters such as commas, colons, tab spaces, and more.

You have a recurring multiline event where a different field/value pair sits on a separate line, and each pair is separated by a colon followed by a tab space. Here's a sample event:

```
ComponentId:     Application Server
ProcessId:   5316
ThreadId:    00000000
ThreadName:  P=901265:O=0:CT
SourceId:    com.ibm.ws.runtime.WsServerImpl
ClassName:    
MethodName:    
Manufacturer:    IBM
Product:     WebSphere
Version:     Platform 7.0.0.7 [BASE 7.0.0.7 cf070942.55]
ServerName:  sfeserv36Node01Cell\sfeserv36Node01\server1
TimeStamp:   2010-04-27 09:15:57.671000000
UnitOfWork:    
Severity:    3
Category:    AUDIT
PrimaryMessage:  WSVR0001I: Server server1 open for e-business
ExtendedMessage:
```

**Steps**

1. Configure the following stanza in `transforms.conf`:

```plaintext
[activity_report]
```

```
KV_MODE=none
REPORT-a = mytransform, myplaintransform
```
DELIMS = "\n", ":\t"
This states that the field/value pairs in the event are on separate lines
("\n"), and then specifies that the field name and field value on each line
is separated by a colon and tab space (":\t").
2. Rewrite the props.conf stanza above as:

[activitylog]
LINE_BREAKER = [-]{8,}([[\r\n]+)
SHOULD_LINEMERGE = false
REPORT-activity = activity_report

These two brief configurations will extract the same set of fields as before, but
they leave less room for error and are more flexible.

Handling events with multivalue fields

You can use the MV_ADD attribute to extract fields in situations where the same
field is used more than once in an event, but has a different value each time.
Ordinarily, Splunk Enterprise only extracts the first occurrence of a field in an
event; every subsequent occurrence is discarded. But when MV_ADD is set to true
in transforms.conf, Splunk Enterprise treats the field like a multivalue field and
extracts each unique field/value pair in the event.

Example

You have a set of events.

event1.epochtime=1282182111 type=type1 value=value1 type=type3
   value=value3
event2.epochtime=1282182111 type=type2 value=value4 type=type3
   value=value5 type=type4 value=value6

The type and value fields are repeated several times in each event. In order to
have search type=type3 return both events or to run a count(type) report on the
two events that returns 5, create a custom multivalue extraction of the type field
for these events.

Steps Set up your transforms.conf and props.conf files to configure multivalue
extraction.

1. In transforms.conf, add the following.

   [mv-type]
   REGEX = type=\(<type>\s+\)
   MV_ADD = true
Configure extractions of multivalue fields with fields.conf

Multivalue fields are fields that can appear multiple times in an event and have a different value for each appearance. One of the more common examples of multivalue fields is that of email address fields, which typically appears two to three times in a single sendmail event?once for the sender, another time for the list of recipients, and possibly a third time for the list of Cc addresses, if one exists. If all of these fields are labeled identically (as AddressList, for example), they lose meaning that they might otherwise have if they’re identified separately as From, To, and Cc.

Multivalue fields are parsed at search time, which enables you to process the resulting values in the search pipeline. Search commands that work with multivalue fields include makemv, mvcombine, mvexpand, and nomv. For more information on these and other commands see Manipulate and evaluate fields with multiple values in the Search Manual. The complete command reference is in the Search Reference manual.

Use the TOKENIZER setting to define a multivalue field in fields.conf

You can use the TOKENIZER setting to define a multivalue field in fields.conf. At search time, TOKENIZER uses a regular expression to tell the Splunk platform how to recognize and extract multiple field values for a recurring field in an event.

The TOKENIZER setting is used by the where, timeline, and stats commands. It also provides the summary and XML outputs of the asynchronous search API.

Tokenization of indexed fields (fields extracted at index time) is not supported. If you have set INDEXED=true for a field, you cannot also use the TOKENIZER setting for that field. You can use a transform extraction defined in props.conf and transforms.conf to break an indexed field into multiple values.

Prerequisites

- Review the TOKENIZER multivalue field configuration syntax.
• See fields.conf in the Admin Manual to learn how the fields.conf file works.
• For an overview of configuration file usage in the Splunk platform, see About configuration files in the Admin Manual.
• For a primer on regular expression syntax and usage, see About Splunk regular expressions. You can test regular expressions by using them in searches with the rex search command.

Steps

1. Open the fields.conf file that you want to edit.
   If you have Splunk Enterprise, you edit fields.conf in $SPLUNK_HOME/etc/system/local/, or your own custom app directory in $SPLUNK_HOME/etc/apps/.
2. Add a stanza for the multivalue field. The stanza name should be the name of the field.
3. Add a line in the stanza that matches the TOKENIZER setting with a regular expression that is designed to capture multiple values for a field.
4. (Optional) If you have other attributes to set for the multivalue field, set them in the same stanza underneath the TOKENIZER line.
5. Save your changes to the file.

TOKENIZER multivalue field configuration syntax

[<field name 1>]
TOKENIZER = <regular expression>

[<field name 2>]
TOKENIZER = <regular expression>

• <regular expression> should be designed to capture multiple values for a field. For example, if a field name is followed by a list of email addresses, the regular expression should be able to extract each individual address as a separate value of the field without capturing delimiters like commas and spaces.
• TOKENIZER defaults to empty. When TOKENIZER is empty, the field can only take on a single value.
• When TOKENIZER is not empty, the first group is taken from each match to form the set of field values.
• TOKENIZER separates the multiple values of a field with the following delimiter characters: \n.
Example

You start with a poorly formatted email log file where all of the addresses involved are grouped together under AddressList. Here is a sample from that log file.

From: sender@splunkexample.com
To: recipient1@splunkexample.com, recipient2@splunkexample.com, recipient3@splunkexample.com
CC: cc1@splunkexample.com, cc2@splunkexample.com, cc3@splunkexample.com
Subject: Multivalue fields are out there!
X-Mailer: Febooti Automation Workshop (Unregistered)
Content-Type: text/plain; charset=UTF-8
Date: Wed, 3 Nov 2017 17:13:54 +0200
X-Priority: 3 (normal)

This example from $SPLUNK_HOME/etc/system/README/fields.conf.example breaks email fields To, From, and CC into multiple values.

[To]
TOKENIZER = (\w[\w\.-]*@[\w\.-]*\w)

[From]
TOKENIZER = (\w[\w\.-]*@[\w\.-]*\w)

[Cc]
TOKENIZER = (\w[\w\.-]*@[\w\.-]*\w)

Because the TOKENIZER process adds a \n delimiter between each value it extracts for a field, the multiple values for To in the sample event for this example will display like this:

recipient1@splunkexample.com
recipient2@splunkexample.com
recipient3@splunkexample.com
Calculated fields

About calculated fields

Calculated fields are fields added to events at search time that perform calculations with the values of two or more fields already present in those events. Use calculated fields as a shortcut for performing repetitive, long, or complex transformations using the eval command.

The `eval` command enables you to write an expression that uses extracted fields and creates a new field that takes the value that is the result of that expression's evaluation. For more information, see `eval`.

Eval expressions can be complex. If you need to use a long and complex eval expression on a regular basis, retyping the expression accurately can be tedious.

Calculated fields enable you to define fields with eval expressions. When writing a search, you can cut out the eval expression and reference the field like any other extracted field. The fields are extracted at search time and added to events that include the fields in the eval expressions.

You can create calculated fields in Splunk Web and in `props.conf`. For information on creating calculated fields in Splunk Web, see Create calculated fields with Splunk Web. For information on creating calculated fields with `props.conf`, see Configure calculated fields with `props.conf`.

Calculated fields and the search-time operations sequence

When you run a search, Splunk software runs several operations to derive knowledge objects and apply them to events returned by the search. Splunk software performs these operations in a specific sequence.

**Search-time operations order**

Calculated fields come fifth in the search-time operations sequence, after field aliasing but before lookups.

**Restrictions**

All `EVAL=<fieldname>` configurations within a single `props.conf` stanza are processed in parallel, rather than in any particular sequence. This means you
cannot "chain" calculated field expressions, where the evaluation of one calculated field is used in the expression for another calculated field.

Calculated fields can reference all types of field extractions and field aliasing, but they cannot reference lookups, event types, or tags.

**For more information**

For more information about search-time operations, see search-time operations sequence.

**Preventing overrides of existing fields**

If a calculated field has the same name as a field that has been extracted by normal means, the calculated field will override the extracted field, even if the `eval` statement evaluates to null. You can cancel this override with the `coalesce` function for `eval` in conjunction with the `eval` expression. Coalesce takes an arbitrary number of arguments and returns the first value that is not null.

If you do not want the calculated field to override existing fields when the `eval` statement returns a value, use:

\[
\text{EVAL-field} = \text{coalesce(field, <eval expression>)}
\]

If you do not want the calculated field to override existing fields when the `eval` statement returns null, use:

\[
\text{EVAL-field} = \text{coalesce(<eval expression>, field)}
\]

For more information about `coalesce` and other `eval` functions, see *evaluation functions* in the Search Reference.

**Calculated fields independence**

When Splunk software evaluates calculated fields, it evaluates each expression as if it were independent of all other fields. You cannot chain calculated field expressions, where the evaluation of one calculated field is used in the expression for another calculated field.

In the following example, for any individual event, the value of \( x \) is equivalent to the value of calculated field \( y \) because the two calculations are carried out independently of each other. Both expressions use the original value of \( x \) when they calculate \( x*2 \).
EVAL-x = x * 2
EVAL-y = x * 2
For a specific event x=4, these calculated fields would replace the value of x with 8, and would add y=8 to the event.

Another example which involves the extracted field response_time. When it is first extracted, the value of response_time is expressed in milliseconds. Here are two calculated fields that make use of response_time in different ways.

EVAL-response_time =  response_time/1000
EVAL-bitrate = bytes*1000/response_time
In this example, two things are happening with the access_common sourcetype.

• The first EVAL changes the value of the response_time in all sourcetype=access_common events so that it is expressed in seconds rather than milliseconds. The new “in seconds” value overrides the old “in milliseconds” value.
• The second EVAL calculates a new field called bitrate for all sourcetype=access_common events. It is expressed in terms of bytes per second. Bytes is another extracted field.

In both calculations, response_time is initially expressed in terms of milliseconds, as both EVALs are calculated independently of the other.

Create calculated fields with Splunk Web

Create a calculated field with Splunk Web. Use calculated fields as a shortcut for performing repetitive, long, or complex transformations using the eval command.

Prerequisites

• Review About calculated fields.

Creating a new calculated field in settings

1. Select Settings > Fields.
2. Select Calculated Fields > New.
3. Select the app that will use the calculated field.
4. Select host, source, or sourcetype to apply to the calculated field and specify a name.

You can also enter a wildcard if you want to apply this for all hosts, sources, or sourcetypes.

5. Name the resultant calculated field.

6. Define the eval expression.

Configure calculated fields with props.conf

To create a calculated field, add a calculated field key to a new or preexisting props.conf stanza. You can find props.conf in $SPLUNK_HOME/etc/system/local/, or your own custom app directory in $SPLUNK_HOME/etc/apps/. Best practices for transferring your data customizations to other search servers suggest using your own custom app directory.

Do not edit files in $SPLUNK_HOME/etc/system/default/.

For more information on configuration files, see About configuration files.

The format of a calculated field key in props.conf is:

```
[<stanza>]
EVAL-<field_name> = <eval statement>
```

- `<stanza>` can be:
  - `<source type>`, the source type of an event.
  - `host::<host>`, where `<host>` is the host for an event.
  - `source::<source>`, where `<source>` is the source for an event.

- Calculated field keys must start with "EVAL-" (including the hyphen), but "EVAL" is not case-sensitive (can be "eVaL" for example).
- `<field_name>` is case sensitive. This is consistent with all other field names in Splunk software.
- `<eval_statement>` is as flexible as it is for the eval search command. It can be evaluated to any value type, including multivals, boolean, or null.

Calculated fields with props.conf example
Prerequisites

- Review About calculated fields for more information about calculated fields.

- Review this example search from the Search Reference discussion of the `eval` command. This example examines earthquake data and classifies quakes by their depth by creating a `Description` field:

```
source=eqs7day-M1.csv | eval Description=case(Depth<=70, "Shallow", Depth>70 AND Depth<=300, "Mid", Depth>300 AND Depth<=700, "Deep") | table Datetime, Region, Depth, Description
```

Steps

Using calculated fields, you could define the eval expression for the `Description` field in `props.conf`.

1. Create the following stanza in `props.conf`.

```
<Stanza>
Eval-Description = case(Depth<=70, "Shallow", Depth>70 AND Depth<=300, "Mid", Depth>300 AND Depth<=700, "Deep")
</Stanza>
```

2. Rewrite the search as:

```
source=eqs7day-M1.csv | table Datetime, Region, Depth, Description
```

You can now search on `Description` as if it is any other extracted field. Splunk software will find the calculated field key and evaluate it for every event that contains a `Depth` field. You can also run searches like this:

```
source=eqs7day-M1.csv Description=Deep
```

After defining a calculated field key, Splunk software calculates the field at search time for events that have the extracted fields that appear in the eval statement. Calculated field evaluation takes place after `search-time field extraction` and `field aliasing`, but before derivation of `lookup fields`.
Event types

About event types

Event types are a categorization system to help you make sense of your data. Event types let you sift through huge amounts of data, find similar patterns, and create alerts and reports.

Note: Using event types as a short cut for search is not recommended. If you want to shorten a portion of a search, it is much better to use a search macro. Search macros are more flexible in what they can express, can include other search commands and not just base query terms, can be parameterized, and do not incur costs when events are retrieved. This can sometimes be easier to manage, because, for example, a single search macro can take the place of multiple event types.

For more information about using search macros, see using search macros in searches.

Event types and the search-time operations sequence

When you run a search, Splunk software runs several operations to derive knowledge objects and apply them to events returned by the search. Splunk software performs these operations in a specific sequence.

Search-time operations order

Event types come seventh in the search-time operations order, before tags but after lookups.

Restrictions

Splunk software processes event types first by priority score and then by ASCII sort order. Search strings that define event types cannot reference tags, because event types are always processed and added to events before tags.

For more information

For more information about search-time operations, see search-time operations sequence.
How event types work

Every event that can be returned by that search gets an association with that event type. For example, say you have this search:

sourcetype=access_combined status=200 action=purchase

If you save that search as an event type named successful_purchase, any event that can be returned by that search gets eventtype=successful_purchase added to it at search time. This happens even if you are searching for something completely different.

Note: Using event types can consume a lot of data, because any search attempts to correlate events with any known event type. As more event types are defined, the cost in search performance goes up. You can examine the execution costs of search commands with the command.search.typer parameter. See search job inspector.

To build a search that works with events that match that event type, include eventtype=access_combined as a search term.

A single event can match multiple event types. When an event matches two or more event types, eventtype acts as a multi-value field.

Important event type definition restrictions

You cannot base an event type on a search that:

- Includes a pipe operator after a simple search.
- Includes a subsearch.
- Is defined by a simple search that uses the savedsearch command to reference a report name. For example, if you have a report named failed_login_search, you should not use this search to define the event type: | savedsearch failed_login_search. In this case you should instead use the search string that defines failed_login_search as the definition of the event type.

This last point is more of a best practice than a strict limitation. You want to avoid situations where the search string underneath failed_login_search is modified by another user at a future date, possibly in a way that breaks the event type. You have more control over the ongoing validity of the event type if you use actual search strings in its definition.
**Note:** If you want to use event types as a way to short cut your search, use a search macro. For more information on event types vs search macros, see About event types.

**Creating event types**

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. For more information about saving searches as event types, see Define and maintain event types in Splunk Web.

You can also create new event types by modifying `eventtypes.conf`. For more information about manually configuring event types in this manner, see Configure event types directly in eventtypes.conf.

**Event type tags**

Event types can have one or more tags associated with them. You can add these tags while you save a search as an event type and from the event type manager, located in **Settings > Event types**. From the list of event types in this window, select the one you want to edit.

Tag event types to organize your data into categories. There can be multiple tags per event. You can tag an event type in Splunk Web or configure it in `tags.conf`. For more information about event type tagging, see Tag event types.

**Event type tags example #1**

Use event type tags to help track abstract field values such as HTTP access logs, IP addresses, or ID numbers by giving them more descriptive names. Add tags to event types by going to **Settings > Event types**. Select the event type from the list of event types in this menu.

After you add tags to your event types, search for them in the same way you search for any tag.

Let’s say that we have saved a search for page not found as the event type `status=404` and then saved a search for failed authentication as the event type `status=403`. If you tagged both of these event types with **HTTP client error**, all events of either of those event types can be retrieved by using the search:
Event type tags example #2

Event type tags are commonly used in the Common Information Model (CIM) add-on for the Splunk platform in order to normalize newly indexed data from an unfamiliar source type. We can use tags to identify different event types within a single data source.

You can apply CIM-compliant tags to your data.

1. From Splunk Web, select Settings > Data Models. Find the data model dataset that you want to map your data to, then identify its associated tags. For example, the cpu_load_percent object in the Performance data model has the following tags associated with it:
   ```
   tag = performance
   tag = cpu
   ```

2. Create the appropriate event types in the Events type manager in Splunk Web by going to Settings > Event types. You can also edit the eventtypes.conf file directly.

3. Create the appropriate tags in Splunk Web. Select Settings > Event types, locate the event type that you want to tag and click on its name. You can also edit the tags.conf file directly.

For more information about the Common Information Model and event tagging, see Configure CIM-compliant event tags.

Define event types in Splunk Web

An event type represents a search that returns a specific type of event or a useful collection of events. Every event that can be returned by that search gets an association with that event type. For example, say you have this search:

```plaintext
sourcetype=access_combined status=200 action=purchase
```

If you save that search as an event type named successful_purchase, any event that could be returned by that search gets eventtype=successful_purchase added to it at search time. This happens even if you are searching for something completely different.
And later, if you want to build a search that works with events that match that event type, include `eventtype=access_combined` in the search string.

A single event can match multiple event types. When an event matches two or more event types, `eventtype` acts as a multivalue field.

**Save a search you ran as an event type**

When you run a search, you can save that search as an event type. Event types usually represent searches that return a specific type of event, or that return a useful variety of events.

When you create an event type, the event type definition is added to `eventtypes.conf` in `$SPLUNK_HOME/etc/users/<your-username>/<app>/local/`, where `<app>` is your current app context. If you change the permissions on the event type to make it available to all users (either in the app, or globally to all apps), the Splunk platform moves the event type to `$SPLUNK_HOME/etc/apps/<App>/local/`.

**Prerequisites**

- See About event types for more information on event types.
- See About tags and aliases, for information about event type tagging.
- See About event type priorities, for more information about the Color and Priority fields.

**Saving a search as an event type**

1. In the Search view, run a search.
2. Click **Save As** and select **Event Type**.
3. Give the event type a unique **Name**.
4. (Optional) Add one or more comma-separated **Tag(s)**.
   
   You can apply the same tag to event types that produce similar results. A search that is just on that tag returns the set of events that collectively belong to those event types.

5. (Optional) Select a **Color**.
   
   This causes a band of color to appear at the start of the listing for any event that fits this event type. For example, this event matches an event
type that has a **Color** of **Purple**.

You can change the color of an event type (or remove its color entirely) by editing it in Settings.

6. (Optional) Give the event type a **Priority**.

**Priority** affects the display of events that match two or more event types. **1** is the best **Priority** and **10** is the worst. See About event type priorities.

7. Click **Save** to save the new event type.

You can access the list of event types that you and other users have created at **Settings > Event types**.

Any event type that you create with this method also appears on the Event Types listing page in Settings. You can update the event type in the Event Types listing page.

**Event Types page in Settings**

The Event Types page in Settings displays a list of the event types that you have permission to view or edit. You can use the Event Types page to create event types and maintain existing event types.

**Add an event type in Settings**

You can create a new event type through the Event Types page.

**Prerequisites**

- See About event types for more information on event types.
- See Save a search you ran as an event type. Most of the fields are discussed there.
• See About tags and aliases, for information about event type tagging.
• See About event type priorities, for more information about the **Color** and **Priority** fields.

**Adding a new event type in Settings**

1. Select **Settings > Event Types**.
2. Click **New**.

3. (Optional) Change the **Destination App** value to the correct app for the event type, if it is not your current app context.
4. Provide a unique **Name** for the event type.
5. Enter the **Search String** for the event type.

This search consistently returns a specific kind of event.

6. (Optional) Add one or more comma-separated **Tag(s)**.

You can apply the same tag to event types that produce similar results. A search that is just on that tag returns the set of events that collectively belong to those event types.

7. (Optional) Select a **Color**.

This causes a band of color to appear at the start of the listing for any event that fits this event type.
8. (Optional) Give the event type a **Priority**.

**Priority** affects the display of events that match two or more event types. 1 is the best **Priority** and 10 is the worst.

**Priority** determines the order of the event type listing in the expanded event. It also determines which color displays for the event type if two or more of the event types matching the event have a defined **Color** value. For more see About event type priorities.

9. Click **Save** to save the event type.

**Note:** All event types are initially created for a specific Splunk app. To make a particular event type available to all users on a global basis, you have to give all roles read or write access to the Splunk app and make it available to all Splunk apps. For more information about setting permissions for event types (and other knowledge object types), see Manage knowledge object permissions, in this manual.

**Update an event type in Settings**

**Prerequisites**

- See About event types for more information on event types.
- See Save a search you ran as an event type. Most of the fields are discussed there.
- See About tags and aliases, for information about event type tagging.
- See About event type priorities, for more information about the **Color** and **Priority** fields.

**Steps**

You can update the definition of any event type that you have created or which you have permissions to edit.

1. Navigate to **Settings > Event Types**.
2. Locate the event type that you would like to update in the Event Types listing page and click its name.
3. Update the **Search String**, **Tag(s)**, **Color**, and **Priority** of the event type as necessary.
4. Click **Save** to save your changes.

**About event type priorities**

You can give your event type a **Priority**. The **Priority** value that you select for an event type affects the display of events that match that event type, when those events also match other event types. For information on event types, see Define event types.

**Priority affects the event type listing order in expanded events.**

Event type matching takes place at search time. When you run a search and an event returned by that search matches an event type, Splunk software adds the corresponding `eventtype` field/value pair to it, where the value is the event type name.

You can see the event types that have been added to an event when you review your search results. Expand the event and check to see if the `eventtype` field is listed. If you see it, the event matches at least one event type.

If the event matches two or more event types, `eventtype` becomes a multivalued field whose values are ordered alphabetically, with the exception of event types that have a **Priority** setting. Event types with a **Priority** setting are listed above the event types without one, and they are ordered according to their **Priority** value.

If you have a number of overlapping event types, or event types that are subsets of larger ones, you may want to give the precisely focused event types a better priority. For example, you could easily have a set of events that are part of a wide-ranging `all_system_errors` event type. Within that large set of events, you could have events that also belong to more precisely focused event types like `critical_disc_error` and `bad_external_resource_error`.

Here is an example of an event that matches the `all_system_errors` and `critical_disc_error` event types.
In this example, the critical_disk_error event type has a priority of 3 while the all_system_errors event type has a priority of 7. 3 is a better priority value than 7, so critical_disk_error appears first in the list order.

**Priority determines which event type color displays for an event**

Only one event type color can be displayed for each event. When an event matches multiple event types, the Color for the event type with the best Priority value is displayed. However, for event types grouped with the transaction command, no color is displayed.

Following from the previous example, here is an example of two events with event type coloration.

Both events match the all_system_errors event type, which has a Color value of Orange. Events that have all_system_errors as the dominant event type display with orange event type coloration. One of the events also matches the critical_disk_error event type, which has a better Priority than all_system_errors. The critical_disk_error event type has Color set to Purple, so the event that matches it has purple event type coloration instead of orange.

**Automatically find and build event types**

The following utilities automatically locate and create event types to help you determine whether you have any potentially useful event types in your data:
• **Find event types:** The `findtypes` search command analyzes an event set and identifies patterns in your events that can be turned into useful event types.

• **Build event types:** The **Build Event Type utility** creates event types based on individual events. This utility also enables you to assign specific colors to event types. For example, if you say that a "sendmail error" event type is red, then the next time you run a search that returns events that fit that event type, they'll be easy to spot, because they'll show up as red in the event listing.

**Use the findtypes command to find event types in your search data**

To see the event types in the data that a search returns, add the `findtypes` command to the end of the search:

```bash
...| findtypes
```

Searches that use `findtypes` return a breakdown of the most common groups of events found in the search results. They are:

- ordered in terms of "coverage" (frequency). This helps you easily identify kinds of events that are subsets of larger event groupings.
- coupled with searches that can be used as the basis for event types that will help you locate similar events.

By default, `findtypes` returns the top 10 potential event types found in the sample, in terms of the number of events that match each kind of event discovered. You can increase this number by adding a `max` argument. For example, `findtypes max=30` returns the top 30 potential event types in an event sample.
The `findtypes` command also indicates whether or not the event groupings that it discovers match other event types.

**Note:** To return these results, the `findtypes` command analyzes up to 5000 events. For a more efficient--but potentially less accurate--search, you can lower this number using the `head` command:

```
...| head 1000 | findtypes
```

**Use the Build Event Type utility to create event types**

The **Build Event Type utility** or "Event Type Builder" leads you through the process of creating an event type that is based on an event in your search results.

1. Run a search that returns events that you want to base an event type on.
2. Identify an event in the results returned by the search that could be an event type and expand it.
3. Click Event Actions and select Build Event Type.

As you use the Build Event Type utility, you design a search that returns a specific set of results. This search string appears under **Generated event type** at the top of the utility interface.

The utility also displays a list of sample events. This list updates dynamically as you refine the event type search string.

4. In the **Event type features** sidebar, select field-value pairings that narrow down the event type search.

As you make selections the **Generated event type** search updates to include them. The list of sample events also updates to illustrate the events that match the event type that you are designing.
5. (Optional) At any time you can edit the event type search directly by clicking **Edit**.

6. (Optional) When you think your search might be a useful event type, test it by clicking **Test**.

   The search runs in a separate window.

7. When you have a search that returns the correct set of events, click **Save** to open the **Save event type** dialog.

8. Give the event type a **Name**.

9. (Optional) Give the event type a **Style**.

   **Style** is the same as **Color** in other event type definition workflows. This causes a band of color to appear at the start of the listing for any event that fits this event type. For example, this event matches an event type that has a **Style** of **Purple**.

   You can change the color of an event type (or remove its color entirely) by editing it in **Settings**.

10. (Optional) Give the event type a **Priority**.

    **Priority** affects the display of events that match two or more event types. **1** is the best **Priority** and **10** is the worst.

    **Priority** determines the order of the event type listing in the expanded event. It also determines which color displays for the event type if two or
more of the event types matching the event have a defined Color value.

See About event type priorities.

11. Click **Save** to save the event type.

**Configure event types in eventtypes.conf**

You can add new event types and update existing event types by configuring eventtypes.conf. There are a few default event types defined in

$SPLUNK_HOME/etc/system/default/eventtypes.conf. Any event types you create through Splunk Web are automatically added to

$SPLUNK_HOME/etc/system/local/eventtypes.conf.

**Important event type definition restrictions**

You cannot base an event type on a search that:

- Includes a **pipe operator** after a simple search.
- Includes a **subsearch**.
- Is defined by a simple search that uses the `savedsearch` command to reference a report name. For example, if you have a report named `failed_login_search`, you should not use this search to define the event type: `| savedsearch failed_login_search`. In this case you should instead use the search string that defines `failed_login_search` as the definition of the event type.

This last point is more of a best practice than a strict limitation. You want to avoid situations where the search string underneath `failed_login_search` is modified by another user at a future date, possibly in a way that breaks the event type. You have more control over the ongoing validity of the event type if you use actual search strings in its definition.

**Configure event types**

When you run a search, you can save that search as an event type. Event types usually represent searches that return a specific type of event, or that return a useful variety of events.
Prerequisites

Review

• About event types for more information on event types.
• About event type priorities for information on event type priorities.
• Event type syntax for information on the syntax for event type configuration.

Steps

1. Make changes to event types in `eventtypes.conf` in
   $SPLUNK_HOME/etc/system/local/ or your own custom app directory in
   $SPLUNK_HOME/etc/apps/. Use
   $SPLUNK_HOME/etc/system/README/eventtypes.conf.example as an
   example, or create your own `eventtypes.conf`.
2. (Optional) Configure a search term for this event type.
3. (Optional) Enter a human-readable description of the event type.
4. (Optional) Give the event type a priority.
5. (Optional) Give the event type a color.

Event type syntax

Use the following format when you define an event type in `eventtypes.conf`.

```
[$EVENTTYPE]
disabled = <1|0>
search = <string>
description = <string>
priority = <integer>
color = <string>
```

The $EVENTTYPE is the header and the name of your event type. You can have
any number of event types, each represented by a stanza and any number of the
following attribute-value pairs.

Note: If the name of the event type includes field names surrounded by the
percent character (for example, %$FIELD%) then the value of $FIELD is substituted
at search time into the event type name for that event. For example, an event
type with the header [cisco-%code%] that has code=432 becomes labeled
[cisco-432].

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disabled</td>
<td>Toggle event type on or off. Set to 1 to disable the event type.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>search</td>
<td>Search terms for this event type. For example, error OR warn.</td>
</tr>
<tr>
<td>description</td>
<td>Optional human-readable description of the event type.</td>
</tr>
<tr>
<td>priority</td>
<td>Specifies the order in which matching event types are displayed for an event. 1 is the highest, and 10 is the lowest.</td>
</tr>
</tbody>
</table>

**Note:** You can tag `eventtype` field values the same way you tag any other field-value combination. See the `tags.conf` spec file for more information.

**Example**

Here are two event types; one is called `web`, and the other is called `fatal`.

```
[web]
search = html OR http OR https OR css OR htm OR html OR shtml OR xls OR cgi

[fatal]
search = FATAL
```

**Disable event types**

Disable an event type by adding `disabled = 1` to the event type stanza in `eventtypes.conf`:

```
[$EVENTTYPE]
disabled = 1
```

$EVENTTYPE is the name of the event type you wish to disable.

So if you want to disable the `web` event type, add the following entry to its stanza:

```
[web]
disabled = 1
```
Configure event type templates

Event type templates create event types at search time. If you have Splunk Enterprise, you define event type templates in eventtypes.conf. Edit eventtypes.conf in $SPLUNK_HOME/etc/system/local/, or your own custom app directory in $SPLUNK_HOME/etc/apps/.

For more information on configuration files in general, see "About configuration files" in the Admin manual.

Event type template configuration

Event type templates use a field name surrounded by percent characters to create event types at search time where the %$FIELD% value is substituted into the name of the event type.

[$NAME-%$FIELD%] $SEARCH_QUERY

So if the search query in the template returns an event where %$FIELD%=bar, an event type titled $NAME-bar is created for that event.

Example

[cisco-%code%] search = cisco

If a search on "cisco" returns an event that has code=432, Splunk Enterprise creates an event type titled "cisco-432".
Transactions

About transactions

A transaction is a group of conceptually-related events that spans time. A transaction type is a transaction that has been configured in transactiontypes.conf and saved as a field.

Transactions can include:

- Different events from the same source and the same host.
- Different events from different sources from the same host.
- Similar events from different hosts and different sources.

For example, a customer purchase in an online store could generate a transaction that ties together events from several sources:

- A set of web access events share a session ID with....
- ....a corresponding event in the application server log, which also contains related account, product, and transaction IDs. The transaction ID in that application server event also appears in...
- ...a message queue event, which contains a message ID. This message ID is in turn shared by...
- ...a purchase fulfillment event logged by the fulfillment application, which also includes the shipping status of the item that the customer purchased.

All of the highlighted here, when grouped together, represent a single user transaction. If you were to define it as a transaction type you might call it an "item purchase" transaction. Other kinds of transactions include web access, application server downloads, emails, security violations, and system failures.

Transaction search

A transaction search enables you to identify transaction events that each stretch over multiple logged events. Use the transaction command and its options to define a search that returns transactions (groups of events). See the documentation of the command in the Search Reference for a variety of examples that show you how you can:
• Find groups of events where the first and last events are separated by a span of time that does not exceed a certain amount (set with the maxspan option).

• Find groups of events where the span of time between included events does not exceed a specific value (set with the maxpause option).

• Find groups of related events where the total number of events does not exceed a specific number (set with the maxevents option).

• Design a transaction that finds event groups where the final event contains a specific text string (set with the endswith option).

Study the transaction command topic to get the full list of available options for the command.

You can also use the transaction command to override transaction options that you have configured in transactiontypes.conf.

To learn more about searching with transaction, read “Identify and group events into transactions” in the Search Manual.

Configure transaction types

After you create a transaction search that you find worthy of repeated reuse, you can make it persistable by adding it to transactiontypes.conf as a transaction type.

To learn more about configuring transaction types, read "Configure transaction types," in this manual.

When to use stats instead of transactions

Transactions aren’t the most efficient method to compute aggregate statistics on transactional data. If you want to compute aggregate statistics over transactions that are defined by data in a single field, use the stats command.

For example, if you wanted to compute the statistics of the duration of a transaction defined by the field session_id:

* | stats min(_time) AS earliest max(_time) AS latest by session_id | eval duration=latest-earliest | stats min(duration) max(duration) avg(duration) median(duration) perc95(duration)

Similarly, if you wanted to compute the number of hits per clientip in an access log:
sourcetype=access_combined | stats count by clientip | sort -count

Also, if you wanted to compute the number of distinct session (parameterized by cookie) per clientip in an access log:

sourcetype=access_combined | stats dc(cookie) as sessions by clientip | sort -sessions

Read the stats command reference for more information about using the search command.

**Search for transactions**

Search for transactions using the transaction search command either in Splunk Web or at the CLI. The transaction command yields groupings of events which can be used in reports. To use transaction, either call a transaction type that you configured via transactiontypes.conf, or define transaction constraints in your search by setting the search options of the transaction command.

**Search options**

Transactions returned at search time consist of the raw text of each event, the shared event types, and the field values. Transactions also have additional data that is stored in the fields: duration and transactiontype.

- **duration** contains the duration of the transaction (the difference between the timestamps of the first and last events of the transaction).
- **transactiontype** is the name of the transaction (as defined in transactiontypes.conf by the transaction’s stanza name).

You can add transaction to any search. For best search performance, craft your search and then pipe it to the transaction command. For more information see the topic on the transaction command in the Search Reference manual.

Follow the transaction command with the following options.

**Note:** Some transaction options do not work in conjunction with others.

[field-list]

- This is a comma-separated list of fields, such as ...|transaction
  host,cookie
• If set, each event must have the same field(s) to be considered part of the same transaction.
• Events with common field names and different values will not be grouped.
  ♦ For example, if you add \[...|transaction host\], then a search result that has host=mylaptop can never be in the same transaction as a search result with host=myserver.
  ♦ A search result that has no host value can be in a transaction with a result that has host=mylaptop.

match=closest

• Specify the matching type to use with a transaction definition.
• The only value supported currently is closest.

maxspan=[<integer> s|m|h|d]

• Set the maximum span across events in a transaction.
• Can be in seconds, minutes, hours or days.
  ♦ For example: 5s, 6m, 12h or 30d.
• Defaults to maxspan=-1, for an "all time" timerange.

maxpause=[<integer> s|m|h|d]

• Specifies the maximum pause between transactions.
• Requires there be no pause between the events within the transaction greater than maxpause.
• If the value is negative, the maxpause constraint is disabled.
• Defaults to maxpause=-1.

startswith=<string>

• A search or eval-filtering expression which, if satisfied by an event, marks the beginning of a new transaction.
• For example:
  ♦ startswith="login"
  ♦ startswith=(username=foobar)
  ♦ startswith=eval(speed_field < max_speed_field)
  ♦ startswith=eval(speed_field < max_speed_field/12)
• Defaults to "."

endswith=<transam-filter-string>

• A search or eval-filtering expression which, if satisfied by an event, marks the end of a transaction.
• For example:
  ♦ endswith="logout"
  ♦ endswith=(username=foobar)
  ♦ endswith=eval(speed_field < max_speed_field)
  ♦ endswith=eval(speed_field < max_speed_field/12)

• Defaults to "."

For **startswith** and **endswith**, `<transam-filter-string>` is defined with the following syntax: 
```
"<search-expression>" | (<quoted-search-expression>) | eval(<eval-expression>)
```

• `<search-expression>` is a valid search expression that does not contain quotes.
• `<quoted-search-expression>` is a valid search expression that contains quotes.
• `<eval-expression>` is a valid eval expression that evaluates to a boolean.

Examples:

• search expression: (name="foo bar")
• search expression: "user=mildred"
• search expression: "search literal"
• eval bool expression: eval(distance/time < max_speed)

Transactions and macro search

Transactions and macro searches are a powerful combination that allows substitution into your transaction searches. Make a transaction search and then save it with `$field$` to allow substitution.

You can find an example of search macro and transaction combination in Search macro examples.

Example transaction search

Run a search that groups together all of the web pages a single user (or client IP address) looked at over a time range.

This search takes events from the access logs, and creates a transaction from events that share the same `clientip` value that occurred within 5 minutes of each other (within a 3 hour time span).

```
sourcetype=access_combined | transaction clientip maxpause=5m maxspan=3h
```
Configure transaction types

Any series of events can be turned into a transaction type. Read more about use cases in "About transactions", in this manual.

You can create transaction types via transactiontypes.conf. See below for configuration details.

For more information on configuration files in general, see "About configuration files" in the Admin manual.

Configure transaction types in transactiontypes.conf

1. Create a transactiontypes.conf file in $SPLUNK_HOME/etc/system/local/, or your own custom app directory in $SPLUNK_HOME/etc/apps/.
2. Define transactions by creating a stanza and listing specifications for each transaction within its stanza. Use the following attributes:

   [transactiontype]
   maxspan = [integer s|m|h|d|-1]
   maxpause = [integer s|m|h|d|-1]
   fields = <comma-separated list of fields>
   startswith = <transam-filter-string>
   endswith = <transam-filter-string>

   [TRANSACTIONTYPE]

   ♦ Create any number of transaction types, each represented by a stanza name and any number of the following attribute/value pairs.
   ♦ Use the stanza name, [TRANSACTIONTYPE], to search for the transaction in Splunk Web.
   ♦ If you do not specify an entry for each of the following attributes, Splunk Enterprise uses the default value.

   maxspan = [integer s|m|h|d|-1]

   ♦ Set the maximum time span for the transaction.
   ♦ Can be in seconds, minutes, hours or days, or set to -1 for unlimited.
       ♦ For example: 5s, 6m, 12h or 30d.
   ♦ Defaults to -1.
   maxpause = [integer s|m|h|d|-1]

   ♦ Set the maximum pause between the events in a transaction.
Can be in seconds, minutes, hours or days, or set to -1 for unlimited.
  ◊ For example: 5s, 6m, 12h or 30d.
  ◆ Defaults to -1.

maxevents = <integer>

◊ The maximum number of events in a transaction. This constraint is disabled if the value is a negative integer.
◆ Defaults to 1000.

fields = <comma-separated list of fields>

◊ If set, each event must have the same field(s) to be considered part of the same transaction.
◆ For example: fields = host, cookie
◆ Defaults to ".

connected= [true|false]

◆ Relevant only if fields is not empty. Controls whether an event that is not inconsistent and not consistent with the fields of a transaction opens a new transaction (connected=true) or is added to the transaction.
◆ An event can be not inconsistent and not consistent if it contains fields required by the transaction but none of these fields has been instantiated in the transaction (by a previous event addition).
◆ Defaults to: connected = true

startswith = <transam-filter-string>

◆ A search or eval filtering expression which, if satisfied by an event, marks the beginning of a new transaction
◆ For example:
  ◊ startswith="login"
  ◊ startswith=(username=foobar)
  ◊ startswith=eval(speed_field < max_speed_field)
  ◊ startswith=eval(speed_field < max_speed_field/12)
◆ Defaults to: " ".

endswith=<transam-filter-string>

◆ A search or eval filtering expression which if satisfied by an event marks the end of a transaction
◆ For example:
  ◊ endswith="logout"
  ◊ endswith=(username=foobar)
  ◊ endswith=eval(speed_field > max_speed_field)
  ◊ endswith=eval(speed_field > max_speed_field/12)
◆ Defaults to: " "

159
For both `startswith` and `endswith`, `<transam-filter-string>` has the following syntax:

"<search-expression>" | (<quoted-search-expression> | eval(<eval-expression>)

Where:

- `<search-expression>` is a valid search expression that does not contain quotes.
- `<quoted-search-expression>` is a valid search expression that contains quotes.
- `<eval-expression>` is a valid eval expression that evaluates to a boolean. For example, `startswith=eval(foo<bar*2)` will match events where `foo` is less than `2 x bar`.

Examples:

- "<search-expression>": `startswith="foo bar"
- `<quoted-search-expression>`: `startswith=(name="foo bar")`
- `<quoted-search-expression>`: `startswith=("search literal")`
- `eval(<eval-expression>)`: `eval(distance/time < max_speed)`

3. Use the transaction command in Splunk Web to call your defined transaction (by its transaction type name). You can override configuration specifics during search.

For more information about searching for transactions, see "Search for transactions" in this manual.

**Additional transaction configuration attributes**

`transactions.conf` includes a few more sets of attributes that are designed to handle situations such as multivalue fields and memory constraint issues.

**Transaction options for memory constraint issues**

`maxopentxn=<int>`

- Specifies the maximum number of not yet closed transactions to keep in the open pool before starting to evict transactions, using LRU (least-recently-used memory cache algorithm) policy.
- The default value of this attribute is read from the transactions stanza in `limits.conf`.  

160
maxopenevents=<int>

- Specifies the maximum number of events (which are) part of open transactions before transaction eviction starts happening, using LRU (least-recently-used memory cache algorithm) policy.
- The default value of this attribute is read from the transactions stanza in limits.conf.

keepevicted=[true|false]

- Whether to output evicted transactions. Evicted transactions can be distinguished from non-evicted transactions by checking the value of the evicted field, which is set to 1 for evicted transactions.
- Defaults to keepevicted=false.

Transaction options for rendering multivalue fields

mvlist=[true|false] | <field-list>

- The mvlist attribute controls whether the multivalue fields of the transaction are (1) a list of the original events ordered in arrival order or (2) a set of unique field values ordered lexicographically. If a comma- or space-delimited list of fields is provided, only those fields are rendered as lists.
- Defaults to: mvlist=false.

delim=<string>

- A string used to delimit the original event values in the transaction event fields.
- Defaults to: delim=" "

nullstr=<string>

- The string value to use when rendering missing field values as part of multivalue fields in a transaction.
- This option applies only to fields that are rendered as lists.
- Defaults to: nullstr=NULL
Use lookups in Splunk Web

About lookups

Lookups enrich your event data by adding field-value combinations from lookup tables. Splunk software uses lookups to match field-value combinations in your event data with field-value combinations in external lookup tables. If Splunk software finds those field-value combinations in your lookup table, Splunk software will append the corresponding field-value combinations from the table to the events in your search.

Types of lookups

There are four types of lookups:

- CSV lookups
- External lookups
- KV Store lookups
- Geospatial lookups

You can create lookups in Splunk Web through the Settings pages for lookups.

If you have Splunk Enterprise or Splunk Light and have access to the configuration files for your Splunk deployment, you can configure lookups by editing configuration files.

<table>
<thead>
<tr>
<th>Lookup type</th>
<th>Data source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV</td>
<td>A CSV file</td>
<td>Populates your events with fields pulled from CSV files. Also referred to as a static lookup because CSV files represent static tables of data. Each column in a CSV table is interpreted as the potential values of a field. Use CSV lookups</td>
</tr>
</tbody>
</table>

Create in Splunk Web | Configure in .conf files
Link | Link
<table>
<thead>
<tr>
<th>External</th>
<th>An external source, such as a DNS server.</th>
<th>CSV inline lookup table files and inline lookup definitions that use CSV files are both dataset types. See About datasets.</th>
<th>Link</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>KV Store</td>
<td>A KV Store collection</td>
<td>Uses Python scripts or binary executables to populate your events with field values from an external source. Also referred to as a scripted lookup.</td>
<td>Link</td>
<td>Link</td>
</tr>
<tr>
<td>Geospatial</td>
<td>A Keyhole Markup Zipped (KMZ) or Keyhole Markup Language (KML), used to define</td>
<td>A geospatial lookup matches location coordinates in your events to geographic feature collections in a KMZ or KML file and outputs fields to your events that</td>
<td>Link</td>
<td>Link</td>
</tr>
</tbody>
</table>
boundaries of mapped regions such as countries, US states, and US counties. 
provide corresponding geographic feature information encoded in the KMZ or KML, like country, state, or county names. Use a geospatial lookup to create a query that Splunk software uses to configure a choropleth map.

Not a dataset type.

Lookup table files

Lookup table files are files that contain a lookup table. A standard lookup pulls fields out of this table and adds them to your events when corresponding fields in the table are matched in your events.

All lookup types use lookup tables, but only two lookup types require that you upload a lookup table file: CSV lookups and geospatial lookups. A single lookup table file can be used by multiple lookup definitions.

For example, say you have a CSV lookup table file that provides the definitions of http_status fields. If you have events that include http_status = 503 you can have a lookup that finds the value of 503 in the lookup table column for the http_status field and pulls out the corresponding value for status_description in that lookup table. The lookup then adds status_description = Service Unavailable, Server Error to every event with http_status = 503.

Lookup definitions

A lookup definition provides a lookup name and a path to find the lookup table. Lookup definitions can include extra settings such as matching rules, or restrictions on the fields that the lookup is allowed to match. One lookup table can have multiple lookup definitions.

All lookup types require a lookup definition. After you create a lookup definition you can invoke the lookup in a search with the lookup command.
Automatic lookups

Use automatic lookups to apply a lookup to all searches at search time. After you define an automatic lookup for a lookup definition, you do not need to manually invoke it in searches with the lookup command.

See Define an automatic lookup.

Search commands and lookups

After you define your lookups and share them with apps, you can interact with them through search commands:

- **lookup**: Use to add fields to the events in the results of the search.
- **inputlookup**: Use to search the contents of a lookup table.
- **outputlookup**: Use to write fields in search results to a static lookup table file or KV store collection that you specify. You cannot use the outputlookup command with external lookups.

Lookups and the search-time operations sequence

**Search-time operation order**

Lookups are sixth in the search-time operations sequence and are processed after calculated fields but before event types.

**Restrictions**

The Splunk software processes lookups belonging to a specific host, source, or source type in ASCII sort order.

Lookup configurations can reference fields that are added to events by field extractions, field aliases, and calculated fields. They cannot reference event types and tags.

**For more information**

See The sequence of search-time operations.
Define a CSV lookup in Splunk Web

CSV lookups are file-based lookups that match field values from your events to field values in the static table represented by a CSV file. They output corresponding field values from the table to your events. They are also referred to as static lookups.

CSV lookups are best for small sets of data. The general workflow for creating a CSV lookup in Splunk Web is to upload a file, share the lookup table file, and then create the lookup definition from the lookup table file. CSV inline lookup table files, and inline lookup definitions that use CSV files, are both dataset types. See Dataset types and usage.

About the CSV files

There are some restrictions to the files that can be used for CSV lookups.

- The table in the CSV file should have at least two columns. One column represents a field with a set of values that includes values belonging to a field in your events. The column does not have to have the same name as the event field. Any column can have multiple instances of the same value, which is a multivalued field.
- The characters in the CSV file must be plain ASCII text and valid UTF-8 characters. Non-UTF-8 characters are not supported.
- CSV files cannot have "\r" line endings (OSX 9 or earlier)
- CSV files cannot have header rows that exceed 4096 characters.

Upload the lookup table file

To use a lookup table file, you must upload the file to your Splunk platform.

Prerequisites

- See lookup for an example of how to define a CSV lookup.
- An available .csv or .gz table file.

Steps

1. Select Settings > Lookups to go to the Lookups manager page.
2. In the Actions column, click Add new next to Lookup table files.
3. Select a Destination app from the list.
   - Your lookup table file is saved in the directory where the application
resides. For example:
$SPLUNK_HOME/etc/users/<username>/<app_name>/lookups/.

4. Click Choose File to look for the CSV file to upload.
5. Enter the destination filename. This is the name the lookup table file will have on the Splunk server. If you are uploading a gzipped CSV file, enter a filename ending in ".gz". If you are uploading a plaintext CSV file, use a filename ending in ".csv".
6. Click Save.

Share a lookup table file with apps

After you upload the lookup file, tell the Splunk software which applications can use this file. The default app is Launcher.

1. Select Settings > Lookups.
2. From the Lookup manager, click Lookup table files.
3. Click Permissions in the Sharing column of the lookup you want to share.
4. In the Permissions dialog box, under Object should appear in, select All apps to share globally. If you want the lookup to be specific to this app only, select This app only. You can also keep your look private by selecting Keep private.
5. Click Save.

Create a CSV lookup definition

You must create a lookup definition from the lookup table file.

Prerequisites
In order to create the lookup definition, share the lookup table file so that Splunk software can see it.

Review

- About lookups and field actions.
- Configure a time-bounded lookup.
- Make your lookup automatic.

Steps

1. Select Settings > Lookups.
2. Click Lookup definitions.
3. Click New.
4. Select a **Destination app** from the drop-down list.  
   Your lookup table file is saved in the directory where the application resides. For example  
   `$SPLUNK_HOME/etc/users/<username>/<app_name>/lookups/`.  
5. Give your lookup definition a unique **Name**.  
6. Select **File-based** as the lookup **Type**.  
7. Select the **Lookup file** from the drop-down list. For a CSV lookup, the file extension must be `.csv`.  
8. (Optional) If the CSV file contains time fields, make the CSV lookup time-bounded by selecting the **Configure time-based lookup** check box.   

<table>
<thead>
<tr>
<th>Time-based options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name of time field</td>
<td>The minimum number of matches for each input lookup value. This defaults to 0.</td>
</tr>
<tr>
<td>Time format</td>
<td>Enter a number from 1-1000 to specify the maximum number of matches for each lookup value. If time-based, the default is 1; otherwise, the default is 1000.</td>
</tr>
<tr>
<td>Minimum offset</td>
<td>When fewer than the minimum number of matches are present for any given input, the Splunk software provides this value one or more times until the minimum is reached.</td>
</tr>
<tr>
<td>Maximum offset</td>
<td>If the check box is selected, case-sensitive matching will be performed for all fields in a lookup table. The default value is true.</td>
</tr>
</tbody>
</table>

9. (Optional) To define advanced options for your lookup, select the **Advanced options** check box.   

<table>
<thead>
<tr>
<th>Advanced options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum matches</td>
<td>The minimum number of matches for each input lookup value. Defaults to 0.</td>
</tr>
<tr>
<td>Maximum matches</td>
<td>Enter a number from 1-1000 to specify the maximum number of matches for each lookup value. If time-based, the default value is 1; otherwise, the default value is 1000.</td>
</tr>
<tr>
<td>Default matches</td>
<td>When fewer than the minimum number of matches are present for any given input, the Splunk software provides this value one or more times until the minimum is reached.</td>
</tr>
<tr>
<td>Case sensitive match</td>
<td>If the check box is selected, case-sensitive matching will be performed for all fields in a lookup table. The default value is true.</td>
</tr>
</tbody>
</table>
Replicate delta

Select the check box to turn on delta replication. The search head replicates only the changes to the lookup table, rather than replicating the entire lookup table. The delta file is sent to the indexers and, if you are using a search head cluster, to the other search heads in the search head cluster. For more information on when to use delta replication, see Handle large CSV lookup tables.

Batch index query

Select this check box if you are using a large lookup file that may affect performance.

Match type

A comma and space-delimited list of <match_type>(<field_name>) specification to allow for non-exact matching. The available match_type values are WILDCARD, CIDR, and EXACT. EXACT is the default. Specify the fields that use WILDCARD or CIDR in this list.

Filter lookup

Filter results from the lookup table before returning data. Create this filter like you would a typical search query using Boolean expressions and/or comparison operators.

For CSV lookups, filtering is done in memory.

10. Click Save.

Your lookup is defined as a file-based CSV lookup and appears in the list of lookup definitions.

Share the lookup definition with apps

After you create the lookup definition, specify in which apps you want to use the definition.

1. Select Settings > Lookups.
2. Click Lookup definitions.
3. In the Lookup definitions list, click Permissions in the Sharing column of the lookup definition you want to share.
4. In the Permissions dialog box, under Object should appear in, select All apps to share globally. If you want the lookup to be specific to this app only, select This app only. You can also keep your lookup private by selecting Keep private.
5. Click Save.

Permissions for lookup table files must be at the same level or higher than those of the lookup definitions that use those files.
You can use this field lookup to add information from the lookup table file to your events. You can use the field lookup with the `lookup` command in a search string. Or, you can set the field lookup to run automatically. For information on creating an automatic lookup, see Create a new lookup to run automatically.

**Handle large CSV lookup tables**

Lookup tables are created and modified on a **search head**. The search head replicates a new or modified lookup table to other search heads, or to **indexers** to perform certain tasks.

- **Knowledge bundle replication**. When a search head distributes searches to indexers, it also distributes a related **knowledge bundle** to the indexers. The knowledge bundle contains knowledge objects, such as lookup tables, that the indexers need to perform their searches. See What search heads send to search peers in Distributed Search.

- **Configuration replication (search head clusters)**. In **search head clusters**, runtime changes made on one search head are automatically replicated to all other search heads in the cluster. If a user creates or updates a lookup table on a search head in a cluster, that search head then replicates the updated table to the other search heads. See Configuration updates that the cluster replicates in Distributed Search.

When a lookup table changes, the search head must replicate the updated version of the lookup table to the other search heads or the indexers, or both, depending on the situation. By default, the search head sends the entire table each time any part of the table changes.

**Make the lookup automatic**

Instead of using the lookup command in your search when you want to apply a field lookup to your events, you can set the lookup to run automatically. See Define an automatic lookup for more information.

**Configure a CSV lookup with .conf files**

CSV lookups can also be configured using `.conf` files. See Configure CSV lookups.
Define an external lookup in Splunk Web

External lookups use python scripts or binary executables to populate events with field values from an external source.

External lookups are often referred to as scripted lookups, because they are facilitated through the use of a script. See About the external lookup script.

Create an external lookup

If you have Splunk Cloud and want to define external lookups, file a Support ticket in order to add a script, or use an existing Splunk software script.

Prerequisites

- You must be an admin user with .conf and file directory access to upload a script for the lookup.

Review

- About lookups
- About the external lookup script
- Configure a time-bounded lookup
- Make your lookup automatic

Steps

1. Add the script for the lookup to your Splunk deployment. The script must be located in either one of two places:
   - $SPLUNK_HOME/etc/searchscripts
   - $SPLUNK_HOME/etc/apps/<app_name>/bin
2. Select Settings > Lookups.
3. Select Lookup definitions.
4. Click New.
5. Change the Type to External.
6. Select the destination app.
7. Type a unique Name for your external lookup.
8. Type the command and arguments for the lookup. The command must be the name of the script, for example external_lookup.py. The arguments are the names of the fields that you want to pass to the script, separated by spaces, for example: clienthost clientip.
9. List all of the fields that are supported by the external lookup. The fields must be delimited by a comma followed by a space.
10. (Optional) Make this lookup a time-based lookup.

<table>
<thead>
<tr>
<th>Time-based options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name of time field</td>
<td>The name of the field in the lookup table that represents the timestamp. This defaults to 0.</td>
</tr>
<tr>
<td>Time format</td>
<td>The strftime format of the timestamp field. You can include subseconds but the Splunk platform will ignore them. This defaults to %s.%Q or seconds from unix epoch in UTC and optional milliseconds.</td>
</tr>
<tr>
<td>Minimum offset</td>
<td>The minimum time (in seconds) that the event timestamp can be later than the lookup entry timestamp for a match to occur. This defaults to 0.</td>
</tr>
<tr>
<td>Maximum offset</td>
<td>The maximum time (in seconds) that the event timestamp can be later than the lookup entry time for a match to occur. This defaults to 2000000000.</td>
</tr>
</tbody>
</table>

11. (Optional) To define advanced options for your lookup, select the **Advanced options** check box.

<table>
<thead>
<tr>
<th>Advanced options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum matches</td>
<td>The minimum number of matches for each input lookup value. The default value is 0.</td>
</tr>
<tr>
<td>Maximum matches</td>
<td>Enter a number from 1-1000 to specify the maximum number of matches for each lookup value. If time-based, the default value is 1; otherwise, the default value is 1000.</td>
</tr>
<tr>
<td>Default matches</td>
<td>When fewer than the minimum number of matches are present for any given input, the Splunk software provides this value one or more times until the minimum is reached.</td>
</tr>
<tr>
<td>Case sensitive match</td>
<td>If the check box is unselected, case-insensitive matching will be performed for all fields in a lookup table. Defaults to true.</td>
</tr>
<tr>
<td>Allow caching</td>
<td>Allows output from lookup scripts to be cached. The default value is true.</td>
</tr>
<tr>
<td>Match type</td>
<td>A comma and space-delimited list of &lt;match_type&gt;(&lt;field_name&gt;) specification to allow for non-exact matching. The available match_type values are WILDCARD, CIDR, and EXACT. EXACT is the default. Specify the fields that use WILDCARD or CIDR in this list.</td>
</tr>
</tbody>
</table>
Filter results from the lookup table before returning data. Create this filter like you would a typical search query using Boolean expressions and/or comparison operators.

For CSV lookups, filtering is done in memory.

12. Click **Save**.

Your lookup is now defined as an external lookup and will show up in the list of lookup definitions.

**Share the lookup definition**

Now that you have created the lookup definition, you need to specify in which apps you want to use the definition.

1. In the Lookup definitions list, for the lookup definition you created, click **Permissions**.
2. In the Permissions dialog box, under **Object should appear in**, select **All apps** to share globally. If you want the lookup to be specific to this app only, select **This app only**. You can also keep your lookup private by selecting **Keep private**.
3. Click **Save**.

In the Lookup definitions page, your lookup now has the permissions you have set.

Permissions for lookup table files must be at the same level or higher than those of the lookup definitions that use those files.

**External lookup example**

The following is an example of an external lookup that is delivered with Splunk software. It matches with information from a DNS server. It is not an automatic lookup. You can access it by running a search with the `lookup` command.

Splunk Enterprise ships with a script located in `$SPLUNK_HOME/etc/system/bin/ called `external_lookup.py`, which is a DNS lookup script that:

- if given a host, returns the IP address.
- if given an IP address, returns the host name.

In the following section, you will use the default script `external_lookup.py` to create a lookup.
Define the external lookup

1. Select Settings > Lookups.
2. Select Lookup definitions.
3. Click New.
4. Type the lookup name dnslookup in the Name field.
5. Change the Type to External.
6. For the Command, enter the python script name external_lookup.py and the arguments clienthost and clientip as shown below.
   
   ```python
   external_lookup.py clienthost clientip
   ```
7. For the Supported fields, enter clienthost, clientip
8. Click Save.

Share the external lookup

1. In the lookup definitions list, click Permissions.
2. Select All apps for the lookup definition to be shared globally.
3. Click Save.

You can now run a search with the lookup command that uses the dnslookup lookup definition that you created.

```
sourcetype=access_combined | lookup dnslookup clienthost AS host | stats count by clientip
```

This search:

- Matches the clienthost field in the external lookup table with the host field in your events.
- Returns a table that provides a count for each of the clientip values that corresponds with the clienthost matches.

This search does not add fields to your events.

You can also design a search that performs a reverse lookup, which in this case returns a host value for each IP address it receives.

```
sourcetype=access_combined | lookup dnslookup clientip | stats count by clienthost
```

This reverse lookup search does not include an AS clause. This is because Splunk automatically extracts IP addresses as clientip.
About the external lookup script

Your external lookup script must take in an incomplete CSV file and output a complete CSV file. The arguments that you pass to the script are the headers for these input and output files.

In the DNS lookup example, the CSV file contains the two fields clienthost and clientip. The fields that you pass to this script are specified in the lookup definition that you have created. If you do not pass these arguments, the script returns an error:

When you run this search string:

... | lookup dnsLookup clienthost

You are telling Splunk software to:

1. Use the lookup table that you defined in Splunk Web as dnslookup.
2. Pass the values for the clienthost field into the external command script as a CSV file. The CSV file appears as follows:

clienthost,clientip
work.com
home.net

This is a CSV file with clienthost and clientip as column headers, but without values for clientip. The script includes the two headers because they are the fields you specified in the fields_list attribute of the [dnslookup] stanza in the default transforms.conf.

The script outputs the following CSV file, which is used to populate the clientip field in your results:

host,ip
work.com,127.0.0.1
home.net,127.0.0.2
Note: When writing your script, if you refer to any external files, the reference must be relative to the directory where the script is located.

See also

In addition to using external lookups to add fields from external sources to events, you can use a scripted input to send data from non-standard sources for indexing or to prepare this data for parsing. For more information, see the Scripted inputs overview in Developing Views and Apps for Splunk Web.

Make the lookup automatic

Instead of using the lookup command in your search when you want to apply a field lookup to your events, you can set the lookup to run automatically. See Define an automatic lookup for more information.

Configure external lookups with .conf files

External lookups can also be configured using .conf files. See Configure external lookups for more information.

Define a KV Store lookup in Splunk Web

KV Store lookups populate your events with fields pulled from your App Key Value Store (KV Store) collections. KV Store lookups can be invoked through REST endpoints or by using the search commands lookup, inputlookup, and outputlookup. Use a KV Store lookup when you have a large lookup table or a table that is updated often.

KV Store vs. CSV files

The KV Store adds a lookup type to use with your apps named KV Store lookups. Before the KV Store feature was added, you might have used CSV-based lookups to augment data within your apps. Consider the following tradeoffs when deciding whether a KV Store lookup or a CSV-based lookup is best for your scenario:

<table>
<thead>
<tr>
<th>Lookup type</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| KV Store lookup | • Enables per-record insert and updates.  
• Allows optional data type enforcement on write operations.  
• Allows you to define field accelerations to improve search performance.  
• Provides REST API access to the data collection. | Does not support case-insensitive field lookups. |
|----------------|-------------------------------------------------------------------------------------------------|-------------------------------------------------|
| CSV lookup     | • Performs well for files that are small or rarely modified.  
• CSV files are easier to modify manually.  
• Integrating with other applications such as Microsoft Excel is easier because CSV is a standard format.  
• Supports case-sensitive field lookups. | • Does not provide multiuser access locking.  
• Requires a full rewrite of the file for edit operations (outputlookup).  
• Does not support REST API access. |

**KV Store collections**

Before you create a KV Store lookup, your Splunk deployment must have at least one KV Store collection defined in `collections.conf`. See Use configuration files to create a KV Store collection on the Splunk Developer Portal.

Certain apps, such as Enterprise Security, also include KV Store collections with their installation. If you have Splunk Cloud and want to define KV Store lookups, use one of the default KV Store collections or file a Support ticket to add a unique KV Store collection.

KV Store collections are databases. They store your data as key/value pairs. When you create a KV Store lookup, the collection should have at least two fields. One of those fields should have a set of values that match with the values of a field in your event data, so that lookup matching can take place.

When you invoke the lookup in a search with the `lookup` command, you designate a field in your search data to match with the field in your KV Store collection. When a value of this field in an event matches a value of the designated field in your KV Store collection, the corresponding value(s) for the other field(s) in your KV Store collection can be added to that event.
The KV Store field does not have to have the same name as the field in your events. Each KV Store field can be **multivalued**.

KV Store collections live on the search head, while CSV files are replicated to indexers. If your lookup data changes frequently you may find that KV Store lookups offer better performance than an equivalent CSV lookup.

**Define a KV Store lookup**

**Prerequisites**

- You must be an admin user with .conf and file directory access to create a KV Store collection. If you have Splunk Cloud and want to define KV Store lookups, file a Support ticket in order to add a collection.

**Review**

- About lookups
- Use configuration files to create a KV Store collection store
- Configure a time-bounded lookup
- Make your lookup automatic

**Steps**

1. Select **Settings > Lookups**.
2. Click **Lookup definitions**.
3. Click **Add new**.
4. Change the **Type** to **KV Store**.
5. Enter the collection name to use as defined in **collections.conf**.
6. List all of the fields that are supported by the KV Store lookup. The fields must be delimited by a comma followed by a space. A field can be any combination of key and value that you have in your KV store collection.
7. (Optional) Configure time-based lookup.

<table>
<thead>
<tr>
<th>Time-based options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name of time field</td>
<td>The minimum number of matches for each input lookup value. The default is 0.</td>
</tr>
<tr>
<td>Time format</td>
<td>Enter a number from 1-1000 to specify the maximum number of matches for each lookup value. If time-based, the default is 1; otherwise, the default is 1000.</td>
</tr>
<tr>
<td>Minimum offset</td>
<td></td>
</tr>
</tbody>
</table>
When fewer than the minimum number of matches are present for an given input, the Splunk software provides this value one or more times until the minimum is reached.

<table>
<thead>
<tr>
<th>Maximum offset</th>
<th>If the check box is selected, case-sensitive matching is performed for all fields in a lookup table. The default value is true.</th>
</tr>
</thead>
</table>

8. (Optional) To define advanced options for your lookup, select the **Advanced options** check box.

<table>
<thead>
<tr>
<th>Advanced options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum matches</td>
<td>The minimum number of matches for each input lookup value. The default value is 0.</td>
</tr>
<tr>
<td>Maximum matches</td>
<td>Enter a number from 1-1000 to specify the maximum number of matches for each lookup value. If time-based, the default value is 1; otherwise, the default value is 1000.</td>
</tr>
<tr>
<td>Default matches</td>
<td>When fewer than the minimum number of matches are present for an input, the Splunk software provides this value one or more times until the minimum is reached.</td>
</tr>
<tr>
<td>Maximum external batch</td>
<td>The maximum size of the external batch. The range is 1 to 1000. The default is 300. Do not change this value unless you know what you are doing.</td>
</tr>
<tr>
<td>Match type</td>
<td>Optionally set up non-exact matching of a comma-and-space-delimited field list. The format is &lt;match_type&gt;(&lt;field_name1&gt;&lt;field_name2&gt;,...&lt;field_nameN&gt;). Available values for match_type are WILDCARD and CIDR.</td>
</tr>
<tr>
<td>Filter lookup</td>
<td>Filter results from the lookup table before returning data. Create this filter as a search query with Boolean expressions and comparison operators.</td>
</tr>
</tbody>
</table>

9. Click **Save**.

Your lookup is now defined as a KV Store lookup and will show up in the list of Lookup definitions.

**Share the lookup definition**

Now that you have created a KV store lookup definition, you need share the definition with other users. You can share it with users of a specific app, or you can share it globally to users of all apps.
1. In the Lookup definitions list, for the lookup definition you created, click Permissions.

2. In the Permissions dialog box, under Object should appear in, select All apps to share globally or the app that you want to share it with.

3. Click Save.
   In the Lookup definitions page, your lookup now has the permissions you have set.

Permissions for lookup table files must be at the same level or higher than those of the lookup definitions that use those files.

**Make the lookup automatic**

Instead of using the lookup command in your search when you want to apply a KV store lookup to your events, you can set the lookup to run automatically. When your lookup is automatic, the Splunk software applies it to all searches at search time.

See Define an automatic lookup in Splunk Web for more information.

**Prefilter large KV Store collections**

When your KV Store collection is extremely large, performance can suffer when your lookups must search through the entire collection to retrieve matching field values. If you know that you only need results from a subset of records in the lookup table, improve search performance by using the filter attribute to filter out all of the records that do not need to be looked at.

The filter attribute requires a string containing a search query with Boolean expressions and/or comparison operators (==, !=, >, <, <=, >=, OR, AND, and NOT). This query runs whenever you run a search that invokes this lookup.

For example, if your lookup configuration has filter = (CustID>500) AND (CustName="P*"), it tries to retrieve values only from those records in the KV Store collection that have a CustID value that greater than 500 and a CustName value that begins with the letter P.

If you do not want to install a filter in the lookup definition you can get a similar effect when you use the where clause in conjunction with the inputlookup command.
Configure KV Store lookups with .conf files

KV Store lookups can also be configured using .conf files. See Configure KV store lookups for more information.

For developer-focused KV Store lookup configuration instructions, see Use lookups with KV Store data in the Splunk Developer Portal.

Define a geospatial lookup in Splunk Web

Use geospatial lookups to create queries that return results that Splunk software can use to generate a choropleth map visualization. Choropleth maps cannot be rendered without the data generated by corresponding geospatial lookups.

A geospatial lookup matches location coordinates in your events to location coordinate ranges in a geographic feature collection known as a Keyhole Markup Zipped (KMZ) or Keyhole Markup Language (KML) file and outputs fields to your events that provide corresponding geographic feature information that is encoded in the feature collection. This information represents a geographic region that shares borders with geographic regions of the same type, such as a country, state, province, or county.

Splunk provides two geospatial lookups; one for the United States and one for world countries, enabling you to render choropleth maps:

- The USA, divided into states
- The world, divided into countries

This topic shows you how to create additional geospatial lookups that break up choropleth maps into other types of regions, such as counties, provinces, timezones, and so on.

For information about choropleth maps and geographic data visualizations, see Mapping data in the Dashboards and Visualizations manual.

The workflow to create a geospatial lookup in Splunk Web is to upload a file, share the lookup table file, and then create the lookup definition from the lookup table file.
The FeatureId and featureCollection fields

Geospatial lookups differ from other lookup types in that they are designed to output these two fields: featureId and featureCollection. The featureId is the name of the feature, such as California or CA or whatever name is encoded in the feature collection. The featureCollection field provides the name of the lookup in which the feature was found.

If you pipe the output of a geospatial lookup into a geom command, the command does not need to be given the lookup name. The geom command detects the featureId and featureCollection fields in the event and uses the lookup to generate the geographic data structures that the Splunk software requires to generate a choropleth map. However, geographic data structures can be large. It is strongly discouraged to pipe events into the geom command, because geographic data structures are attached to every event. Instead, first perform stats on the results of your geographic lookup, and only perform geom on an aggregated statistic like count by featureId.

The Feature Id Element field

The Feature Id Element field is an XPath expression that defines a path from a Polygon element in the KML file to some other XML element that contains the name of the feature. Splunk software calls these Polygon elements a "feature". This is needed in cases where the typical style of named Placemark element is not in use.

- The Feature Id Element field may be required in cases where the featureId field generated by the lookup is an empty string, or when the feature collection returns incorrect features by default. In the latter case, the feature may be a peer of the default feature or is located relative to the default feature.
- To determine what path you need, review the geographic feature collection. Each feature in the collection is tagged with <Placemark>, and each <Placemark> contains a name that the lookup writes out as featureId fields. For an example, see feature_id_element.
- The default setting for Feature Id Element is /Placemark/name.

XPath and feature id element example

The following is an example <Placemark> element extracted from a KML file.

<Placemark>
The `<Placemark>` element contains both a `<name>` element and a `<Polygon>` element. A `<Placemark>` can have multiple `<Polygons>`. `<Placemark>` associates a name to a set of `<Polygons>`, called a "feature." However, different KML files may organize their data differently, so we need to tell Splunk software where to find the name, relative to the `<Placemark>` element. We can do this with the Feature Id Element field. By default, Feature Id Element contains the XPath...
expression /Placemark/name.

Let's take a look at another <Placemark> element extracted from a KML file.

<Placemark>
  <name>MyFeature</name>
  <ExtendedData>
    <SchemaData>
      <SimpleData name="placename">foo</SimpleData>
      <SimpleData name="bar">baz</SimpleData>
    </SchemaData>
  </ExtendedData>
</Placemark>

... 

The XPath expression for this Placemark fragment would be

feature_id_element=/Placemark/ExtendedData/SchemaData/SimpleData[@name='placename']

### Upload the lookup table file

To use a lookup table file, you must upload the file to your Splunk platform.

#### Prerequisites

- About lookups and field actions for information on lookups.
- An available .kmz or .kml table file.

#### Steps

1. Select **Settings > Lookups**.
2. In the Actions column, click **Add new** next to **Lookup table files**.
3. Select a **Destination app** from the list. Your lookup table file is saved in the directory where the application resides. For example:
   
   $SPLUNK_HOME/etc/users/<username>/<app_name>/lookups/

4. Click **Choose File**.
5. Enter the destination file name. This is the name the lookup table file will have on the Splunk server. Use a file name ending in ".kmz" or "kml".
6. Click **Save**.

### Share the lookup table file

After you upload the lookup file, tell the Splunk software which applications can use this file. The default app is Launcher.

1. Select **Settings > Lookups**.
2. From the Lookup manager, click on **Lookup table files**.
3. Click **Permissions** under Sharing for the file you want to share.
4. If you want the lookup to be available globally, select **Global**. If you want the lookup to be specific to this app only, select **This app only**.
5. Click **Save**.

### Create the lookup definition

You must create a **lookup definition** from the lookup table file.

#### Prerequisites

In order to create the lookup definition, share the lookup table file so that Splunk software can see it.

#### Review

- About lookups and field actions.
- Configure a time-bounded lookup.
- Make your lookup automatic.
- The feature_id_element field.

#### Steps

1. Select **Settings > Lookups**.
2. Click **Lookup definitions**.
3. Click **New** to add a new definition.
4. Select a **Destination app** from the list. Your lookup table file is saved in the directory where the application resides. For example: `$SPLUNK_HOME/etc/users/<username>/<app_name>/lookups/`.
5. Give your lookup definition a unique **Name**.
6. Select **Geospatial** for the lookup **Type**.
7. Select the **Lookup file**.
8. (Optional) To define advanced options for your lookup, select the **Advanced options** check box.

<table>
<thead>
<tr>
<th>Advanced options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Id Element</td>
<td>An XPath expression that defines a path from a Polygon element in the KML file to another XML element or attribute that contains the name of the feature. Required when named Placemark elements are not in use.</td>
</tr>
</tbody>
</table>

9. Click **Save**.
Your lookup is defined as a geospatial lookup and appears in the list of Lookup definitions.

**Share the lookup definition**

Now that you have created the lookup definition, you need to specify in which apps you want to use the definition.

1. In the Lookup definitions list, for the lookup definition you created, click Permissions.
2. In the Permissions dialog box, under Object should appear in, select All apps to share globally or the app that you want to share it with.
3. Click Save.

In the Lookup definitions page, your lookup now has the permissions you have set.

Permissions for lookup table files must be the same or larger than those of the lookup definitions that use those files.

You can use this field lookup to add information from the lookup table file to your events. You can use the field lookup by specifying the lookup command in a search string. Or, you can set the field lookup to run automatically.

**Make the lookup automatic**

Instead of using the lookup command in your search when you want to apply a field lookup to your events, you can set the lookup to run automatically. See Define an automatic lookup for more information.

**Search commands and geospatial lookups**

If you have created a geospatial lookup definition, you can interact with geospatial lookup through the inputlookup search command. You can use inputlookup to show all geographic features on a choropleth map visualization.

This example uses the default geo_us_states lookup.

**Prerequisites**

- A geospatial lookup, see above.

**Steps:**
1. From the Search and Reporting app, use the `inputlookup` command to search on the contents of your geospatial lookup.

   ```
   | inputlookup geo_us_states
   ```

2. Click on the Visualization tab.
3. Click on Cluster Map and select Chloropleth Map for your visualization.

A chloropleth map displaying the featureIds of your geospatial lookup appears. For more information on chloropleth maps, see Generate a chloropleth map in the Dashboard and Visualizations manual.

**Configure geospatial lookups with .conf files**

Geospatial lookups can also be configured using .conf files. See Configure geospatial store lookups for more information.

**Define a time-based lookup in Splunk Web**

If your lookup table has a field that represents time, you can use it to create a time-bounded lookup; which is also referred to as a temporal lookup. You can define CSV lookups, external lookups, and KV Store lookups as time-based lookups, but you cannot define a geospatial lookup as a time-based lookup.

**Prerequisites**

Review the following topics:

- Lookups and the search-time operations sequence for field lookup restrictions
- Define a CSV lookup in Splunk Web
- Define an external lookup in Splunk Web
- Define a KV Store lookup in Splunk Web

**Create a time-based lookup**

1. Select **Settings > Lookups**.
2. Click **Lookup definitions**.
3. Click the lookup that you want to define as a time-based lookup.
4. Click the **Configure time-based lookup** checkbox.
5. Enter the name of the field in the lookup table that represents the timestamp.
6. Enter the time format of the timestamp field. The default format is UTC
time.
7. Enter the minimum time in seconds that the event time can be ahead of
   the lookup entry time for a match to occur. The default is 0.
8. Enter the maximum time in seconds that the event time can be ahead of
   lookup entry time for a match to occur. The default is 2000000000.
9. Click Save.

The Lookup definition page appears, and the lookup that you defined is listed.

**Define an automatic lookup in Splunk Web**

Manual lookups are applied to the results of a search when they are invoked
with the lookup command. Automatic lookups are applied to all searches at
search time.

Splunk software does not support nested automatic lookups.

**Add a new lookup to run automatically**

**Prerequisites**
Review the following topics:

- Lookups and the search-time operations sequence for field lookup
  restrictions
- Define a CSV lookup in Splunk Web
- Define an external lookup in Splunk Web
- Define a KV Store lookup in Splunk Web
- Define a geospatial lookup in Splunk Web
- An example lookup in Splunk Web

A lookup definition that you have defined previously.

**Steps**

1. In Splunk Web, select **Settings > Lookups**.
2. Under Actions for Automatic Lookups, click **Add new**.
3. Select the **Destination app**.
4. Give your automatic lookup a unique **Name**.
5. Select the **Lookup table** that you want to use in your fields lookup.
   This is the name of the lookup definition that you defined on the
   Lookup Definition page.
6. In the **Apply to** menu, select a host, source, or source type value to apply the lookup and give it a name in the **named** field.

7. Under **Lookup input fields** provide one or more pairs of input fields. The first field is the field in the lookup table that you want to match. The second field is a field from your events that matches the lookup table field. For example, you can have an `ip_address` field in your events that matches an `ip` field in the lookup table. So you would enter `ip = ip_address` in the automatic lookup definition.

8. Under **Lookup output fields** provide one or more pairs of output fields. The first field is the corresponding field that you want to output to events. The second field is the name that the output field should have in your events. For example, the lookup table may have a field named `country` that you may want to output to your events as `ip_city`. So you would enter `country=ip_city` in the automatic lookup definition.

9. You can select the checkbox for **Overwrite field values** to overwrite the field values when the lookup runs.

   **Note:** This is equivalent to configuring your fields lookup in `props.conf`.

10. Click **Save**.

    The Automatic lookup view appears, and the lookup that you have defined is listed.

### Lookup example in Splunk Web

This example defines a file-based CSV lookup that adds two fields, `status_description` and `status_type`, to your web access events. This lets you search for events when you do not know the specific error code. Instead of searching for all server error codes, use `status="Server Error"`.

**Upload the lookup table to Splunk Enterprise**

**Prerequisites**

- See Define a CSV lookup in Splunk Web.
- Download the `http_status.csv` file: `http_status.csv`

The following is a sample of the file:

```
status,status_description,status_type
100,Continue,Informational
```
Steps

1. From the Search app, then select **Settings > Lookups.**
2. Select **Add new** for **Lookup table files.**
3. Select **search** for the destination app.
4. Browse for the CSV file that you downloaded earlier.
5. Name the lookup table **http_status.**
6. Click **Save.**

After Splunk Enterprise saves the file, it takes you to the following view:

---

**Define the lookup**

**Prerequisites**

- See **Define a CSV lookup in Splunk Web.**

**Steps**

1. From **Settings > Lookups**, select **Add new** for **Lookup definitions.**
2. Select search for the Destination app.
3. Name your lookup definition http_status.
4. Select File-based under Type.
5. Click Save.

Notice there are some actions you can take on your lookup definition. Permissions lets you change the accessibility of the lookup table. You can Disable, Clone, and Move the lookup definition to a different app. Or, you can Delete the definition. Once you define the lookup, you can use the lookup command to invoke it in a search or you can configure the lookup to run automatically.

Set the lookup to run automatically

Prerequisites

• See Define an automatic lookup in Splunk Web

Steps

1. Return to the Settings > Lookups view and select Add new for Automatic lookups.
2. In the Add new page:
3. Select **search** for the **Destination app**.
4. Name the lookup **http_status**.
5. Select **http_status** from the **Lookup table** drop down.
6. Apply the lookup to the **sourcetype** named **access_combined**.

7. **Lookup input fields** are the fields in our events that you want to match with the lookup table. Here, both are named **status** (the CSV column name goes on the left and the field that you want to match goes on the right):

8. **Lookup output fields** are the fields from the lookup table that you want to add to your events: **status_description** and **status_type**. The CSV column name goes on the left and the field that you want to match goes on the right.

9. Click **Save**.
Use the configuration files to configure lookups

Introduction to lookup configuration

Lookups add fields from an external source to your events based on the values of fields that are already present in those events. A simple lookup example would be a lookup that works with a CSV file that combines the possible HTTP status values (303, 404, 201, and so on) with their definitions. If you have an event that includes an HTTP status value, the lookup could add the HTTP status description to the event.

You can also use lookups to perform this action in reverse, so that they add fields from your events to rows in a lookup table.

You can configure different types of lookups. Lookups are differentiated in two ways: by data source and by information type.

For more information on dataset types, see Dataset types and usage.

<table>
<thead>
<tr>
<th>Lookup type</th>
<th>Data source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV lookup</td>
<td>A CSV file</td>
<td>Populates your events with fields pulled from CSV files. Also referred to as a &quot;static lookup&quot; because CSV files represent static tables of data. Each column in a CSV table is interpreted as the potential values of a field. CSV inline lookup table files and inline lookup definitions that use CSV files are both dataset types.</td>
</tr>
<tr>
<td>External lookup</td>
<td>An external source, such as a DNS server.</td>
<td>Uses Python scripts or binary executables to populate your events with field values from an external source. Also referred to as a &quot;scripted lookup.&quot;</td>
</tr>
<tr>
<td>Lookup Type</td>
<td>Description</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>KV Store lookup</td>
<td>Matches fields in your events to fields in a KV Store collection and outputs corresponding fields in that collection to your events.</td>
<td>Not a dataset type.</td>
</tr>
<tr>
<td>Geospatial lookup</td>
<td>You use a geospatial lookup to create a query that Splunk software uses to configure a choropleth map. A geospatial lookup matches location coordinates in your events to geographic feature collections in a KMZ (Keyhole Markup Language) file and outputs fields to your events that provide corresponding geographic feature information encoded in the KMZ, like country, state, or county names.</td>
<td>Not a dataset type.</td>
</tr>
</tbody>
</table>

**Configure CSV lookups**

CSV lookups match field values from your events to field values in the static table represented by a CSV file. Then they output corresponding field values from that table to your events. They are also referred to as "static lookups". CSV inline lookup table files and inline lookup definitions that use CSV files are both dataset types. See Dataset types and usage.

Each column in a CSV table is interpreted as a potential value of a field.

**About the CSV files**

There are a few restrictions to the kinds of CSV files that can be used for CSV lookups:

- The table represented by the CSV file must have at least two columns. One of those columns should represent a field with a set of values that
includes those belonging to a field in your events. The column does not have to have the same name as the event field. Any column can have multiple instances of the same value, as this represents a multivalued field.

- The CSV file cannot contain non-utf-8 characters. Plain ascii text is supported, as is any character set that is also valid utf-8.
- The following are unsupported:
  - CSV files with pre-OS X (OS 9 or earlier) Macintosh-style line endings (carriage return (“\r”) only)
  - CSV files with header rows that exceed 4096 characters.

Create a CSV lookup

Prerequisites

- See about lookups and field actions for more information on lookups.
- See use field lookups to add information to your events for information on how to edit lookups.
- See Add field matching rules to your lookup configuration for information on field/value matching rules.
- See Prefilter large CSV lookup tables for information on prefiltering large CSV lookup tables.
- See Configure a time-bounded lookup for information on configuring a time-bounded lookup.
- See Make your lookup automatic for information on configuring an automatic lookup.

Steps

1. Add the CSV file for the lookup to your Splunk deployment. The CSV file must be located in one of the following places:
   
   ```
   $SPLUNK_HOME/etc/system/lookups
   $SPLUNK_HOME/etc/apps/<app_name>/lookups
   ```

   Create the lookups directory if it does not exist.

2. Add a CSV lookup stanza to transforms.conf.
   If you want the lookup to be available globally, add its lookup stanza to the version of transforms.conf in
   
   ```
   $SPLUNK_HOME/etc/system/local/
   ```
   If you want the lookup to be specific to a particular app, add its stanza to the version of transforms.conf in $SPLUNK_HOME/etc/apps/<app_name>/local/.

   **Caution:** Do not edit configuration files in $SPLUNK_HOME/etc/system/default.
The CSV lookup stanza names the lookup table and provides the name of the CSV file that the lookup uses. It uses these required fields.

♦ [<lookup_name>]: The name of the lookup table.
♦ filename = <string>: The name of the CSV file that the lookup references.

3. (Optional) Use the `check_permission` field in `transforms.conf` and `outputlookup_check_permission` in `limits.conf` to restrict write access to users with the appropriate permissions when using the `outputlookup` command.

Both `check_permission` and `outputlookup_check_permission` default to false. Set to true for Splunk software to verify permission settings for lookups for users.

You can change lookup table file permissions in the `.meta` file for each lookup file, or in `Settings > Lookups > Lookup table files`. By default, only users who have the admin or power role can write to a shared CSV lookup file.

4. (Optional) Use the `filter` field to prefilter large CSV lookup tables.

You may need to prefilter significantly large CSV lookup tables. To do this use the `filter` field to restrict searches.

5. (Optional) Set up field/value matching rules for the CSV lookup.

6. (Optional) If the CSV lookup table contains time fields, make the CSV lookup time-bounded.

7. (Optional) Make the CSV lookup automatic by adding a configuration to `props.conf`.

   If you want the automatic lookup to be available globally, add its lookup stanza to the version of `props.conf` in `$SPLUNK_HOME/etc/system/local/`. If you want the lookup to be specific to a particular app, add its stanza to the version of `props.conf` in `$SPLUNK_HOME/etc/apps/<app_name>/local/`.

Caution: Do not edit configuration files in `$SPLUNK_HOME/etc/system/default`.

8. Restart Splunk Enterprise to implement your changes.

   If you have set up an automatic lookup, after restart you should see the output fields from your lookup table listed in the fields sidebar. From there, you can select the fields to display in each of the matching search results.

Handle large CSV lookup tables

Lookup tables are created and modified on a search head. The search head replicates a new or modified lookup table to other search heads, or to indexers.
to perform certain tasks.

- **Knowledge bundle replication.** When a search head distributes searches to indexers, it also distributes a related knowledge bundle to the indexers. The knowledge bundle contains knowledge objects, such as lookup tables, that the indexers need to perform their searches. See What search heads send to search peers in *Distributed Search.*

- **Configuration replication (search head clusters).** In search head clusters, runtime changes made on one search head are automatically replicated to all other search heads in the cluster. If a user creates or updates a lookup table on a search head in a cluster, that search head then replicates the updated table to the other search heads. See Configuration updates that the cluster replicates in *Distributed Search.*

When a lookup table changes, the search head must replicate the updated version of the lookup table to the other search heads or the indexers, or both, depending on the situation. By default, the search head sends the entire table each time any part of the table changes.

If a lookup table is very large, for example larger than 100MB, repeated replication of the table can add significantly to internal network traffic.

**Replicating lookup tables only on the search heads**

There are situations in which you might not want to replicate lookup tables to the indexers. For example, if you are using the `outputcsv` or `inputcsv` commands, those commands always run on the search head. If you only want to replicate the lookup table on the search heads in a search head clustering setup, set `replicate=false` in the `transforms.conf` file.

**Enable custom indexing**

You can also improve lookup performance by configuring which fields are indexed by using the `index_fields_list` setting in the `transforms.conf` file. The `index_fields_list` is a list of all fields that need to be indexed for your static CSV lookup file.

**Prerequisites**

- Access to the `transforms.conf` files, located in $SPLUNK_HOME/etc/apps/<app_name>/local/
- A CSV lookup file
Steps

1. In the `transforms.conf` file, add the `index_fields_list` setting to your lookup table.
   
   The `index_fields_list` setting is a comma and space-delimited list of all of the fields that need to be indexed for your static CSV lookup file.

The default for the `index_fields_list` setting is all of the fields that are defined in the lookup table file header. Restricting the fields will improve lookup performance.

Prefilter CSV lookup tables

If you know that you only need results from a subset of records in the lookup table, improve search performance by using the `filter` field to filter out all of the records that do not need to be looked at. The `filter` field requires a string containing a search query with Boolean expressions and/or comparison operators (==, !=, >, <, <=, >=, OR, AND, and NOT). This query runs whenever you run a search that invokes this lookup.

For example, if your lookup configuration has `filter = (CustID>500) AND (CustName="P*")`, it will try to retrieve values only from records that have a `CustID` value greater than 500 and a `CustName` value beginning with the letter P.

You can also filter records from CSV tables when you use the `WHERE` clause in conjunction with the `inputlookup` and `inputcsv` commands, when you use those commands to search CSV files.

CSV lookup example

This example explains how you can set up a lookup for HTTP status codes in an `access_combined` log. In this example, you design a lookup that matches the `status` field in your events with the `status` column in a lookup table named `http_status.csv`. Then you have the lookup output the corresponding `status_description` and `status_type` fields to your events.

The following is the text for the `http_status.csv` file.

```
status,status_description,status_type
100,Continue,Informational
```
101, Switching Protocols, Informational
200, OK, Successful
201, Created, Successful
202, Accepted, Successful
203, Non-Authoritative Information, Successful
204, No Content, Successful
205, Reset Content, Successful
206, Partial Content, Successful
300, Multiple Choices, Redirection
301, Moved Permanently, Redirection
302, Found, Redirection
303, See Other, Redirection
304, Not Modified, Redirection
305, Use Proxy, Redirection
307, Temporary Redirect, Redirection
400, Bad Request, Client Error
401, Unauthorized, Client Error
402, Payment Required, Client Error
403, Forbidden, Client Error
404, Not Found, Client Error
405, Method Not Allowed, Client Error
406, Not Acceptable, Client Error
407, Proxy Authentication Required, Client Error
408, Request Timeout, Client Error
409, Conflict, Client Error
410, Gone, Client Error
411, Length Required, Client Error
412, Precondition Failed, Client Error
413, Request Entity Too Large, Client Error
414, Request-URI Too Long, Client Error
415, Unsupported Media Type, Client Error
416, Requested Range Not Satisfiable, Client Error
417, Expectation Failed, Client Error
500, Internal Server Error, Server Error
501, Not Implemented, Server Error
502, Bad Gateway, Server Error
503, Service Unavailable, Server Error
504, Gateway Timeout, Server Error
505, HTTP Version Not Supported, Server Error

1. Put the `http_status.csv` file in `~/SPLUNK_HOME/etc/apps/search/lookups/`. This indicates that the lookup is specific to the Search App.
2. In the `transforms.conf` file located in `~/SPLUNK_HOME/etc/apps/search/local`, put:

   ```
   [http_status]
   filename = http_status.csv
   ```

3. Restart Splunk Enterprise to implement your changes.

Now you can invoke this lookup in search strings with the following commands:
• **lookup**: Use to add fields to the events in the results of the search.
• **inputlookup**: Use to search the contents of a lookup table.
• **outputlookup**: Use to write fields in search results to a CSV file that you specify.

See the topics on these commands in the *Search Reference* for more information about how to do this.

For example, you could run this search to add `status_description` and `status_type` fields to events that contain `status` values that match `status` values in the CSV table.

```
... | lookup http_status status OUTPUT status_description, status_type
```

**Use search results to populate a CSV lookup table**

You can edit a local or app-specific copy of `savedsearches.conf` to use the results of a report to populate a lookup table.

In a report stanza, where the search returns a results table:

1. Add the following line to enable the lookup population action.

   ```
   action.populate_lookup = 1
   ```
   This tells Splunk software to save your results table into a CSV file.

2. Add the following line to specify where to copy your lookup table.

   ```
   action.populate_lookup.dest = <string>
   ```
   The `action.populate_lookup.dest` value is a lookup name from `transforms.conf` or a path to a CSV file where the search results are to be copied. If it is a path to a CSV file, the path should be relative to `$SPLUNK_HOME`.

   For example, if you want to save the results to a global lookup table, you might include:

   ```
   action.populate_lookup.dest = etc/system/lookups/myTable.csv
   ```
   The destination directory, `$SPLUNK_HOME/etc/system/lookups` or `$SPLUNK_HOME/etc/<app_name>/lookups`, should already exist.

3. Add the following line if you want this search to run when Splunk Enterprise starts up.

   ```
   run_on_startup = true
   ```
If it does not run on startup, it will run at the next scheduled time. We recommend that you set `run_on_startup = true` for scheduled searches that populate lookup tables.

Because the results of the reporter copied to a CSV file, you can set up this lookup the same way you set up a CSV lookup.

**Configure external lookups**

External lookups invoke a script that matches fields in your events with fields in an external source and outputs corresponding fields from that external source and adds them to your events.

External lookups are often referred to as scripted lookups, because they are facilitated through the use of a script. See About the external lookup script for information about how these scripts work.

**Create an external lookup**

The following is the steps required to create an external lookup for a Splunk Enterprise deployment. If you have Splunk Cloud and want to define external lookups, file a Support ticket.

**Prerequisites**

- About lookups
- Define an external lookup in Splunk Web
- Add field matching rules to your lookup configuration
- Configure a time-bounded lookup
- Make your lookup automatic

**Steps**

1. Add the script for the lookup to your Splunk deployment.
   The script must be located in one of two places:
   ♦ $SPLUNK_HOME/etc/searchscripts
   ♦ $SPLUNK_HOME/etc/apps/<app_name>/bin
2. Add an external lookup stanza to `transforms.conf`.
   If you want the lookup to be available globally, add its lookup stanza to the version of `transforms.conf` in $SPLUNK_HOME/etc/system/local/. If you want the lookup to be specific to a particular app, add its stanza to the version of
transforms.conf in $SPLUNK_HOME/etc/apps/<app_name>/local/.

Caution: Do not edit configuration files in $SPLUNK_HOME/etc/system/default.

The external lookup stanza names the lookup table, provides the script and argument to perform lookups, identifies the script type, and supplies a list of fields that are supported by the script. It uses these required attributes.

♦ [<lookup_name>]: The name of the lookup.
♦ external_cmd = <string>: The command and arguments issued to perform the lookup. The command must be the name of the script, such as external_lookup.py. The arguments are the names of the fields that you want to pass to the script, separated by spaces, like this: clienthost clientip.
♦ external_type = [python|executable|kvstore|geo]: The type of script being used for the lookup. Can be python, for a Python script, or executable, for a binary executable. The kvstore and geo values are reserved for KV store lookups and geospatial lookups, respectively.
♦ fields_list = <string>: is a list of all fields that are supported by the external lookup. The fields must be delimited by a comma followed by a space.

3. (Optional) Set up field/value matching rules for the external lookup.

4. (Optional) If the data source for the external lookup contains time fields, make the external lookup time-bounded.

5. (Optional) Make the external lookup automatic by adding a configuration to props.conf.
   
   If you want the automatic lookup to be available globally, add its lookup stanza to the version of props.conf in $SPLUNK_HOME/etc/system/local/. If you want the lookup to be specific to a particular app, add its stanza to the version of props.conf in $SPLUNK_HOME/etc/apps/<app_name>/local/.

Caution: Do not edit configuration files in $SPLUNK_HOME/etc/system/default.

6. Restart Splunk Enterprise to implement your changes.

   If you have set up an automatic lookup, after restart you should see the output fields from your lookup table listed in the fields sidebar. From there, you can select the fields to display in each of the matching search results.

External lookup example

Here’s an example of an external lookup that is delivered with Splunk software. It matches with information from a DNS server. It does not have a props.conf
component, so it is not an automatic lookup. You access it by running a search with the `lookup` command.

Splunk Enterprise ships with a script located in `$SPLUNK_HOME/etc/system/bin/` called `external_lookup.py`, which is a DNS lookup script that:

- if given a host, returns the IP address.
- if given an IP address, returns the host name.

The configuration for this script resides in `$SPLUNK_HOME/etc/system/default/transforms.conf`.

```bash
[dnslookup]
external_cmd = external_lookup.py clienthost clientip
fields_list = clienthost,clientip
```

You can run a search with the `lookup` command that uses the `[dnslookup]` stanza from the default `transforms.conf`.

```splunk
sourcetype=access_combined | lookup dnslookup clienthost AS host | stats count by clientip
```

This search:

- Matches the `clienthost` field in the external lookup table with the `host` field in your events.</code>
- Returns a table that provides a count for each of the `clientip` values that corresponds with the `clienthost` matches.

This search does not add fields to your events.

You can also design a search that performs a reverse lookup, which in this case returns a host value for each IP address it receives.

```splunk
sourcetype=access_combined | lookup dnslookup clientip | stats count by clienthost
```

Note that this reverse lookup search does not include an AS clause. This is because Splunk automatically extracts IP addresses as `clientip`.

**About the external lookup script**

Your external lookup script must take in a partially empty CSV file and output a filled-in CSV file. The arguments that you pass to the script are the headers for
these input and output files.

In the DNS lookup example above, the CSV file contains two fields: clienthost and clientip. The fields that you pass to this script are the ones you specify in transforms.conf using the external_cmd attribute. If you do not pass these arguments, the script returns an error.

```
external_cmd = external_lookup.py clienthost clientip
```

When you run this search string:

```
... | lookup dnsLookup clienthost
```

You are telling Splunk software to:

1. Use the lookup table that you defined in transforms.conf as [dnsLookup]  
2. Pass the values for the clienthost field into the external command script as a CSV file. The CSV file looks like this.

```
clienthost,clientip
work.com
home.net
```

This is a CSV file with clienthost and clientip as column headers, but without values for clientip. The script includes the two headers because they are the fields you specified in the fields_list attribute of the [dnslookup] stanza in the default transforms.conf.

The script then outputs the following CSV file, which is used to populate the clientip field in your results:

```
host,ip
work.com,127.0.0.1
home.net,127.0.0.2
```

**Note:** When writing your script, if you refer to any external resources (such as a file), the reference must be relative to the directory where the script is located.

**See also**

In addition to using external lookups to add fields from external sources to events, you might use a scripted input to send data from non-standard sources for indexing or to prepare this data for parsing. For more information, see the Scripted inputs overview in Developing Views and Apps for Splunk Web.
Configure KV Store lookups

KV Store lookups populate your events with fields pulled from your App Key Value Store (KV Store) collections. KV Store lookups can be invoked through REST endpoints or by using the following search commands: lookup, inputlookup, and outputlookup.

This topic shows you how to set up and manage KV Store lookups by configuring lookup stanzas in props.conf. Configuration files give you a greater degree of control over lookup design and behavior than you get when you set up lookup files using Splunk Web. However, if you do not have access to the .conf files, or if you prefer to maintain lookups through Splunk Web whenever possible, you can configure KV Store lookups using the pages at Settings > Lookups. See Define a KV Store lookup in Splunk Web.

You can also set up KV Store lookups as automatic lookups. Automatic lookups run in the background at search time and automatically add output fields to events that have the correct match fields. You do not need to invoke automatic lookups with the lookup command. See Make your lookup automatic.

**Splunk Cloud users:** You must use Splunk Web to define lookups. If your Splunk Cloud deployment is a managed deployment, you must request a restart from Splunk Support after uploading lookup files, to make newly uploaded files appear in the list of files available for defining lookups.

For developer-focused KV Store lookup configuration instructions, see Use lookups with KV Store data in the Splunk Developer Portal.

About KV Store collections

Before you create a KV Store lookup, your Splunk deployment must have at least one KV Store collection defined in collections.conf. See Use configuration files to create a KV Store collection on the Splunk Developer Portal.

KV Store collections are containers of data similar to a database. They store your data as key/value pairs. When you create a KV Store lookup, the collection should have at least two fields. One of those fields should have a set of values that that match with the values of a field in your event data, so that lookup matching can take place.

When you invoke the lookup in a search with the lookup command, you designate a field in your search data to match with the field in your KV Store
collection. When a value of this field in an event matches a value of the
designated field in your KV Store collection, the corresponding value(s) for the
other field(s) in your KV Store collection can be added to that event.

The KV Store field does not have to have the same name as the field in your
events. Each KV Store field can be multivalued.

**Note:** KV Store collections live on the search head, while CSV files are replicated
to indexers. If your lookup data changes frequently you may find that KV Store
lookups offer better performance than an equivalent CSV lookup.

**Define a KV Store lookup stanza in transforms.conf**

A *transforms.conf* KV Store lookup stanza provides the location of the KV Store
collection that is to be used as a lookup table. It can optionally include field
matching rules and rules for time-bounded lookups.

If you want a KV Store lookup to be available globally, add its lookup stanza to
the version of *transforms.conf* in $SPLUNK_HOME/etc/system/local/. If you want
the lookup to be specific to a particular app, add its stanza to the version of
*transforms.conf* in $SPLUNK_HOME/etc/apps/<app_name>/local/.

**Caution:** Do not edit configuration files in $SPLUNK_HOME/etc/system/default.

**The KV Store lookup stanza format**

When you add a KV Store lookup stanza to *transforms.conf* it should follow this
format.

```
[<lookup_name>]
external_type = kvstore
collection = <string>
case_sensitive_match = <bool>
fields_list = <string>
filter = <string>
```

- `<lookup_name>` is the name of the lookup.
- `external_type` should be set to `kvstore` if you are defining a KV store
  lookup.
- `case_sensitive_match` defaults to true. If set to false, case insensitive
  matching will be performed for all fields in a lookup table. Output fields and
  values in the KV Store used for matching must be lower case.
- `collection` is the name of the KV Store collection associated with the lookup.
- `fields_list` is a list of all fields that are supported by the KV Store lookup. The fields must be delimited by a comma followed by a space. A field can be any combination of key and value that you have in your KV store collection.

By default, each KV Store record has a unique key ID, which is stored in the internal `_key` field. Add `_key` to the list of fields in `fields_list` if you want to be able to modify specific records through your KV Store lookup. You can then specify the key ID value in your lookup operations.

When you use the `outputlookup` command to write to the KV Store without specifying a key ID, a key ID is generated for you.

- `filter`: Optionally use this attribute to improve search performance when working with significantly large KV Store collections. See Prefilter large KV Store collections.

### Configure a KV Store lookup

#### Prerequisites

- See About lookups for more information on lookups.
- See Make your lookup automatic for information on configuring an automatic KV store lookup.
- See Use configuration files to create a KV Store collection store on the Splunk Developer Portal.
- See Prefilter large KV Store collections for information on prefiltering large KV store collections.
- See Add field matching rules to your lookup configuration for information on field/value matching rules.
- See Configure a time-bounded lookup for information on configuring a time-bounded lookup.

#### Steps

If you have Splunk Cloud and want to define KV store lookups, file a Support ticket. If you have Splunk Enterprise, perform the following steps.

1. Define a KV Store collection in `collections.conf`.
2. Create a KV Store lookup stanza in `transforms.conf`, following the stanza format described above.
If you want the lookup to be available globally, add its lookup stanza to the version of `transforms.conf` in `$SPLUNK_HOME/etc/system/local/`. If you want the lookup to be specific to a particular app, add its stanza to the version of `transforms.conf` in `$SPLUNK_HOME/etc/apps/<app_name>/local/`.

**Caution:** Do not edit configuration files in `$SPLUNK_HOME/etc/system/default`.

3. (Optional) Use the `filter` attribute to prefilter significantly large KV Store lookup tables.
   
   You can speed up lookup searches against significantly large KV Store collections by using the `filter` attribute to restrict the searches.

4. (Optional) Set up field/value matching rules for the KV Store lookup.

5. (Optional) If the KV Store collection contains time fields, make the KV Store lookup time-bounded.

6. (Optional) Make the KV Store lookup an automatic lookup by adding a configuration to `props.conf`.
   
   If you want the automatic lookup to be available globally, add its lookup stanza to the version of `props.conf` in `$SPLUNK_HOME/etc/system/local/`. If you want the lookup to be specific to a particular app, add its stanza to the version of `props.conf` in `$SPLUNK_HOME/etc/apps/<app_name>/local/`.

   **Caution:** Do not edit configuration files in `$SPLUNK_HOME/etc/system/default`.

7. Save your `.conf` file changes.

8. Restart Splunk Enterprise to implement your changes.
   
   If you have set up an automatic lookup, after restart you should see the output fields from your lookup table listed in the fields sidebar. From there, you can select the fields to display in each of the matching search results.

### Prefilter large KV Store collections

When your KV Store collection is extremely large, performance can suffer when your lookups must search through the entire collection to retrieve matching field values. If you know that you only need results from a subset of records in the lookup table, improve search performance by using the `filter` attribute to filter out all of the records that do not need to be looked at.

The `filter` attribute requires a string containing a search query with Boolean expressions and/or comparison operators (==, !=, >, <, <=, >=, OR, AND, and NOT). This query runs whenever you run a search that invokes this lookup.
For example, if your lookup configuration has `filter = (CustID>500) AND (CustName="P*")`, it tries to retrieve values only from those records in the KV Store collection that have a `CustID` value that greater than 500 and a `CustName` value that begins with the letter P.

**Note:** If you do not want to install a filter in the lookup definition you can get a similar effect when you use the `where` clause in conjunction with the `inputlookup` command.

**KV store lookup example**

Here is a KV Store lookup called `employee_info`. It is located in your app's `$SPLUNK_HOME/etc/system/local/` directory.

```
[employee_info]
external_type = kvstore
case_sensitive_match = true
collection = kvstorecoll
fields_list = _key, CustID, CustName, CustStreet, CustCity, CustZip
filter = (CustID>500) AND (CustName="P*)
```

The `employee_info` lookup takes an employee ID in an event and outputs corresponding employee information to that event such as the employee name, street address, city, and zip code. The lookup works with a KV Store collection called `kvstorecoll`. The `filter` restricts the lookup query to records with a customer ID greater than 500 and a customer name that begins with the letter "P".

To see how to make this KV Store lookup automatic by adding a configuration to `props.conf`, see Make your lookup automatic.

**Search commands and KV Store lookups**

After you save a KV Store lookup stanza and restart Splunk Enterprise, you can interact with the new KV store lookup through search commands.

Use `lookup` to match values in a KV Store collection with field values in the search results and then output corresponding field values to those results. This search uses the `employee_info` lookup defined in the preceding use case example.

```
... | lookup employee_info CustID AS ID OUTPUT CustName AS Name | ...
```
It matches employee id values in `kvstorecoll` with employee id values in your events and outputs the corresponding employee name values to your events.

You can use the `inputlookup` search command to search on the contents of a KV Store collection. See the Search Reference topic on `inputlookup` for examples.

You can use the `outputlookup` search command to write search results from the search pipeline into a KV store collection. See the Search Reference topic on `outputlookup` for examples.

You can also find several examples of KV Store lookup searches in Use lookups with KV Store data in the *Splunk Developer Portal*.

**Configure geospatial lookups**

You use geospatial lookups to create queries that return results that Splunk software can use to generate a choropleth map visualization. Choropleth maps cannot be rendered without the data generated by corresponding geospatial lookups.

A geospatial lookup matches location coordinates in your events to location coordinate ranges in a geographic feature collection known as a Keyhole Markup Zipped (KMZ) or Keyhole Markup Language (KML) file and outputs fields to your events that provide corresponding geographic feature information that is encoded in the feature collection. This information represents a geographic region that shares borders with geographic regions of the same type, such as a country, state, province, or county.

Splunk provides two geospatial lookups for the United States and for world countries, enabling you to render choropleth maps of:

- The USA, divided up into state regions.
- The world, divided up into countries.

This topic shows you how to create additional geospatial lookups that break up choropleth maps into other types of regions (counties, provinces, timezones, and so on).

For more information about choropleth maps and geographic data visualizations, see Mapping data, in the *Dashboards and Visualizations* manual.
The FeatureId and featureCollection fields

Geospatial lookups differ from other lookup types in that they are designed to always output these two fields: featureId and featureCollection. The featureId is the "name" of the feature, like "California" or "CA" or whatever is encoded in the feature collection. The featureCollection field provides the name of the lookup in which the feature was found.

If you pipe the output of a geospatial lookup directly into a geom command, the command does not need to be given the lookup name. The geom command detects the featureId and featureCollection fields in the event and uses the lookup to generate the geographic data structures that the Splunk software requires to generate a choropleth map. However, be aware that geographic data structures can be large; so much so that it is strongly discouraged to pipe events into the geom command, as geographic data structures will be attached to every event. Instead, you should first perform stats on the results of your geographic lookup, and only perform geom on a "boiled down" aggregated statistic like count by featureId.

Define a geospatial lookup stanza in transforms.conf

The geospatial lookup stanza provides the location of the geographic feature collection that is to be used as a lookup table. It can optionally include:

- a feature_id_element attribute.
- field matching rules.
- rules for time-bounded lookups.

See the geospatial lookup stanza format for details.

If you want a geospatial lookup to be available globally, add its lookup stanza to the version of transforms.conf in $SPLUNK_HOME/etc/system/local/. If you want the lookup to be specific to a particular app, add its stanza to the version of transforms.conf in $SPLUNK_HOME/etc/apps/<app_name>/local/.

Caution: Do not edit configuration files in $SPLUNK_HOME/etc/system/default.

The geospatial lookup stanza format

When you create a geospatial lookup definition, it should follow this format.

[<lookup_name>]
external_type = geo
filename = <name_of_KMZ_file>
feature_id_element = <XPath_expression>

• [<lookup_name>] is the name of the lookup.
• external_type should be set to geo if you are defining a geospatial lookup.
• filename is the name of the KMZ file that you are using. KMZ files are also referred to as "geographic feature collections."
  ♦ Two feature collections are provided: geo_us_states for the United States, and geo_countries for the countries of the world.
  ♦ You can optionally upload geographic feature collections for other regions and feature types, such as US counties or European provinces.
• feature_id_element is an optional attribute. It is an XPath expression that defines a path from a Polygon element in the KML file to some other XML element or attribute that contains the name of the feature. Splunk software calls these Polygon elements a "feature". This is needed in cases where the typical style of named Placemark element is not in use.
  ♦ feature_id_element may be required in cases where the featureID field generated by the lookup is an empty string, or when the feature collection returns incorrect features by default. In the latter case the feature that you want may be a peer of the default feature or is located relative to the default feature.
  ♦ To determine what path you need, study the geographic feature collection. Each feature in the collection is tagged with Placemark, and each Placemark contains a name that the lookup writes out as featureId fields. For an example, see feature_id_element.
  ♦ The default setting for feature_id_element is /Placemark/name.

XPath and feature_id_element example

The following is an example Placemark element extracted from a KML file.

```xml
<Placemark>
  <name>Bayview Park</name>
  <visibility>0</visibility>
  <styleUrl>#msn_ylw-pushpin15</styleUrl>
  <Polygon>
    <tessellate>1</tessellate>
    <outerBoundaryIs>
      <LinearRing>
        <coordinates>
          -122.3910323868129,37.70819686392956,0
          -122.3902274700583,37.71036559447013,0
          -122.3885849520798,37.71048623150828,0
        </coordinates>
      </LinearRing>
    </outerBoundaryIs>
  </Polygon>
</Placemark>
```
The Placemark element contains both a name element and a Polygon element. A Placemark can have multiple Polygons. Placemark associates a name to a set of Polygons, called a "feature." However, different KML files may organize their data differently, so we need to tell Splunk software where to find the name, relative to the Placemark element. We can do this with the feature_id_element configuration. By default, feature_id_element contains the XPath expression /Placemark/name.

Let's take a look at another Placemark element extracted from a KML file.

```xml
<Placemark>
  <name>MyFeature</name>
  <ExtendedData>
    <SchemaData>
      <SimpleData name="placename">foo</SimpleData>
      <SimpleData name="bar">baz</SimpleData>
    </SchemaData>
  </ExtendedData>
</Placemark>
```
The XPath expression for this Placemark fragment would be
`feature_id_element=/Placemark/ExtendedData/SchemaData/SimpleData[@name='placename']`.

**Configure a geospatial lookup**

**Prerequisites**

- See about lookups and field actions for more information on lookups.
- See Add field matching rules to your lookup configuration for information on field/value matching rules.
- See Make your lookup automatic for information on configuring an automatic lookup.

**Steps**

1. (Optional) Upload a geographic feature collection to your Splunk deployment, if you need to use a collection other than `geo_us_states` or `geo_countries`.
   - Geographic feature collections are encoded as KMZ (Keyhole Markup Language) files.
   - Upload the feature collection in Settings. Navigate to **Settings > Lookups > Lookup table files**.
   - If you have a KML file, you can convert it to a KMZ file by compressing it and replacing the `.zip` extension with `.kmz`.

2. Create a geospatial lookup stanza in `transforms.conf`, following the stanza format described in "The geospatial lookup stanza format," above.
   - If you want the lookup to be available globally, add its lookup stanza to the version of `transforms.conf` in `$SPLUNK_HOME/etc/system/local/`. If you want the lookup to be specific to a particular app, add its stanza to the version of `transforms.conf` in `$SPLUNK_HOME/etc/apps/<app_name>/local/`.
   - **Caution:** Do not edit configuration files in `$SPLUNK_HOME/etc/system/default`.

3. (Optional) Set up field/value matching rules for the geospatial lookup.

4. (Optional) Make the geospatial lookup an automatic lookup by adding a configuration to `props.conf`.
   - If you want the automatic lookup to be available globally, add its lookup stanza to the version of `props.conf` in `$SPLUNK_HOME/etc/system/local/`. If you want the lookup to be specific to a particular app, add its stanza to the version of `props.conf` in `$SPLUNK_HOME/etc/apps/<app_name>/local/`.

214
Caution: Do not edit configuration files in $SPLUNK_HOME/etc/system/default.

5. Save your .conf file changes.
6. Restart Splunk Enterprise to implement your changes.
   If you have set up an automatic lookup, after restart you should see the output fields from your lookup table listed in the fields sidebar. From there, you can select the fields to display in each of the matching search results.

Search commands and geospatial lookups

After you save a geospatial lookup stanza and restart Splunk Enterprise, you can interact with the new geospatial lookup through the inputlookup search command. You can use inputlookup to quickly check the featureIds of your geospatial lookup or show all geographic features on a Choropleth map visualization.

This example uses the default geo_us_states lookup.

Prerequisites

- A geospatial lookup, see above.

Steps:

1. From the Search and Reporting app, use the inputlookup command to search on the contents of your geospatial lookup.

   | inputlookup geo_us_states

2. Check to make sure that your featureIds are in the lookup with the featureId column.
3. Click on the Visualization tab.
4. Click on Cluster Map and select Chloropleth Map for your visualization.

A chloropleth map displaying the featureIds of your geospatial lookup appears. For more information on chloropleth maps, see Generate a chloropleth map in the Dashboards and Visualizations manual.

Geospatial lookup example

Here is a geospatial lookup called geo_us_states. It is located in $SPLUNK_HOME/etc/system/bin/.
This lookup deals with a geographic feature collection that contains US states.

To use this lookup to build a choropleth map, you need to create a search that queries it in a manner that returns results that can be used to generate the map. The search needs to do all of these things:

- Indicate an events data source.
- Query the lookup by matching fields in your events to fields in the KMZ file.
- Include a transforming command that aggregates the data using `featureId`, the lookup’s geographic output field.
- Use the `geom` search command to generate data that can be used to create a choropleth map.

This is a partial choropleth map query. It meets the first two of the four requirements listed above for a choropleth map lookup search. It returns the latitude and longitude for features in the feature collection.

```
sourcetype=crime_data cc=USA | lookup geo_us_states latitude, longitude
```

The output of that lookup should look something like this:

<table>
<thead>
<tr>
<th>_featureIdField</th>
<th>featureId</th>
<th>featureCollection</th>
<th>latitude</th>
<th>longitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>featureId</td>
<td>AK</td>
<td>geo_us_states</td>
<td>y</td>
<td>x</td>
</tr>
<tr>
<td>featureId</td>
<td>AL</td>
<td>geo_us_states</td>
<td>y</td>
<td>x</td>
</tr>
</tbody>
</table>

You can update this search to display results for the `geom` command. Note that the `geom` command should be preceded by a transforming command operation, such as this one involving `count`.

This is a full choropleth map query. It retrieves crime event counts by US state and adds the geometry of each state as a `geom` column.

```
sourcetype=crime_data cc=USA | lookup geo_us_states latitude, longitude | stats count by featureId | geom
```

The output of that search should look something like this:

```
<table>
<thead>
<tr>
<th>_featureIdField</th>
<th>featureId</th>
<th>geom</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
_featureIdField is a hidden field that works with the geomfilter post-process search command, when you run a search that contains it. It allows geomfilter to know which field contains the featureId values, even when featureId is renamed to something else.

For example, say you rename featureId to state. If you run geomfilter, it consults the stored search results in the search dispatch folder and looks in the _featureIdField column, where it finds the value state. This causes it to seek the featureId values it needs for its calculations in the state column.

For more information about geospatial lookup search queries, see Mapping data in the Dashboards and Visualizations manual.

### Add field matching rules to your lookup configuration

These attributes provide field matching rules for lookups. They can be applied to all three lookup types. Add them to the transforms.conf stanza for your lookup.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_matches</td>
<td>Integer</td>
<td>The maximum number of possible matches for each value input to the lookup table from your events. Range is 1-1000. If the time_field attribute is is not specified, Splunk software uses the first &lt;integer&gt; entries, in file order. If the time_field attribute is specified (because it is a time-bounded lookup), Splunk software uses the first &lt;integer&gt; entries, in descending time order. In other words, up to &lt;max_matches&gt; are allowed to match. When this number is surpassed, Splunk software uses the matches closest to the lookup value.</td>
<td>1000 if the time_field attribute is not specified. 1 if the time_field attribute is specified.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td>Value</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>min_matches</td>
<td>Integer</td>
<td>The minimum number of possible matches for each value input to the lookup table from your events. You can use default_match to help with situations where there are fewer than min_matches for any given input.</td>
<td>0 for both non-time-bounded lookups and time-bounded lookups, which means nothing is output to your event if no match is found.</td>
</tr>
<tr>
<td>default_match</td>
<td>String</td>
<td>When min_matches is greater than 0 and Splunk software finds fewer than min_matches for any given input, it provides this default_match value one or more times until the min_matches threshold is reached.</td>
<td>Empty string</td>
</tr>
<tr>
<td>case_sensitive_match</td>
<td>Boolean</td>
<td>Specify true to consider case when matching lookup table fields, false to ignore case.</td>
<td>True</td>
</tr>
<tr>
<td>match_type</td>
<td>String</td>
<td>Allows non-exact matching of one or more fields arranged in a list delimited by a comma followed by a space. Format is match_type = &lt;match_type&gt;(&lt;field_name1&gt;, &lt;field_name2&gt;,...&lt;field_nameN&gt;). Set match_type to WILDCARD to apply wildcard matching, or set it to CIDR to apply CIDR matching (specifically for IP address values).</td>
<td>EXACT (does not need to be specified)</td>
</tr>
</tbody>
</table>

### Configure a time-based lookup

If your lookup table has a field that represents time, you can use it to create a time-based lookup. This is also referred to as a temporal lookup. You can configure all four lookup types as time-based lookups.

Simple time-based lookups attempt to match the event timestamp with the timestamp of a record in the lookup table, and then perform operations like adding one or more fields to the event from the matched record.
You can also define time-bounded lookups, which use the event time to define a range of time within which to match lookup records. For example, you could create a time-bounded lookup that matches the first lookup table record with a timestamp that falls within 10 seconds before the event timestamp.

Defining time-based lookups

To create a simple time-based lookup, add the following lines to your lookup stanza in `transforms.conf`:

```
time_field = <field_name>
time_format = <string>
```

Here are the definitions of these settings.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>time_field</td>
<td>Identifies the field in the lookup table that represents the timestamp. The search processor applies the first matching entry in descending order. When <code>time_field</code> is present in a saved search stanza, <code>max_matches = 1</code> by default. For more information about <code>max_matches</code> see Add field matching rules to your lookup configuration.</td>
<td>Defaults to an empty string, because lookups are not time-based by default.</td>
</tr>
<tr>
<td>time_format</td>
<td>Specifies the <code>strftime()</code> format of the <code>time_field</code> attribute. You can use some nonstandard date-time <code>strftime()</code> formats. See the material about enhanced <code>strftime()</code> support in Configure timestamp recognition in the <em>Getting Data In Manual</em>.</td>
<td><code>%s.%Q</code></td>
</tr>
</tbody>
</table>

Defining time-bounded lookups

To create a time bounded lookup, add these optional settings to your time-based lookup configuration:

```
max_offset_secs = <integer>
min_offset_secs = <integer>
```

Here are the definitions of these settings:
### Setting Description Default

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_offset_secs</td>
<td>The maximum amount of time in seconds that an event timestamp can be later than the lookup record timestamp, for a match to occur.</td>
<td>2000000000 (effectively no default)</td>
</tr>
<tr>
<td>min_offset_secs</td>
<td>The minimum amount of time in seconds that an event timestamp can be later than the lookup record timestamp, for a match to occur.</td>
<td>0</td>
</tr>
</tbody>
</table>

The `max_offset_secs` and `min_offset_secs` settings define the earliest and latest times within which the search processor can search for matching records in the lookup table. The search processor calculates the earliest and latest time values from the event time like this:

\[
\text{earliest} = \text{event timestamp} - \text{max_offset_secs} \\
\text{latest} = \text{event timestamp} - \text{min_offset_secs}
\]

Within this window of time, the search processor applies a match in descending order of time up to the point where we get `max_matches` number of matches for that event. If `max_matches` is not set, it defaults to 1. For more information about `max_matches` see Add field matching rules to your lookup configuration.

### Time-based lookup example

Here's an example of a CSV lookup that uses DHCP logs to identify users on a network based on their IP address and the timestamp. The DHCP logs are in a file, `dhcp.csv`, which contains the timestamp, IP address, and the user's name and MAC address.

### Prerequisites

- See about lookups and field actions for more information on lookups.
- See Make your lookup automatic for information on configuring an automatic lookup.

### Steps

1. In a `transforms.conf` file, put:

   ```plaintext
   [dhcpLookup] 
   filename = dhcp.csv 
   time_field = timestamp 
   ```
2. In a props.conf file, make the lookup automatic:

```
[dhcp]
LOOKUP-table = dhcpLookup ip mac OUTPUT user
```

3. Save your file changes.

If you wanted to turn this into a time-bounded lookup, you could add the following settings to the [dhcpLookup] stanza in transforms.conf:

```
max_offset_secs = 10
min_offset_secs = 0
```

This would cause the lookup to match events to the first lookup table record with a timestamp that falls within a range of time bounded by the event timestamp and ten seconds before the event timestamp.

**Make your lookup automatic**

When you create a lookup configuration in transforms.conf, you invoke it by running searches that reference it. However, you can optionally create an additional props.conf configuration that makes the lookup "automatic." This means that it runs in the background at search time and automatically adds output fields to events that have the correct match fields.

You can make all lookup types automatic. However, KV Store lookups have an additional setup step that you must complete before you configure them as automatic lookups in props.conf. See Enable replication for a KV Store collection.

Each automatic lookup configuration you create is limited to events that belong to a specific host, source, or source type. Automatic lookups can access any data in a lookup table that belongs to you or which you have shared.

When your lookup is automatic you do not need to invoke its transforms.conf configuration with the `lookup` command.

Splunk software does not support nested automatic lookups.

**The automatic lookup format in props.conf**

An automatic lookup configuration in props.conf:

```
time_format = %d/%m/%y %H:%M:%S
```
References the lookup table you configured in *transforms.conf*.

- Specifies the fields in your events that the lookup should match in the lookup table.
- Specifies the corresponding fields that the lookup should output from the lookup table to your events.

At search time, the `LOOKUP-<class>` configuration identifies a lookup and describes how that lookup should be applied to your events. To create an automatic lookup, follow this syntax:

```
[<spec>]
LOOKUP-<class> = $TRANSFORM <match_field_in_lookup_table>
OUTPUT|OUTPUTNEW <output_field_in_lookup_table>
```

- **The stanza header is** `[<spec>]`. `<spec>` can be:
  - `<sourcetype>`, the source type of an event.
  - `host::<host>`, where `<host>` is the host, or host-matching pattern, for an event.
  - `source::<source>`, where `<source>` is the source, or source-matching pattern, for an event.
- `<spec>` cannot use regular expression syntax.
- `$TRANSFORM`: References the *transforms.conf* stanza that defines the lookup table.
- `match_field_in_lookup_table`: This variable is the field in your lookup table that matches a field in your events with the same host, source, or source as this *props.conf* stanza. If the match field in your events has a different name from the match field in the lookup table, use the AS clause as specified in Step 3, below.
- `output_field_from_lookup_table`: The corresponding field in the lookup table that you want to add to your events. If the output field in your events should have a different name from the output field in the lookup table, use the AS clause as specified in Step 3, below.

You can have multiple fields on either side of the lookup. For example, you can have:

```
$TRANSFORM <match_field_in_lookup_table1>,
<match_field_in_lookup_table2>OUTPUT|OUTPUTNEW
<output_field_from_lookup_table1>, <output_field_from_lookup_table2>
```

You can also have one matching field return two output fields, three matching fields return one output field, and so on.
If you do not include an `OUTPUT|OUTPUTNEW` clause, Splunk software adds all the field names and values from the lookup table to your events. When you use `OUTPUTNEW`, Splunk software can add only the output fields that are "new" to the event. If you use `OUTPUT`, output fields that already exist in the event are overwritten.

If the "match" field names in the lookup table and your events are not identical, or if you want to "rename" the output field or fields that get added to your events, use the `AS` clause:

```
[<stanza name>]
LOOKUP-<class> = $TRANSFORM <match_field_in_lookup_table> AS <match_field_in_event> OUTPUT|OUTPUTNEW <output_field_from_lookup_table> AS <output_field_in_event>
```

For example, if the lookup table has a field named `dept` and you want the automatic lookup to add it to your events as `department_name`, set `department_name` as the value of `<output_field_in_event>`.

**Note:** You can have multiple `LOOKUP-<class>` configurations in a single `props.conf` stanza. Each lookup should have its own unique lookup name. For example, if you have multiple lookups, you can name them `LOOKUP-table1`, `LOOKUP-table2`, and so on.

You can also have different `props.conf` automatic lookup stanzas that each reference the same lookup stanza in `transforms.conf`.

**Create an automatic lookup stanza in props.conf**

1. Create a stanza header that references the host, source, or source type that you are associating the lookup with.
2. Add a `LOOKUP-<class>` configuration to the stanza that you have identified or created.
   As described in the preceding section this configuration specifies:
   ♦ What fields in your events it should match to fields in the lookup table.
   ♦ What corresponding output fields it should add to your events from the lookup table.
   Be sure to make the `<class>` value unique. You can run into trouble if two or more automatic lookup configurations have the same `<class>` name. See "Do not use identical names in automatic lookup configurations."
3. (Optional) Include the `AS` clause in the configuration when the "match" field names in the lookup table and your events are not identical, or when
you want to "rename" the output field or fields that get added to your events, use the AS clause.

4. Restart Splunk Enterprise to apply your changes.
   If you have set up an automatic lookup, after restart you should see the output fields from your lookup table listed in the fields sidebar. From there, you can select the fields to display in each of the matching search results.

Enable replication for a KV store collection

In Splunk Enterprise, KV Store collections are not bundle-replicated to indexers by default, and lookups run locally on the search head rather than on remote peers. When you enable replication for a KV Store collection, you can run the lookups on your indexers which let you use automatic lookups with your KV Store collections.

To enable replication for a KV Store collection and allow lookups against that collection to be automatic:

1. Open collections.conf.
2. Set replicate to true in the stanza for the collection.
   This parameter is set to false by default.
3. Restart Splunk Enterprise to apply your changes.

If your indexers are running a version of Splunk Enterprise that is older than 6.3, attempts to run an automatic lookup fail with a "lookup does not exist" error. You must upgrade your indexers to 6.3 or later to use this functionality.

For more information, see Use configuration files to create a KV Store collection at the Splunk Developer Portal.

Example configuration of an automatic KV Store lookup

This configuration references the example KV Store lookup configuration in Configure KV Store lookups, in this manual. The KV Store lookup is defined in transforms.conf, in a stanza named employee_info.

```
[access_combined]
LOOKUP-http = employee_info CustID AS cust_ID OUTPUT CustName AS cust_name, CustCity AS cust_city
```

This configuration uses the employee_info lookup in transforms.conf to add fields to your events. Specifically it adds cust_name and cust_city fields to any
access_combined event with a cust_ID value that matches a custID value in the kvstorecoll KV Store collection. It also uses the AS clause to:

- Find matching fields in the KV Store collection.
- Rename output fields when they are added to events.
Workflow actions

About workflow actions in Splunk Web

Enable a wide variety of interactions between indexed or extracted fields and other web resources with workflow actions. Workflow actions have a wide variety of applications. For example, you can define workflow actions that enable you to:

- Perform an external WHOIS lookup based on an IP address found in an event.
- Use the field values in an HTTP error event to create a new entry in an external issue management system.
- Launch secondary searches that use one or more field values from selected events.
- Perform an external search (using Google or a similar web search application) on the value of a specific field found in an event.

In addition, you can define workflow actions that:

- Are targeted to events that contain a specific field or set of fields, or which belong to a particular event type.
- Appear either in field menus or event menus in search results. You can also set them up to only appear in the menus of specific fields, or in all field menus in a qualifying event.
- When selected, open either in the current window or in a new one.

Define workflow actions using Splunk Web

You can set up workflow actions using Splunk Web. To begin, navigate to Settings > Fields > Workflow actions. On the Workflow actions page, you can review and update existing workflow actions by clicking on their names. Or you can click Add new to create a new workflow action. Both methods take you to the workflow action detail page, where you define individual workflow actions.

If you're creating a new workflow action, you need to give it a Name and identify its Destination app.

There are three kinds of workflow actions that you can set up.
<table>
<thead>
<tr>
<th>Workflow action type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET workflow actions</td>
<td>GET workflow actions create typical HTML links to do things like perform Google searches on specific values or run domain name queries against external WHOIS databases.</td>
</tr>
<tr>
<td>POST workflow actions</td>
<td>POST workflow actions generate an HTTP POST request to a specified URI. This action type enables you to do things like creating entries in external issue management systems using a set of relevant field values.</td>
</tr>
<tr>
<td>Search workflow actions</td>
<td>Search workflow actions launch secondary searches that use specific field values from an event, such as a search that looks for the occurrence of specific combinations of <code>ipaddress</code> and <code>http_status</code> field values in your index over a specific time range.</td>
</tr>
</tbody>
</table>

**Target workflow actions to a narrow grouping of events**

When you create workflow actions in Splunk Web, you can optionally target workflow actions to a narrow grouping of events. You can restrict workflow action scope by field, by event type, or a combination of the two.

**Narrow workflow action scope by field**

You can set up workflow actions that only apply to events that have a specified field or set of fields. For example, if you have a field called `http_status`, and you would like a workflow action to apply only to events containing that field, you would declare `http_status` in the **Apply only to the following fields** setting.

If you want to have a workflow action apply only to events that have a set of fields, you can declare a comma-delimited list of fields in **Apply only to the following fields**. When more than one field is listed the workflow action is displayed only if the entire list of fields are present in the event.

For example, say you want a workflow action to only apply to events with `ip_client` and `ip_server` fields. To do this, you would enter `ip_client, ip_server` in **Apply only to the following fields**.

Workflow action field scoping also supports use of the wildcard asterisk. For example, if you declare a simple field listing of `ip_*` Splunk software applies the resulting workflow action to events with either `ip_client` or `ip_server` as well as a combination of both (as well as any other event with a field that matches `ip_*`).

By default the field list is set to `*`, which means that it matches all fields.
If you need more complex selecting logic, we suggest you use event type scoping instead of field scoping, or combine event type scoping with field scoping.

**Narrow workflow action scope by event type**

Event type scoping works the same way as field scoping. You can enter a single event type or a comma-delimited list of event types into the **Apply only to the following event types** setting to create a workflow action that only applies to events belonging to that event type or set of event types. You can also use wildcard matching to identify events belonging to a range of event types.

You can also narrow the scope of workflow actions through a combination of fields and event types. For example, if you have a field called `http_status`, but you only want the resulting workflow action to appear in events containing that field if the `http_status` is greater than or equal to 500. To accomplish this, you would need to set up an event type called `errors_in_500_range` that is applied to events matching a search like

```
http_status >= 500
```

Then, you would define a workflow action that has **Apply only to the following fields** set to `http_status` and **Apply only to the following event types** set to `errors_in_500_range`.

For more information about event types, see About event types in this manual.

**Set up a GET workflow action**

GET link workflow actions drop one or more values into an HTML link. Clicking that link performs an HTTP GET request in a browser, allowing you to pass information to an external web resource, such as a search engine or IP lookup service.

**Note**: During transmission, variables passed in URIs for GET actions are URL encoded. This means you can include values that have spaces between words or punctuation characters. However, if you are working with a field that has an HTTP address as its value, and you want to pass the entire field value as a URI, you should use the `$!` prefix to keep Splunk software from escaping the field value. See "Use the $! prefix to prevent escape of URL or HTTP form field values" below for more information.
Define a GET workflow action

Steps

1. Navigate to Settings > Fields > Workflow Actions.
2. Click New to open up a new workflow action form.
3. Define a Label for the action.
   - The Label field enables you to define the text that is displayed in either the field or event workflow menu. Labels can be static or include the value of relevant fields.
4. Determine whether the workflow action applies to specific fields or event types in your data.
   - Use Apply only to the following fields to identify one or more fields. When you identify fields, the workflow action only appears for events that have those fields, either in their event menu or field menus. If you leave it blank or enter an asterisk the action appears in menus for all fields.
   - Use Apply only to the following event types to identify one or more event types. If you identify an event type, the workflow action only appears in the event menus for events that belong to the event type.
5. For Show action in determine whether you want the action to appear in the Event menu, the Fields menus, or Both.
6. Set Action type to link.
7. In URI provide a URI for the location of the external resource that you want to send your field values to.
   - Similar to the Label setting, when you declare the value of a field, you use the name of the field enclosed by dollar signs.
   - Variables passed in GET actions via URIs are automatically URL encoded during transmission. This means you can include values that have spaces between words or punctuation characters.
8. Under Open link in, determine whether the workflow action displays in the current window or if it opens the link in a new window.
9. Set the Link method to get.
10. Click Save to save your workflow action definition.

Example - Google search from field values

Here’s an example of the setup for a GET link workflow action that sets off a Google search on values of the topic field in search results:
In this example, we set the Label value to \texttt{Google $\text{topic}$} because we have a field called \texttt{topic} in our events and we want the value of \texttt{topic} to be included in the label for this workflow action. For example, if the value for \texttt{topic} in an event is \texttt{CreatefieldactionsinSplunkWeb} the field action displays as \texttt{Google CreatefieldactionsinSplunkWeb} in the \texttt{topic} field menu.

The \texttt{Google $\text{topic}$} action applies to all events.

The \texttt{Google $\text{topic}$} action URI uses the \texttt{GET} method to submit the \texttt{topic} value to Google for a search.

\textbf{Example - Provide an external IP lookup}

You have configured your Splunk app to extract domain names in web services logs and specify them as a field named \texttt{domain}. You want to be able to search an external WHOIS database for more information about the domains that appear.

Here’s how you would set up the GET workflow action that helps you with this.

In the Workflow actions details page, set \textbf{Action type} to \textit{link} and set \textbf{Link method} to \textit{get}.

You then use the **Label** and **URI** fields to identify the field involved. Set a **Label** value of **WHOIS: $domain$**. Set a **URI** value of http://whois.net/whois/$domain$.

After that, you can determine:

- whether the link shows up in the field menu, the event menu, or both.
- whether the link opens the WHOIS search in the same window or a new one.
- restrictions for the events that display the workflow action link. You can target the workflow action to events that have specific fields, that belong to specific event types, or some combination of the two.

**Use the $!$ prefix to prevent escape of URL or HTTP form field values**

When you define fields for workflow actions, you can escape these fields so that they can be passed safely to an external endpoint using HTTP. However, in certain cases this escaping is undesirable. In these cases, use the $!$ prefix to prevent the field value from being escaped. This prefix prevents URL escape for GET workflow actions and HTTP form escape for POST workflow actions.

*Example - Passing an HTTP address to a separate browser window*

You have a GET workflow action that works with a field named **http**. The **http** field has fully formed HTTP addresses as values. This workflow action opens a new browser window that points at the HTTP address value of the **http** field. The workflow action does not work if it opens the new window with an escaped HTTP address.

To prevent the HTTP address from escaping, use the $!$ prefix. In Settings, where you might normally set **URI** to $http$ for this workflow action, instead set it to $!http$.

**Set up a POST workflow action**

You set up POST **workflow actions** in a manner similar to that of GET link actions. However, POST requests are typically defined by a form element in HTML along with some inputs that are converted into POST arguments. This means that you have to identify POST arguments to send to the identified URI.
Note: During transmission, variables passed in URIs for POST actions are URL encoded, which means you can include values that have spaces between words or punctuation characters. However, if you are working with a field that has an HTTP address as its value, and you want to pass the entire field value as a URI, you should use the $! prefix to keep Splunk software from escaping the field value. See "Use the $! prefix to prevent escape of URL or HTTP form field values" below for more information.

1. Navigate to Settings > Fields > Workflow Actions.
2. Click New to open up a new workflow action form.
3. Define a Label for the action.
   The Label field enables you to define the text that is displayed in either the field or event workflow menu. Labels can be static or include the value of relevant fields.
4. Determine whether the workflow action applies to specific fields or event types in your data.
   Use Apply only to the following fields to identify one or more fields. When you identify fields, the workflow action only appears on events that have those fields, either in their event menu or field menus. If you leave it blank or enter an asterisk the action appears in menus for all fields.
   Use Apply only to the following event types to identify one or more event types. If you identify an event type, the workflow action only appears in the event menus for events that belong to the event type.
5. For Show action in determine whether you want the action to appear in the Event menu, the Fields menus, or Both.
6. Set Action type to Link.
7. Under URI provide the URI for a web resource that responds to POST requests.
8. Under Open link in, determine whether the workflow action displays in the current window or if it opens the link in a new window.
9. Set Link method to Post.
10. Under Post arguments define arguments that should be sent to web resource at the identified URI.
    These arguments are key and value combinations. On both the key and value sides of the argument, you can use field names enclosed in dollar signs to identify the field value from your events that should be sent over to the resource. You can define multiple key/value arguments in one POST workflow action. Enter the key in the first field, and the value in the second field. Click Add another field to create an additional POST argument.
11. Click **Save** to save your workflow action definition. Splunk software automatically HTTP-form encodes variables that it passes in POST link actions via URIs. This means you can include values that have spaces between words or punctuation characters.

**Example - Allow an http error to create an entry in an issue tracking application**

You have configured your Splunk app to extract HTTP status codes from a web service log as a field called `http_status`. Along with the `http_status` field the events typically contain either a normal single-line description request, or a multiline python stacktrace originating from the python process that produced an error.

You want to design a workflow action that only appears for error events where `http_status` is in the 500 range. You want the workflow action to send the associated python stacktrace and the HTTP status code to an external issue management system to generate a new bug report. However, the issue management system only accepts POST requests to a specific endpoint.

Here's how you might set up the POST workflow action that fits your requirements:
Note that the first POST argument sends server error $\text{http_status}$ to a title field in the external issue tracking system. If you select this workflow action for an event with an http_status of 500, then it opens an issue with the title server error 500 in the issue tracking system.

The second POST argument uses the _raw field to include the multiline python stacktrace in the description field of the new issue.

Finally, note that the workflow action has been set up so that it only applies to events belonging to the errors_in_500_range event type. This is an event type that is only applied to events carrying http_error values in the typical HTTP error range of 500 or greater. Events with HTTP error codes below 500 do not display the submit error report workflow action in their event or field menus.

**Use the $! prefix to prevent escape of URL or HTTP form field values**

When you define fields for workflow actions, you can escape these fields so that they can be passed safely to an external endpoint using HTTP. However, in
certain cases this escaping is undesirable. In these cases, use the $! prefix to prevent the field value from being escaped. This prefix prevents URL escape for GET workflow actions and HTTP form escape for POST workflow actions.

**Example - Passing an HTTP address to a separate browser window**

You have a GET workflow action that works with a field named `http`. The `http` field has fully formed HTTP addresses as values. This workflow action opens a new browser window that points at the HTTP address value of the `http` field. The workflow action does not work if it opens the new window with an escaped HTTP address.

To prevent the HTTP address from escaping, use the $! prefix. In Settings, where you might normally set `URI` to `$http$` for this workflow action, instead set it to `$!http$`.

**Set up a search workflow action**

To set up workflow actions that launch dynamically populated secondary searches, you start by setting **Action type** to **search** on the Workflow actions detail page. This reveals a set of **Search configuration** fields that you use to define the specifics of the secondary search.

In **Search string** enter a search string that includes one or more placeholders for field values, bounded by dollar signs. For example, if you're setting up a workflow action that searches on client IP values that turn up in events, you might simply enter `clientip=$clientip$` in that field.

Identify the app that the search runs in. If you want it to run in a view other than the current one, select that view. And as with all workflow actions, you can determine whether it opens in the current window or a new one.

Be sure to set a time range for the search (or identify whether it should use the same time range as the search that created the field listing) by entering relative time modifiers in the in the **Earliest time** and **Latest time** fields. If these fields are left blank the search runs over all time by default.

Finally, as with other workflow action types, you can restrict the search workflow action to events containing specific sets of fields and/or which belong to particular event types.
Example - Launch a secondary search that finds errors originating from a specific Ruby On Rails controller

In this example, we will be using a web infrastructure that is built on Ruby on Rails. You’ve set up an event type to sort out errors related to Ruby controllers (titled controller_error), but sometimes you just want to see all the errors related to a particular controller. Here’s how you might set up a workflow action that does this.

1. On the Workflow actions detail page, set up an action with the following Label: See other errors for controller $controller$ over past 24h.
2. Set Action type to Search.
3. Enter the following Search string: sourcetype=rails controller=$controller$ error=* 
4. Set an Earliest time of -24h. Leave Latest time blank.
5. Using the Apply only to the following... settings, arrange for the workflow action to only appear in events that belong to the controller_error event type, and which contain the error and controller fields.

Those are the basics. You can also determine which app or view the workflow action should run in (for example, you might have a dedicated view for this information titled ruby_errors) and identify whether the action works in the
current window or opens a new one.

**Control workflow action appearance in field and event menus**

When workflow actions are set up correctly, they appear in menus associated with fields and events in your search results. You can arrange for workflow actions to be event-level (meaning they apply to an entire event), field-level (meaning they apply to specific fields within events), or both.

To select event-level workflow actions:

- Run a search.
- Go to the **Events** tab.
- Expand an event in your search results and click **Event Actions**.

Here's an example of "Show Source," an event-level workflow action that, when clicked, displays the source for the event in your raw search data.

Alternatively, you can have the workflow action appear in the **Actions** menus for fields within an event. Here's an example of a workflow action that opens a Google search in a separate window for the selected field and value.
Both of these examples are of workflow actions that use the GET link method.

You can also define workflow actions that appear both at the event level and the field level. For example, you might do this for workflow actions that do something with the value of a specific field in an event, such as User_ID.

**Use special parameters in workflow actions**

There are special parameters for workflow actions that begin with an "@" sign. Two of these special parameters are for field menus only. They enable you to set up workflow actions that apply to all fields in the events to which they apply:

- @field_name - Refers to the name of the field being clicked on.
- @field_value - Refers to the value of the field being clicked on.

The other special parameters are:

- @sid - Refers to the sid of the job that returned the event
- @offset - Refers to the offset of the event in the job
- @namespace - Refers to the namespace from which the job was dispatched
- @latest_time - Refers to the latest time the event occurred. It is used to distinguish similar events from one another. It is not always available for all fields.

**Example - Create a workflow action that applies to all fields in an event**

You can update the Google search example discussed above (in the GET link workflow action section) so that it enables a search of the field name and field value for every field in an event to which it applies. All you need to do is change the title to Google this field and value and replace the URI of that action with http://www.google.com/search?q=$@field_name$+$@field_value$.

This results in a workflow action that searches on whichever field/value combination you're viewing a field menu for. If you're looking at the field menu for sourcetype=access_combined and select the Google this field and value field action, the resulting Google search is sourcetype accesscombined.

**Remember:** Workflow actions using the @field_name and/or @field_value parameters are not compatible with event-level menus.
**Example - Show the source of an event**

This workflow action uses the other special parameters to show the source of an event in your raw search data.

The *Action type* is *link* and its *Link method* is *get*. Its *Title* is *Show source*. The *URI* is

```
/app/$@namespace$/show_source?sid=$@sid$&offset=$@offset$&latest_time=$@latest_time$
```

It's only applied to events that have the _cd field.

Try setting this workflow action up in your app (if it isn't installed already) and see how it works.
Tags

About tags and aliases

In your data, you might have groups of events with related field values. To search more efficiently for these groups of event data, you can assign tags and aliases to your data.

If you tag tens of thousands of items, use field lookups. Using many tags will not affect indexing, but your search has better event categorization when using lookups. For more information on field lookups, see About lookups.

Tags

Tags enable you to assign names to specific field and value combinations, including event type, host, source, or source type.

You can use tags to help you track abstract field values, like IP addresses or ID numbers. For example, you could have an IP address related to your main office with the value 192.168.1.2. Tag that IPAddress value as mainoffice, and then search for that tag to find events with that IP address.

You can use a tag to group a set of field values together, so that you can search for them with one command. For example, you might find that you have two host names that refer to the same computer. You could give both of those values the same tag. When you search for that tag, events that involve both host name values are returned.

You can give extracted fields multiple tags that reflect different aspects of their identity, which enable you to perform tag-based searches to help you narrow the search results.

Tags example

You have an extracted field called IPAddress, which refers to the IP addresses of the data sources within your company intranet. You can tag each IP address based on its functionality or location. You can tag all of your routers’ IP addresses as router, and tag each IP address based on its location, for example, SF or Building1. An IP address of a router located in San Francisco inside Building 1 could have the tags router, SF, and Building1.
To search for all routers in San Francisco that are not in Building1, use the following search.

tag=router tag=SF NOT (tag=Building1)

**Tags and the search-time operations sequence**

When you run a search, Splunk software runs several operations to derive knowledge objects and apply them to events returned by the search. Splunk software performs these operations in a specific sequence.

*Search-time operation order*

Tags come last in the sequence of search-time operations.

*Restrictions*

The Splunk software applies tags to field/value pairs in events in ASCII sort order. You can apply tags to any field/value pair in an event, whether it is extracted at index time, search time, or added through some other method, such as an event type, lookup, or calculated field.

*For more information*

For more information about search-time operations, see search-time operations sequence.

**Field aliases**

**Field aliases** enable you to normalize data from multiple sources. You can add multiple aliases to a field name or use these field aliases to normalize different field names. The use of Field aliases does not rename or remove the original field name. When you alias a field, you can search for it with any of its name aliases. You can alias field names in Splunk Web or in props.conf. See Create field aliases in Splunk Web.

You can use aliases to assign different extracted field names to a single field name.

Field aliases for all source types are used in all searches, which can produce a lot of overhead over time.
**Field Aliases example**

One data model might have a field called `http_referrer`. This field might be misspelled in your source data as `http_referer`. Use field aliases to capture the misspelled field in your original source data and map it to the expected field name.

**Field aliases and the search-time operations sequence**

**Search-time operations order**

Field aliasing comes fourth in the search-time operations order, before calculated fields but after automatic key-value field extraction.

**Restrictions**

Splunk software processes field aliases belonging to a specific host, source, or sourcetype in ASCII sort order. You can create aliases for fields that are extracted at index time or search time. You cannot create aliases for fields that are added to events by search-time operations that come after the field aliasing process.

**For more information**

For more information about search-time operations, see search-time operations sequence.

**Tag field-value pairs in Search**

You might have groups of events with related field-value pairs. To help you search for these events, assign the same tag to these related field-value pairs.

See About tags and aliases.

**Tag field-value pairs**

You can tag any field-value pair directly from the results of a search.

**Prerequisites**

- See About tags and aliases for information on tags.
Steps

1. Locate an event with a field-value pair that you want to tag.
2. Expand a row to see the full list of fields extracted from the event.
3. Click the **Actions** arrow for the field-value pair that you want to create a tag for and select **Edit Actions**.

4. In the Create Actions dialog box, define one or more tags for the field-value pair.
   Values for the **Tag(s)** field must be separated by commas or spaces.

5. Click **Save**.

**Remove URL-encoded values from tag definitions**

When you tag a field-value pair, the value part of the pair cannot be URL-encoded. If your tag has any %## format URL-encoding, decode it and then save the tag with the decoded URL.

For example, you want to give the following field-value pair the tag Useful.

```
```

Define the tag in Settings.

1. Select **Settings > Tags > List by tag name**.
2. Click on the **Useful** tag name to open the detail page for that tag.
3. Under **Field-value pair** replace
   
   `url=http%3A%2F%2Fdocs.splunk.com%2FDocumentation` with the decoded
See Define and manage tags in Settings.

**Search for tagged field values**

You have two ways to search for tags. To search for a tag associated with a value in any field, use the following syntax:

\[ \text{tag} = \text{<tagname>} \]

To search for a tag associated with a value in a specific field, use the following syntax:

\[ \text{tag::<field>} = \text{<tagname>} \]

**Use wildcards to search for tags**

You can use the asterisk (*) wildcard when you search keywords and field values, including for eventtypes and tags.

For example, if you have multiple eventtype tags for various types of IP addresses, such as `IP-src` and `IP-dest`, you can search for all of them with:

\[ \text{tag::eventtype} = \text{IP-}^* \]

To find all hosts whose tags contain "local", search for the following tag:

\[ \text{tag::host} = \text{*local}* \]

To search for the events with eventtypes that have no tags, you can search for the following Boolean expression:

\[ \text{NOT} \text{ tag::eventtype} = \text{*} \]

**Disable and delete tags**

You can remove a tag association for a specific field value through the Search app. You can also disable or delete tags, even if they are associated with multiple field values in Settings.

See Define and manage tags in Settings.
**Remove a tag association for a specific field value in search results**

You can remove a tag associated with a field value in your search results.

1. Click the arrow next to the event.
2. Under **Actions**, click open the arrow next to the field value.
3. Select Edit Tags to open the **Create Tags** window.
4. In the **Create Tags** window, delete the tags that you want to disable from the **Tags** field.
5. Click **Save**.

This action removes this tag and field value association from the system. If this is the only field value the tag is associated with, the tag is removed from the system.

**Tagging event types in settings**

You can add tags when you create or edit an event type. See Tag event types.

**Rename source types**

When you configure a source type in `props.conf`, rename the source type. Multiple source types can share the same name, so you can group a set of source types for a search. For example, you can normalize source type names that include "-too_small" to remove the classifier. See Rename source types at search time.

**Define and manage tags in Settings**

Splunk software provides multiple methods for tag creation and management. Most users use the simplest method tagging field-value pairs directly in search results. See Tag and alias field values in Search.

Use the Tags page in Settings to manage the tags created by users of your Splunk deployment.

- Manage tags for your Splunk deployment.
- Create tags.
- Disable or delete tags.
Using the Tags page in Settings

The Tags page in Settings gives you three views of your tags. Each view is a different tag organization.

- List by field value pair
- List by tag name
- All unique tag objects

Use these pages to create, edit, or delete tags, and manage the sets of tags that are associated with specific field-value pairs or apps.

Before you create new field-value pair records on the Settings pages for tags, verify that the field-value pairs exist in your data. The Splunk platform does not perform validation to ensure that you are not associating tags to nonexistent field-value pairs.

Managing associations between field-value pairs and tag sets

The List by field-value pair page lists the field-value pairs that are associated with tag sets. On this page you can perform the following actions if the permissions associated with your role allow it:

- Create a new field-value pair record. Click New to provide a field-value pair and one or more tag names.
- Edit the list of tags that are associated with with a field-value pair. Click a field-value pair name and add or remove tag names.
- Update permissions for a field-value pair. Click Permissions. The Splunk platform applies field-value pair permission updates to all tags associated with the pair.
- Disable all tags associated with a field-value pair. If you need to remove tags from your search results without deleting them, disable them instead.
- Clone, Move, or Delete an association between a field-value pair and a set of tags.

The List by field-value pair page breaks out field-value pairs by app. If you have the same field-value pair in multiple apps, it appears on multiple rows of the List by field-value pair page. This means you can apply a change to the association between a field-value pair and a set of tags in one app without affecting field-value pairs or tags in other apps.

For example, say you have the same field-value pair in the Search & Reporting and Enterprise Security apps, and in both apps it is associated with the same two
tags: `tag-a` and `tag-b`. If you disable the association between the field-value pair and the two tags in the Search & Reporting app, the field-value pair will continue to be associated with the `tag-a` and `tag-b` in the ES app.

As a knowledge manager, consider using a carefully designed and maintained set of tags. This practice aids with data normalization, and can reduce confusion on the part of your users.

See Manage knowledge objects through Settings pages.

**Managing associations between tags and sets of field-value pairs**

The List by tag name page lists each of the tags that are in your Splunk platform deployment. It lists each tag once, even if the tag appears in multiple apps or is associated with multiple field-value pairs.

On this page you can perform the following actions if the permissions associated with your role allow it:

- Create new tags. Click **New** to define a tag name and provide a field-value pair.
- Edit the field-value pair lists for tags. Click the tag name to add, remove, or edit the field-value pairs that are associated with a tag.
- **Clone** or **Delete** tags.
- **Disable** tags.

When you disable a tag through the List by tag name page, the Splunk platform disables the tag across all apps that contain the tag. The row for the disabled tag also disappears from the list. To re-enable disabled tags, go to the List by field-value pair page, locate the related field-value pair, and add the name of the disabled tag to it.

The List by tag name page does not allow you to manage permissions for the set of field-value pairs associated with a tag.

As a knowledge manager, consider using a carefully designed and maintained set of tags. This practice aids with data normalization, and can reduce confusion on the part of your users.

See Manage knowledge objects through Settings pages.
**Reviewing all unique field-value pair and tag combinations**

The All unique tag objects page lists out all of the unique tag name, field-value pairing, and app combinations in your deployment. This page lets you edit one-to-one relationships between tags and field-value pairs. If a tag object is identical to tag objects in other apps, it will appear multiple times in this list, once for each app.

You can search for a particular tag to quickly see all of the field-value pairs with which it's associated, or you can disable or clone a particular tag and field-value association, or you can maintain permissions at that level of granularity.

**Disabling and deleting tags**

If you have a tag that you no longer want to use, or want to have associated with a particular field-value pairing, you can disable it or remove it.

- Remove a tag association for a specific field-value pair in the search results.
- Bulk disable or delete a tag, even if it is associated to multiple field values, with the List by tag name page.
- Bulk disable or delete the associations between a field-value pair and a set of tags by using the List by field-value pair page.

For information about deleting tag associations with specific field-value pairs in your search results, see Tag field-value pairs in Search.

**Delete a tag with multiple field-value pair associations**

You can use Splunk Web to remove a tag from your system, even if it is associated with dozens of field-value pairs. This method lets you get rid of all of these associations in one step.

Select **Settings > Tags > List by tag name**. Delete the tag. If you don't see a delete link for the tag, you don't have permission to delete it. When you delete tags, be aware of downstream dependencies. See Manage knowledge objects through Settings pages.

You can open the edit view for a particular tag and delete a field-value pair association directly.
Disable or delete the associations between a field-value pairing and a set of tags

Use this method to bulk-remove the set of tags that is associated to a field-value pair. This method enables you to get rid of these associations in a single step. It does not remove the field-value pairing from your data, however.

Select Settings > Tags > List by field-value pair. Delete the field-value pair. If you do not see a delete link for the field-value pair, you do not have permission to delete it. When you delete these associations, be aware of downstream dependencies that may be adversely affected by their removal. See Manage knowledge objects through Settings pages.

You can also delete a tag association directly in the edit view for a particular field-value pair.

Disable tags

Depending on your permissions to do so, you can also disable tag and field-value pair associations using the three Tags pages in Settings. When an association between a tag and a field-value pair is disabled, it stays in the system but is inactive until it is enabled again.

When you disable a tag through the List by tag name page, the Splunk platform disables the tag across all apps that contain the tag. The row for the disabled tag also disappears from the list. To reenable disabled tags, go to the List by field-value pair page, locate the related field-value pair, and add the name of the disabled tag to it.

Tag the host field

Tagging the host field is useful for knowledge capture and sharing, and for crafting more precise searches. You can tag the host field with one or more words. Use this to group hosts by function or type, to enable users to easily search for all activity on a group of similar servers. If you've changed the value of the host field for a given input, you can also tag events that are already in the index with the new host name to make it easier to search across your data set.

Add a tag to the host field in search results

You can add a tag to a host field-value combination in your search results.
Prerequisites

- See About tags and aliases.

Steps

1. Perform a search for data from the host you'd like to tag.
2. In the search results, click on the arrow associated with the event containing the field you want to tag. In the expanded list, click on the arrow under Actions associated the field, then select Edit Tags.
3. In the Create Tags dialog enter the host field value that you'd like to tag, for example in Field Value enter Tag host= <current host value>. Enter your tag or tags, separated by commas or spaces, and click Save.

Host names vs. tagging the host field

The value of the host field is set when an event is indexed. It can be set by default based on the Splunk server hostname, set for a given input, or extracted from each event’s data. Tagging the host field with an alternate hostname doesn’t change the actual value of the host field, but it lets you search for the tag you specified instead of having to use the host field value. Each event can have only one host name, but multiple host tags.

For example, if your Splunk server is receiving compliance data from a specific host, tagging that host with compliance will help your compliance searches. With host tags, you can create a loose grouping of data without masking or changing the underlying host name.

You might also want to tag the host field with another host name if you indexed some data from a particular input source and then decided to change the value of
the host field for that input—all the new data coming in from that input will have
the new host field value, but the data that already exists in your index will have
the old value. Tagging the host field for the existing data lets you search for the
new host value without excluding all the existing data.

Tag event types

Tag event types to add information to your data. Any event type can have
multiple tags. For example, you can tag all firewall event types as firewall, tag a
subset of firewall event types as deny and tag another subset as allow. Once an
event type is tagged, any event type matching the tagged pattern will also be
tagged.

Note: You can tag an event type when you create it in Splunk Web or configure it
in eventtypes.conf.

Add tags to event types using Splunk Web

Splunk Web enables you to view and edit lists of event types.

- Navigate to Settings > Event types.
- Locate the event type you want to tag and click on its name to go to its
detail page.
  - Note: Keep in mind that event types are often associated with
    specific Splunk apps. They also have role-based permissions that
can prevent you from seeing and/or editing them.
- On the detail page for the event type, add or edit tags in the Tags field.
- Click Save to confirm your changes.

Once you have tagged an event type, you can search for it in the search bar with
the syntax tag::<field>=<tagname> or tag=<tagname>:

```
tag=foo

tag::host=*local*```
Field aliases

Create field aliases in Splunk Web

In your data, you might have groups of events with related field values. To help you search for these groups of fields, you can assign field aliases to their field values.

Field aliases are an alternate name that you assign to a field allowing you to use that name to search for events that contain that field. A field can have multiple aliases, but a single alias can only apply to one field. For example, the field vendor_action can be aliased to action or message_type, but not both. An alias does not replace or remove the original field name.

Perform field aliasing after key-value extraction, but before field lookups, so that you can specify a lookup table based on a field alias. This can be helpful if one or more fields in the lookup table are identical to fields in your data, but have different names. See Configure CSV and external lookups and Configure KV store lookups.

For more information on aliases, see About tags and aliases.

Use field aliases to normalize your data

You can use Splunk Web to assign an alternate name to a field, allowing you to use that name to search for events that contain that field.

Prerequisites

- See About tags and aliases for more information on aliases.

Steps

1. Locate a field within your search that you would like to alias.
2. Select Settings > Fields > Field aliases.
3. Select an app to use the alias.
4. Enter a name for the alias.
5. Select the host, source, or sourcetype to apply to a default field.
6. Enter the name for the existing field and the new alias.
7. Click Save.
View your new field alias in the Field Aliases page.

### Configure field aliases with props.conf

In your data, you might have groups of events with related field values. To help you search for these groups of fields, you can assign field aliases to their field values. You can assign one or more tags to any extracted field, including event type, host, source, or source type.

Field aliases are an alternate name that you assign to a field, allowing you to use that name to search for events that contain that field. A field can have multiple aliases, but a single alias can only apply to one field. For example, the field `vendor_action` can be aliased to `action` or `message_type`, but not both. An alias does not replace or remove the original field name.

Perform field aliasing after key-value extraction but before field lookups so that you can specify a lookup table based on a field alias. This can be helpful if one or more fields in the lookup table are identical to fields in your data, but are named differently. See Configure CSV and external lookups and Configure KV store lookups.

You can define aliases for fields that are extracted at index time as well as those that are extracted at search time.

Add your field aliases to `props.conf`, which you edit in `$SPLUNK_HOME/etc/system/local/`, or your own custom app directory in `$SPLUNK_HOME/etc/apps/`. Use the latter directory to make it easy to transfer your data customizations to other index servers.

**Note:** Splunk Enterprise supports single value fields only.

### Use props.conf to configure field aliases

**Prerequisites**

- See about tags and aliases for more information on aliases.

**Steps**

1. Add the following line to a stanza in `props.conf`:
FIELDALIAS-<class> = <orig_field_name> AS <new_field_name>

- <orig_field_name> is the original name of the field.
- <new_field_name> is the alias to assign to the field.
- You can include multiple field alias renames in one stanza.
2. Restart Splunk Enterprise for your changes to take effect.

Example of field alias additions for a lookup

You created a lookup for an external static table CSV file, where the field you extracted at search time as ip is referred to as ipaddress. In the props.conf file where you defined the extraction, add a line that defines ipaddress as an alias for ip, as follows:

```
[accesslog]
EXTRACT-extract_ip = (?<ip>\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})
FIELDALIAS-extract_ip = ip AS ipaddress
```

When you set up the lookup in props.conf, use ipaddress where you would otherwise use ip:

```
[dns]
lookup_ip = dnsLookup ipaddress OUTPUT host
```

See Create and maintain search-time field extractions through configuration files.

See Introduction to lookup configuration and Configure KV store lookups.
Search macros

Use search macros in searches

Search macros are reusable chunks of Search Processing Language (SPL) that you can insert into other searches. Search macros can be any part of a search, such as an eval statement or search term and do not need to be a complete command. You can also specify whether the macro field takes any arguments.

Insert search macros into search strings

When you put a search macro in a search string, place a back tick character ( ` ) before and after the macro name. On most English-language keyboards, this character is located on the same key as the tilde (~). You can reference a search macro within other search macros using this same syntax. For example, if you have a search macro named `mymacro` it looks like the following when referenced in a search:

```
sourcetype=access_* | `mymacro`
```

Macros inside of quoted values are not expanded. In the following example, the search macro `bar` is not expanded.

```
"foo`bar`baz"
```

Preview search macros in search strings

Check the contents of your search macro from the Search bar in the Search page using the following keyboard shortcut:

- Command-Shift-E (Mac OSX)
- Control-Shift-E (Linux or Windows)

The shortcut opens a preview that displays the expanded search string, including all nested search macros and saved searches. If syntax highlighting or line numbering are enabled, those features also appear in the preview.

You can copy parts of the expanded search string. You can also click Open in Search to run the expanded search string in a new window. See Preview your search.
**Search macros that contain generating commands**

When you use a search macro in a search string, consider whether the macro expands to an SPL string that begins with a Generating command like `from`, `search`, `metadata`, `inputlookup`, `pivot`, and `tstats`. If it does, you need to put a pipe character before the search macro.

For example, if you know the search macro `mygeneratingmacro` starts with the `tstats` command, you would insert it into your search string as follows:

```
| `mygeneratingmacro`
```

See Define search macros in Settings.

**When search macros take arguments**

If your search macro takes arguments, define those arguments when you insert the macro into the search string. For example, if the search macro `argmacro(2)` includes two arguments that are integers, you might have inserted the macro into your search string as follows: `argmacro(120,300)`.

If your search macro argument includes quotes, escape the quotes when you call the macro in your search. For example, if you pass a quoted string as the argument for your macro, you use: `mymacro("He said \"hello!\"\")`.

Your search macro definition can include the following:

- A validation expression that determines whether the arguments you enter are valid.
- A validation error message that appears when you provide invalid arguments.

**Additional resources**

For more information, see the following resources.

- Define search macros in Settings
- Search macro examples
- Generating commands, in the Search Reference.
Define search macros in Settings

Search macros are reusable chunks of Search Processing Language (SPL) that you can insert into other searches. Search macros can be any part of a search, such as an eval statement or search term, and do not need to be a complete command. You can also specify whether the macro field takes any arguments.

Prerequisites

- See Insert search macros into search strings.
- See Design a search macro definition.
- (Optional) If your search macros require the search writer to provide argument variables, you can design validation expressions that tell the search writer when invalid arguments have been submitted. See Validate search macro arguments.

Steps

1. Select Settings > Advanced Search > Search macros.
2. Click New to create a search macro.
3. (Optional) Check the Destination app and verify that it is set to the app that you want to restrict your search macro to. Select a different app from the Destination app list if you want to restrict your search macro to a different app.
4. Enter a unique Name for the search macro. If your search macro includes an argument, append the number of arguments to the name. For example, if your search macro mymacro includes two arguments, name it mymacro(2).
5. In Definition, enter the search string that the macro expands to when you reference it in another search.
6. (Optional) Click Use eval-based definition? to indicate that the Definition value is an eval expression that returns a string that the search macro expands to.
7. (Optional) Enter any Arguments for your search macro. This is a comma-delimited string of argument names. Argument names may only contain alphanumeric characters (a-Z, A-Z, 0-9), underscores, and dashes. The string cannot contain repetitions of argument names.
8. (Optional) Enter a Validation expression that verifies whether the argument values used to invoke the search macro are acceptable. The validation expression is an eval expression that evaluates to a Boolean or string value.
9. (Optional) Enter a Validation error message if you defined a validation expression. This message appears when the argument values that invoke
Design a search macro definition

The fundamental part of a search macro is its definition, which is the SPL chunk that the macro expands to when you reference it in another search.

If your search macro definition has variables, the macro user must input the variables into the definition as tokens with dollar signs on either side of them. For example, $arg1$ might be the first argument in a search macro definition.

Pipe characters and generating commands in macro definitions

When you use generating commands such as search, inputlookup, or tstats in searches, put them at the start of the search, with a leading pipe character.

If you want your search macro to use a generating command, remove the leading pipe character from the macro definition. Place it at the start of the search string that you are inserting the search macro into, in front of the search macro reference.

For example, you have a search macro named mygeneratingmacro that has the following definition:

```
tstats latest(_time) as latest where index!=filemon by index host source sourcetype
```

The definition of mygeneratingmacro begins with the generating command tstats. Instead of preceding tstats with a pipe character in the macro definition, you put the pipe character in the search string, before the search macro reference. For example:

```
| `mygeneratingmacro`
```

Validate search macro arguments

When you define a search macro that includes arguments that the user must enter, you can define a Validation expression that determines whether the arguments supplied by the user are valid. You can define a Validation error message that appears when search macro arguments fail validation.
The validation expression must be an `eval` expression that evaluates to a Boolean or a string. If the validation expression is boolean, validation succeeds when the validation expression returns `true`. If it returns `false`, or returns null, validation fails.

If the validation expression is not Boolean, validation succeeds when the validation expression returns null. If it returns a string, validation fails.

**Additional resources**

For more information, see the following resources.

- Search macro examples
- `macros.conf` in the Admin Manual. The `macros.conf` configuration file is where Splunk software stores search macro definitions.
- Generating commands in the Search Reference.

**Search macro examples**

Review these search macro use cases and their solutions.

**Prerequisites**

- See Use search macros in searches.
- See Define search macros in Settings.

**Simple search macro with argument**

The following set of partial searches are nearly identical.

```
sourcetype="iis" cs_username="-" /TM/ .pdf
sourcetype="iis" cs_username="-" /TD/ .pdf
sourcetype="iis" cs_username="-" /TDB/ .pdf
```

You want to create a search macro that uses the common parts of this fragment, and that allows you to pass an argument for the variable material between the slashes.

**Steps**
1. Create a search macro named `iis_search(1)` with the following definition:

```
sourcetype="iis" cs_username!="-" /$fragment$/ .pdf
```

2. In the Arguments field, enter fragment as the argument.
3. Click Save.

You can insert `iis_search(fragment=TM)` into your search string to call the search macro for the TM fragment.

**Preview your search to see the contents of your macro**

Use the search preview feature to see the contents of search macros that are embedded within the search, without actually running the search. When you preview a search, the feature expands all of the macros within the search, including macros that are nested within other macros.

**Steps**

1. Navigate to the Splunk Search page.
2. In the Search bar, type the default macro `audit_searchlocal(error)`.
3. Use the keyboard shortcut Command-Shift-E (Mac OSX) or Control-Shift-E (Linux or Windows) to open the search preview.
   The search preview displays syntax highlighting and line numbers, if those features are enabled.

4. (Optional) Copy a fragment of the search.
5. (Optional) Click Open in Search to run the expanded search in a new browser window.

**Combine search macros and transactions**

You can combine transactions and macro searches to simplify your transaction searches and reports. The following example demonstrates how you can use search macros to build reports based on a defined transaction.

A search macro named `makesessions` defines a transaction session from events that share the same `clientip` value, and that occur within 30 minutes of each
other. Following is the definition of `makesessions`:

```
transaction clientip maxpause=30m
```

The following search uses the `makesessions` search macro to take web traffic events and break them into sessions:

```
sourcetype=access_* | `makesessions`
```

The following search uses the `makesessions` search macro to return a report of the number of pageviews per session for each day:

```
sourcetype=access_* | `makesessions` | timechart span=1d
sum(eventcount) as pageviews count as sessions
```

To build the same report with varying span lengths, save the report as a search macro with an argument for the span length. Name the macro `pageviews_per_session(1)`. The macro references the original `makesessions` macro. Following is the definition for this macro:

```
sourcetype=access_* | `makesessions` | timechart $span$ sum(eventcount)
as pageviews count as sessions
```

When you insert the `pageviews_per_session(1)` macro into a search string, you use the argument to specify a span length.

```
`pageviews_per_session(span=1h)`
```

**Validate arguments to determine whether they are numeric**

The following example demonstrates search macro argument validation.

**Steps**

1. Select **Settings > Advanced Search > Search Macros**.
2. Click **New** to create a new search macro.
3. For **Name**, enter `newrate(2)`.
   The (2) indicates that the macro contains two arguments.
4. For **Definition**, enter the following:

   ```
eval new_rate=$val*$rate$
```

   This definition includes the argument variables `val` and `rate`.
5. For the **Argument** field, enter `val` and `rate`.
6. Enter a **Validation expression** that verifies that the value supplied for rate is numeric, as follows:
   `isnum($rate$)`

7. Enter the following **Validation error message**: The rate value that you provided is not numeric. Enter a numeric rate value.

8. Click **Save**.

When another user includes the `newrate(2)` macro in a search, they might fill out the arguments like this: `newrate(revenue, 0.79)`.

If they leave the 0 out (`newrate(revenue, .79)`) the macro is invalid because the value `.79` lacks a leading zero and is interpreted as a string. To ensure that the argument is read as a floating point number, the user should use the `tonumber` function as follows: `newrate(revenue, tonumber(.79))`
Manage and explore datasets

Dataset types and usage

A dataset is a collection of data that you define and maintain for a specific business purpose. It is represented as a table, with fields for columns and field values for cells. You can view and manage datasets with the Datasets listing page.

The Splunk Datasets Add-on, available from Splunkbase, gives Splunk Enterprise users additional dataset management capabilities. Splunk Cloud users have the Splunk Datasets Add-on by default.

Dataset types

You can work with three dataset types. Two of these dataset types, lookups and data models, are existing knowledge objects that have been part of the Splunk platform for a long time. Table datasets, or tables, are a new dataset type that you can create and maintain in Splunk Cloud, and after you download and install the Splunk Datasets Add-on in Splunk Enterprise.

Use the Datasets listing page to view and manage your datasets. See View and manage datasets.

Lookups

The Datasets listing page displays two categories of lookup datasets: lookup table files and lookup definitions. It lists lookup table files for .csv lookups and lookup definitions for .csv lookups and KV Store lookups. Other types of lookups, such as external lookups and geospatial lookups, are not listed as datasets.

You upload lookup table files and create file-based lookup definitions through the Lookups pages in Settings. See About lookups.

Data model datasets

Data models are made up of one or more data model datasets. When a data model is composed of multiple datasets, those datasets can be arranged hierarchically, with a root dataset at the top and child datasets beneath it. In data model dataset hierarchies, child datasets inherit fields from their parent dataset but can also have additional fields of their own.
You create and edit data model dataset definitions with the Data Model Editor. See About data models.

**Note:** In previous versions of the Splunk platform, data model datasets were called data model objects.

**Table datasets**

Table datasets, or tables, are focused, curated collections of event data that you design for a specific business purpose. You can derive their initial data from a simple search, a combination of indexes and source types, or an existing dataset of any type. For example, you could create a new table dataset whose initial data comes from a specific data model dataset. After this new dataset is created, you can modify it by updating field names, adding fields, and more.

You define and maintain datasets with the Table Editor, which translates sophisticated search commands into simple UI editor interactions. It is easy to use, even if you have minimal knowledge of Splunk search processing language (SPL).

The Splunk Datasets Add-on gives you the ability to create and edit table datasets. See Table datasets and the Table Editor.

**Manage datasets**

The Datasets listing page shows all of the datasets that you have access to in your Splunk implementation. You can see what types of datasets you have, who owns them, and how they are shared.

This topic covers the default capabilities of the Datasets listing page.

If the Splunk Datasets Add-on is installed, the Datasets listing page provides additional management features for table datasets. The Splunk Datasets Add-on installed by default in Splunk Cloud and Splunk Light.

For information about the table dataset features of the Datasets listing page, see Manage table datasets.

**Open the Datasets listing page**

In the Search app, click **Datasets** in the green Apps bar.
View dataset detail information

You can expand a dataset row to see details about that dataset, such as the fields contained in the dataset, or the date that the dataset was last modified. When you view the detail information of a table dataset, you can also see the datasets that that table dataset is extended from, if applicable.

1. In the Search & Reporting app, click Datasets to open the Datasets listing page.
2. Find a dataset that you want to review.
3. Click the > symbol in the first column to expand the row of the dataset details.

Explore a dataset

Use the Explorer view to inspect a dataset and determine whether it contains information you want. The Explorer view provides tools for the exploration and management of individual datasets.

- Explore datasets with the View Results and Summarize Fields views.
- Use a time range picker to see what datasets contain for specific time ranges.
- Manage dataset search jobs.
- Export dataset contents.
- Save datasets as scheduled reports.
- Perform the same dataset management actions that exist on the Datasets listing page.

For information on using the Explorer view, see Explore a dataset.

Visualize a dataset with Pivot

Use Pivot to create a visualization based on your dataset. You can save the visualization as a report or as a dashboard panel. You do not need to know how to use the Splunk Search Processing Language (SPL) to use Pivot.

You can open all dataset types in Pivot.
Prerequisites

- Introduction to Pivot in the *Pivot Manual*.

Steps

1. In the Search & Reporting app, click **Datasets**.
2. Find a dataset that you want to work with in Pivot.
3. Select **Explore > Visualize with Pivot**.

You can also access Pivot from the Explorer view. See Explore a dataset.

Investigate a dataset in Search

You can create a search string that uses the `from` command to reference the dataset, and optionally add **SPL** to the search string. You can save the search as a report, alert, or dashboard panel.

The saved report, alert, or dashboard panel is **extended** from the original dataset through a `from` command reference. An extended child dataset is distinct from, but dependent on, the parent dataset from which it is extended. If you change a parent dataset, that change propagates down to all child datasets that are extended from that parent dataset.

Prerequisites

- Get started with Search in the *Search Manual*
- Extend datasets

Steps

1. In the Search & Reporting app, click **Datasets** to open the Datasets listing page.
2. Locate a dataset that you want to explore in Search.
3. Select **Explore > Investigate in Search**.
   The search returns results in event list format by default. Switch the results format from **List** to **Table** to see the table view of the dataset.
4. (Optional) Update the search string with additional SPL. Do not remove the `from` reference.
5. (Optional) Click **Save as** to save your search, and select either **Report**, **Dashboard Panel**, or **Alert**.
6. (Optional) Click **New Table** to create a new table dataset based on the search string.
This option is only available in Splunk Cloud and Splunk Light, and in Splunk Enterprise with the Splunk Datasets Add-on installed.

**Edit datasets**

From the Datasets listing page you can access editing options for the different dataset types.

<table>
<thead>
<tr>
<th>Dataset Type</th>
<th>Select</th>
<th>Result</th>
<th>More info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Model</td>
<td>Manage &gt; Edit Data Model</td>
<td>Opens the Data Model Editor.</td>
<td>See Design data models.</td>
</tr>
<tr>
<td>Lookup Table</td>
<td>Manage &gt; Edit Lookup Table Files</td>
<td>Opens the Lookup table files listing page in Settings.</td>
<td>See About lookups.</td>
</tr>
<tr>
<td>Lookup Definition</td>
<td>Manage &gt; Edit Lookup Definition</td>
<td>Opens the detail page for the lookup definition from the Lookup definitions listing page in Settings.</td>
<td>See About lookups.</td>
</tr>
</tbody>
</table>

You cannot edit table datasets unless you use Splunk Cloud or Splunk Light, or you use Splunk Enterprise and have installed the Splunk Datasets Add-on.

See Manage table datasets.

**Manage dataset permissions**

Change dataset permissions to widen or restrict their availability to other users. You can set up read and write access by role, and you can make datasets globally accessible, restricted to a particular app context, or private to a single user.

By default, only the Power and Admin roles can set permissions for datasets.

**Lookup table files and lookup definitions**

1. On the Datasets listing page, identify a lookup table file or lookup definition that requires permission edits.
2. Select **Manage > Edit Permissions**.

For information about setting permissions for these dataset types, see Manage knowledge object permissions.
Lookup table files and lookup definitions are interdependent. Every CSV lookup definition includes a reference to a CSV lookup table file, and any CSV lookup table file can potentially be associated with multiple CSV lookup definitions. This means that each lookup table file must have permissions that are wider in scope or equal to the permissions of the lookup definitions that refer to it. For example, if your lookup table file is referenced by a lookup definition that is shared only to users of the Search app, that lookup table file must also be shared with users of the Search app, or it must be shared globally to all users. If the lookup table file is private, the lookup definition cannot connect to it, and the lookup will not work.

See About lookups and field actions.

**Data model datasets**

Permissions for data model datasets are set at the data model level. All datasets within a data model have the same permissions settings. There are two ways to set permissions for data models:

- Through the Data Model Editor
- Through the Data Models listing page in Settings

**Prerequisites**

- Manage knowledge object permissions
- Manage data models

**Steps for setting data model dataset permissions with the Data Model Editor**

1. In the Search & Reporting app, click **Datasets** to open the Datasets listing page.
2. Identify the data model dataset for which you want to update permissions.
3. Select **Manage > Edit data model**.
4. Select **Edit > Edit permissions** to set permissions for the data model that your selected data model dataset belongs to.
5. (Optional) Change the audience that you want the data model to **Display for**. It can display for users of a specific **App** or users of **All apps**.
6. (Optional) If the data model displays for an **App** or **All apps**, you can change the **Read** and **Write** settings that determine which roles can view or edit the data model.
7. Click **Save** or **Cancel**.
Steps for setting data model dataset permissions with the Data Models listing page in Settings

1. Select **Settings > Data models**.
2. Identify the data model for which you would like to change permissions.
3. Select **Edit > Edit permissions** to set permissions for the data model that your selected data model dataset belongs to.
4. (Optional) Change the audience that you want the data model to **Display for**. It can display for users of a specific **App** or users of **All apps**.
5. (Optional) If the data model displays for an **App** or **All apps**, you can change the **Read** and **Write** settings that determine which roles can view or edit the data model.
6. Click **Save** or **Cancel**.

**Share private lookup and data model datasets that you do not own**

If you want to share a private dataset that you do not own, you can change its permissions through the appropriate management page in Settings. You cannot see private datasets that you do not own in the Datasets listing page.

**Steps**

1. Select the Settings page for the type of data model that you are looking for, such as **Settings > Lookups > Lookup table files**.
2. Locate the dataset that you want to share and select **Edit > Edit Permissions**.
3. Share the dataset at the **App** or **All apps** level, and set read/write permissions as necessary.
4. Click **Save**.

When you return to the Datasets listing page you see that the dataset is visible and has the new permissions that you set for it.

**Delete datasets**

You can delete lookups and table datasets through the Datasets listing page. You can delete a data model dataset from the Data Model editor.

**Lookups and table datasets**

1. In the Search & Reporting app, click **Datasets** to open the Datasets listing page.
2. Locate a lookup or table dataset that you want to delete.
3. Select **Manage > Delete**.
4. On the **Delete Dataset** dialog, click **Delete** again to verify that you want to delete the dataset.

**Data model datasets**

1. In the Search & Reporting app, click **Datasets** to open the Datasets listing page.
2. Locate a data model dataset that you want to delete.
3. Select **Manage > Edit Dataset**.
4. In the Data Model Editor, click **Delete** for the data model dataset.

**Explore a dataset**

The Explorer view shows the contents of any **dataset** on the Datasets listing page. You can inspect the contents of any dataset listed on the page, including data model datasets and lookups.

The Explorer view provides several dataset exploration and management capabilities:

- Use two views for dataset exploration:
  - **View Results**, which renders the dataset in a standard table format.
  - **Summarize Fields**, which displays statistical information for each of the fields in your table and their values.
- Set the dataset time range.
- Manage the dataset search job.
- Export the contents of the dataset for a given time range.
- Extend your dataset as a scheduled report.

You can perform the same dataset management actions that you have access to through the Datasets listings page. See Manage datasets and Manage table datasets.

**Open the Explorer view for a dataset**

Use the Datasets listing page to access the Explorer view for a selected dataset.

1. In the Search & Reporting app, click **Datasets** to open the Datasets listing page.
2. Find a dataset you want to explore.
3. Click the dataset name to open it in the Explorer view.

Ways to view datasets

The Explorer view gives you two ways to view your dataset. You can View Results or you can Summarize Fields.

View Results

View Results is the default Explorer view. It displays your table dataset as a table, with fields as columns, values in cells, and sample events in rows. It displays the results of a search over the time range set by the time range picker.

Summarize Fields

Click Summarize Fields to see analytical details about the fields in the table. You can see top value distributions, null value percentages, numeric value statistics, and more. These statistics are returned by a search job that runs over the range defined by the time range picker. It is separate from the search job that populates the View Results display.
Set the dataset time range

The time range picker lets you restrict the data displayed by the view to events that fall within specific ranges of time. It applies to search-based dataset types like data model datasets and table datasets. The time range picker used in the Explorer view does not include options for real-time searches.

Lookup table files and lookup definitions get their data from static CSV files and KV store collections, so the time range picker does not apply to them. They display the same rows of data no matter what time range you select.

The time range picker is set to Last 24 hours by default. If your dataset has no results from the last 24 hours, this view is empty at first. You can adjust the time range picker to a range where events are present.

The time range picker gives you several time range definition options. You can choose a pre-set time range, or you can define a custom time range. For help with the time range picker, see Select time ranges to apply to your search in the Search Manual.

Manage the dataset search job

When you enter the Explorer view, a search job runs within the time range set by the time range picker. The search results populate the View Results display.

After you launch a dataset search, a set of controls at the top right of the dataset view lets you manage the search job without leaving the Explorer view. In the middle of this control set are pause/start and stop icons that you can use while the dataset search is in progress.
The Explorer job controls only manage the search job that produces the results displayed in View Results. They do not affect the job that runs when you open the Summarize Fields display of the dataset.

**Use the Job menu actions**

The Job menu lets you access the View Results search job, and information about it. You can use it when a search job is running, paused, or finalized.

1. Click Job.

2. Choose from the list options.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Job Settings...</td>
<td>Opens the Job Settings dialog box, where you can change the read permissions for the job, extend the job lifespan, and get a URL for the job. You can use the URL to share the job with others or to add a bookmark to the job in your Web browser.</td>
</tr>
<tr>
<td>Send Job to Background</td>
<td>Runs the job on the background. Use this option if the search job is slow to complete. This enables you to work on other activities, including running a new search job.</td>
</tr>
<tr>
<td>Inspect Job</td>
<td>Opens the <strong>Search Job Inspector</strong> window and displays information and metrics about the search job. You can select this action while the search is running or after the search completes. For more information, see View search job properties in the <em>Search Manual</em>.</td>
</tr>
</tbody>
</table>

For more information, see About jobs and job management in the *Search Manual*.

**Share a job**

Click the Share icon to share the View Results search job. When you select this, the lifetime of the job is extended to 7 days and its read permissions are set to Everyone. For more information about jobs, see About jobs and job management in the *Search Manual*.
**Export the job results**

Click the **Export** icon to export the results of the View Results search job. You can select to output to CSV, XML, or JSON and specify the number of results to export.

For information about other export methods, see Export search results in the *Search Manual.*

**Extend the dataset as a scheduled report**

You can **extend** your dataset to a new **scheduled report**. The report uses a `from` command in its base search to reference the dataset that you are viewing. Changes you make to the dataset are passed down to the report. Changes you make to the report are not passed up to the dataset.

Select **Manage > Schedule Report** to extend the dataset as a scheduled report. This opens the Schedule Report dialog box, where you can create the report schedule and define actions that are triggered each time the report runs. For example, you can arrange to have the Splunk software add the report results to a specific CSV file each time the report runs. You can also define scheduled report actions that email the results to a set of people, or that run scripts.

For more information about using this dialog box to create the report schedule and define actions for it, see Schedule reports, in the *Reporting Manual.*

For more information about dataset extension, see Extend datasets.

**Manage your dataset**

The Explorer view gives you the same dataset management capabilities as the Dataset listing page. If you review the contents of a dataset and decide you want to work with it, you do not need to return to the Dataset listing page. You can apply management actions to it from this view.

The Explorer view includes management actions for all dataset types:

- Visualize a dataset with Pivot
- Investigate a dataset with Search
- Edit a dataset
- Update dataset permissions
- Delete a dataset
For more information about these tasks, see Manage datasets.

If you have the Splunk Datasets Add-on installed, the Explorer view includes additional **table dataset** management capabilities:

- Extend a dataset as a new table dataset
- Clone a table dataset
- Edit table dataset descriptions
- Accelerate table datasets

For more information about these tasks, see Manage table datasets.
Create and edit table datasets

Table datasets and the Splunk Datasets Add-on

You can create and manage table datasets if you use Splunk Enterprise and have installed the Splunk Datasets Add-on, or if you use Splunk Cloud.

Table datasets, or tables, are a type of dataset that you can create, shape, and curate for a specific purpose. You begin by defining the initial data for the table, such as an index, source type, search string, or existing dataset. Then you edit and refine that table in the Table Editor until it fits the precise shape that you and your users require for later analysis and reporting work.

After you create your table, you can continue to iterate on it over time, or you can share it with others so they can refine it further. You can also use techniques like dataset cloning and dataset extension to create new datasets that are based on datasets you have already created.

You can manage table datasets alongside other dataset types that are available to all users of Splunk Enterprise and Splunk Cloud, like data model datasets and lookups. All of these dataset types appear in the Datasets listing page.

The Splunk Datasets Add-on is preinstalled for all users of Splunk Cloud.

Default datasets functionality for Splunk Enterprise users

This table explains what all Splunk Enterprise and Splunk Cloud users can do with datasets by default.

<table>
<thead>
<tr>
<th>Dataset activity</th>
<th>Why this is useful</th>
</tr>
</thead>
<tbody>
<tr>
<td>View dataset contents</td>
<td>Check a dataset to determine whether it contains fields and values that you want to work with. For example, you can view lookup table files directly instead of searching their contents in the Search view.</td>
</tr>
<tr>
<td>Open datasets in Pivot</td>
<td>With Pivot you can design a visualization-rich analytical report or dashboard panel that is based on your dataset. Pivot can also help you discover data trends and field correlations within a dataset.</td>
</tr>
</tbody>
</table>

276
Extend datasets in Search

Extend your dataset as a search, modify its search string as necessary, and save the search as a report, alert, or dashboard panel.

Next steps

- Read an overview of the of the dataset types that you can explore, manage, and create. See Dataset types and usage.
- Learn about the Datasets listing page. See View and manage datasets.
- Learn about dataset extension. See Dataset extension.

Additional dataset features provided by the Splunk Datasets Add-on

Splunk Enterprise users who install the Splunk Datasets Add-on gain the following datasets functionality. Cloud users get this functionality by default.

<table>
<thead>
<tr>
<th>Dataset activity</th>
<th>Why this is useful</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Table Editor to create tables</td>
<td>You can design sophisticated and tightly-focused collections of event data that fit specific business needs, even if you have minimal SPL skills.</td>
</tr>
<tr>
<td>Share and refine tables over time</td>
<td>After you create a table you can give other users read or write access to it so they can curate and refine it. For example, you can create a simple dataset, and then pass it to another user with deep knowledge of the source data to shape it for a specific use. You can also extend your dataset and let other people refine the extension without affecting the original dataset.</td>
</tr>
<tr>
<td>View field analytics</td>
<td>The Table Editor offers a Summarize Fields view that provides analytical information about the fields in your dataset. You can use this knowledge to determine what changes you need to make to the dataset to focus it to your needs.</td>
</tr>
<tr>
<td>Extend any dataset as a table</td>
<td>Dataset extension enables you to create tables that use the definition of any dataset type as their foundation. This enables you to create tables that are based on lookups and data model datasets and then modify those tables to fit your specific use cases.</td>
</tr>
<tr>
<td>Clone tables</td>
<td>You can make exact copies of table datasets and save the copy with a new name. Only table datasets can be cloned.</td>
</tr>
</tbody>
</table>
Accelerate tables

You can accelerate table datasets in a manner similar to report and data model acceleration. This can be helpful if you are using a very large dataset as the basis for a pivot report or dashboard panel. Once accelerated, the table returns results faster than it would otherwise.

Next steps

- Install the Splunk Datasets Add-on. See Install the Splunk Datasets Add-on in Install the Splunk Datasets Add-on.
- Define initial data for a new table dataset (requires add-on). See Define initial data for a new table dataset.
- Edit a table dataset (requires add-on). See Define initial data for a new table dataset.
- Accelerate a table dataset (requires add-on). See Accelerate tables.

Manage table datasets

This topic covers functions of the Datasets listing page that are available to users of Splunk Enterprise deployments with the Splunk Datasets Add-on installed. The functions involve the creation and management of table datasets.

Splunk Cloud and Splunk Light implementations have the Splunk Datasets Add-on installed by default.

For information about default features of the Datasets listing page, such as accessing Explorer views of datasets, investigating datasets in Search, and visualizing datasets with Pivot, see Manage datasets.

Open the Datasets listing page

In the Search app, click Datasets in the green Apps bar.

Create table datasets

To create a table dataset, click Create New Table Dataset. This takes you to the initial data definition workflow. After you define initial data you can edit the dataset in the Table Editor.

See Define initial data for a new table dataset and Use the Table Editor.
Extend a dataset as a new table dataset

You can extend any dataset to a new table dataset. This means you are creating a new dataset that is bound to the original dataset through its reference to that dataset. If the definition of the parent dataset changes, those changes are passed down to any child datasets that are extended from it.

To see from what datasets a table dataset is extended, expand its row in the Datasets listing page by clicking its > symbol in the first column. The parent datasets for that dataset are listed in an Extends line item. If a table dataset does not have an Extends line item, it is not extended from another dataset.

Prerequisites

- Dataset extension
- Use the Table Editor

Steps

1. On the Datasets listing page, find a dataset that you want to extend.
2. For that dataset, select Manage > Extend in Table.
3. (Optional) Use the Table Editor to modify the new table.
4. Click Save As to open the Save As New Table dialog.
5. Enter a Table Title.
6. Click Save to save the table.

Clone a table dataset

Clone a table dataset to make a new table dataset that is a copy of an existing dataset. Cloning differs from dataset extension in that you can make changes to the original dataset without affecting datasets that are cloned from that dataset.

Table datasets are the only dataset type that can be cloned through the Datasets listing page. You can clone lookup definition datasets through the Lookup Definitions page in Settings.

Steps

1. On the Datasets listing page, find a table dataset that you want to copy.
2. Select Clone.
3. Enter a Table Title.
4. (Optional) Enter a Description.
5. Click Clone Dataset.
6. (Optional) Click **Edit** to edit your cloned dataset.
7. (Optional) Click **Pivot** to open the cloned dataset in Pivot and create a visualization based on it.

**Edit a table dataset**

Use the Datasets listing page to edit selected table datasets. You can edit a table description, or you can edit the table. If you want to edit the table you can do so with the Table Editor.

**Add or edit a table dataset description**

Table dataset descriptions are visible in two places:

- The Dataset listing page, when you expand the table dataset row.
- The Explorer view of the table dataset, under the dataset name.

**Steps**

1. On the Datasets listing page, find a table dataset whose description you would like to add or edit. Expand the dataset row by clicking the > symbol in the first column to see its current description.
2. Select **Manage > Edit description**.
3. Add or update the description.
4. Click **Save**.

**Open a table dataset in the Table Editor**

Use the Datasets listing page to open a dataset in the Table Editor.

See Use the Table Editor.

**Steps**

1. On the Datasets listing page, find a table dataset that you want to edit.
2. Select **Manage > Edit table**.
3. Edit the table in the Table Editor.
4. Click **Save**.

**Update table dataset permissions**

New table datasets are private by default. They are available only to the users who created them. If you want other users to be able to view or edit a private
table dataset you can change its permissions.

By default, only the Power and Admin roles can set permissions for table datasets.

On the Datasets listing page, select Manage > Edit Permissions for the table whose permissions you want to edit. You can see whether the table is shared with users of a specific app, or globally with users of all apps. You can also see which roles have read or write access to the app.

For information about setting permissions, see Manage knowledge object permissions.

**Share private table datasets that you do not own**

If you want to share a private table dataset that you do not own, you can change its permissions though the Data Models management page in Settings. You cannot see private datasets that you do not own in the Datasets listing page.

**Steps**

1. Select Settings > Data models.
2. Locate the table dataset that you want to share and select Edit > Edit Permissions.
3. Share the dataset at the App or All apps level, and set read/write permissions as necessary.
4. Click Save.

When you return to the Datasets listing page you will see that the dataset is visible and has the new permissions that you set for it.

**Manage table dataset acceleration**

Table datasets that contain a large amount of data can be accelerated so that their underlying search completes faster when you view the dataset or visualizations that are backed by it. Table acceleration only applies to table datasets when the tstats or pivot commands are applied to them.

On the Datasets listing page, select Manage > Edit Acceleration for the table you want to accelerate.

On the datasets listing page, accelerated table datasets and data model datasets have a icon.
Delete a table dataset

You can delete any table dataset that you have write permissions for.

Before deleting a table dataset, verify that it has not been extended to one or more child table datasets. Deletion of a parent dataset breaks tables and other objects that are extended from it. For example, if table Alpha is extended to table Beta, and table Beta is in turn used to create a Pivot visualization that is used in a dashboard panel, that dashboard panel will cease to function if you delete table Alpha.

See Dataset Extension

Prerequisites

See Disable or delete knowledge objects.

Steps

1. On the Datasets listing page, find the table dataset that you want to delete.
2. Select Manage > Delete.
3. Click Delete again to confirm.

Define initial data for a new table dataset

When you create a new table dataset with the Table Editor you start by defining initial data. You have three options for initial data.

- **An index and source type combination** - You can populate your new dataset with events associated with a combination of indexes and source types.
- **An existing dataset** - Your dataset can get its initial data from a dataset that already exists. The dataset can be a table dataset, a data model dataset, a CSV lookup table, or a CSV lookup definition.
- **A search** - You can base your dataset on the results of any search string, as long as it does not include transforming commands.
If you use Splunk Analytics for Hadoop and want to create a dataset based on data from a virtual index, you must get your initial data either from a search that references the virtual index or from an existing dataset that already has the virtual index data.

**Identify an index and source type combination for initial data**

1. In the Search & Reporting app, open the Datasets listing page.
2. Click **Create New Table Dataset** to go to the initial data setup screen of the Table Editor.
3. Select **Indexes & Source Types**.
4. Choose an index that you want to use for initial data. If you do not want to select a specific index, select **All indexes**.
5. Select a source type that you want to use for initial data. If you do not want to select a specific source type, select **All source types**.

If you select both **All indexes** and **All source types**, you risk creating an overly broad dataset that contains all of the events indexed by your Splunk implementation (with the exception of events in `_internal` and other internal indexes, which you must specify by name). In general you should avoid creating overly broad datasets. The datasets feature is designed for creating narrow views of data.

A preview of your dataset appears. Rows are events, columns are fields, and cells are field values.

6. (Optional) Click **Add an index and one or more source types...** to create a dataset that pulls data from more than one index and source type combination.
7. Select existing fields that you want to see in your dataset. Click **OK** when you are done.

Hover over a listed field to see field statistics, such as the percentage of events in the dataset that have the field, and the top values for the field.

8. (Optional) If you are not seeing a field choice that you are expecting, add the missing field.
   - At the bottom of the field list, click **Add a missing existing field**.
   - Enter the field and click **Add**.
   - Select the added field.

9. Use the dataset preview pane to verify that this is the initial data that you want. If you do not find the existing fields or field values that you were expecting you can remove this selection and select another one.
10. (Optional) If you are not sure whether the index and source type combination you have chosen contains the events you are looking for,
change the Sample setting at the top of the preview pane to see random events from the dataset or select a new sample.

11. When you are satisfied that your index, source type, and field selections provide the correct initial data for your dataset, click Done to move on to the Table Dataset Editor.

**Use an existing dataset for initial data**

The Datasets tab lets you select an existing dataset for your initial data. You can select any dataset that you can otherwise see on the Datasets listing page, including data model datasets, lookup tables, and lookup definitions.

When you create a dataset that uses an existing dataset for initial data, you can choose between cloning and extending the existing dataset.

**Prerequisites**

- See About datasets.
- See Dataset extension.

**Steps**

1. In the Search & Reporting app, open the Datasets listing page.
2. Click Create New Table Dataset to go to the initial data setup screen of the Table Editor.
3. Select Existing Datasets.
4. Select either Clone or Extend.

<table>
<thead>
<tr>
<th>Selection</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Creates an identical copy of the original dataset. Only table datasets can be cloned.</td>
</tr>
<tr>
<td>Extend</td>
<td>Creates a dataset that is extended from an existing dataset. Changes made to the original dataset propagate down to the extended dataset. All dataset types can be extended.</td>
</tr>
</tbody>
</table>

5. If you are working with a lookup table file, select the fields that you want to use in your table.

The fields you select are the only fields that will make up your dataset, along with _raw and _time, which are required. You can hover over a field to see field statistics, such as the percentage of events in the dataset that have the field, and the top values for the field.

Table datasets, data model datasets, and lookup definitions have fixed
fields. When you create a new dataset by cloning or extending a dataset with fixed fields, you do not get to choose which of those fields you want to start with in your dataset.

6. (Optional) If you are not seeing a field choice that you are expecting, add the missing field.
   - At the bottom of the field list, click **Add a missing existing field**.
   - Enter the field and click **Add**.
   - Select the added field.

7. Use the dataset preview pane to verify that this is the initial data that you want. If you do not find the existing fields or field values that you were expecting you can select a different dataset.

8. (Optional) If you are not sure whether the dataset you have chosen contains the events you are looking for, change the **Sample** setting at the top of the preview pane to see random events from the dataset or select a new sample.

9. When you are satisfied that your dataset selection provides the correct initial data for your dataset, click **Done** to move on to the Table Dataset Editor.

**Provide a search string for initial data**

There are four methods that you can follow to derive the search string for initial data. Once you provide the search string, the other initial data setup steps are the same.

The search string you provide must identify the fields that its search commands operate on. For example, a search that only includes commands like `sendemail`, `highlight`, or `delete` will be invalid because those commands do not require that you identify the fields that they operate upon.

**Provide the full search string in the Table Editor**

1. In the Search & Reporting app, open the Datasets listing page.
2. Click **Create New Table Dataset** to go to the initial data setup screen of the Table Editor.
3. Click **Search (Advanced)**.
4. Provide a search string that returns data you want in your table.
5. Continue the initial data definition process by following the steps in Preview your dataset and select its starting fields.
**Use a search string you have designed in the Search view**

1. In the Search view, design a search that returns events that you want in your table.
2. Click **New Table** to use the search as the initial data for a new table dataset.
   The Table Editor opens with **Search (Advanced)** selected and the search string you designed in the search field.
3. (Optional) Add additional SPL until you have a search that returns results that you would like to use in a dataset.
4. Continue the initial data definition process by following the steps in Preview your dataset and select its starting fields.

**Start with a search string that includes your index and source type selections**

1. In the Search & Reporting app, open the Datasets listing page.
2. Click **Create New Table Dataset** to go to the initial data setup screen of the Table Editor.
3. Select **Indexes & Source Types**.
4. Choose an index that you want to use for initial data. If you do not want to select a specific index, select **All indexes**.
5. Choose a source type that you want to use for initial data.
6. (Optional) Click **Add an index and one or more source types...** to create a dataset that pulls data from more than one index and source type combination.
7. Click **Search (Advanced)**. The search field populates with the index and source type combination that you have selected.
8. (Optional) Add additional SPL until you have a search that returns results that you would like to use in a dataset.
9. Continue the initial data definition process by following the steps in Preview your dataset and select its starting fields.

**Start with a search string that extends an existing dataset**

This method creates a dataset that is **extended** from an existing dataset. Changes made to the original dataset propagate down to the extended dataset. All dataset types can be extended.

1. In the Search & Reporting app, open the Datasets listing page.
2. Click **Create New Table Dataset** to go to the initial data setup screen of the Table Editor.
3. Select **Existing Datasets**.
4. Select **Extend**.

5. Select a dataset that you want to use for initial data. If you have a significant number of datasets to choose from, click the magnifying glass to search for the dataset you want.

6. Click **Search (Advanced)**. The search field populates with SPL referencing your selected dataset.

7. (Optional) Select the fields you would like to see in your dataset. You can select fields whether or not the original dataset type has fixed fields.

8. (Optional) Add additional SPL until you have a search that returns results that you would like to use in a dataset.

9. Continue the initial data definition process by following the steps in Preview your dataset and select its starting fields.

**Preview your dataset and select its starting fields**

When you begin this task, you must have first used one of the previous four tasks to define the search string for your initial data.

1. After you define the search string for your initial data, press the **Enter** key on your keyboard or click the magnifying glass icon to run the search.

   A preview of your dataset appears. Rows are events, columns are fields, and cells are field values. Update the search and run it again until you are satisfied with the results.

2. Select existing fields that you want to see in your dataset. Click **OK** when you are done.

   Hover over a listed field to see field statistics, such as the percentage of events in the dataset that have the field, and the top values for the field.

3. (Optional) If you are not seeing a field choice that you are expecting, add the missing field.
   
   ♦ At the bottom of the field list, click **Add a missing existing field**.
   
   ♦ Enter the field and click **Add**.
   
   ♦ Select the added field.

4. Use the dataset preview pane to verify that this is the initial data that you want. If you do not find the existing fields or field values that you were expecting you can modify the search.

5. (Optional) If you are not sure whether your search string will return the events you are looking for, change the **Sample** setting at the top of the preview pane to see random events from the dataset or select a new sample.

6. When you are satisfied that your index, source type, and field selections provide the correct initial data for your dataset, click **Done** to move on to
Use the Table Editor

After you define the initial data for your table dataset, you use the Table Editor to refine it and maintain it. You also use the Table Editor to make changes to existing tables.

The Table Editor includes several table dataset editing tools:

- Work with your table in two views:
  - **Preview Rows**, which renders the dataset in a standard table format.
  - **Summarize Fields**, which displays statistical information for each of the fields in your table and their values.
- Apply actions to the table that filter events, add fields, edit field names and field values, perform statistical data aggregations, and more. You can apply actions through menu selections, or by making edits directly to table elements.
- Use a command history feature to review, edit, and delete actions that were applied to the table.

Get to the Table Editor

There are three ways to get to the Table Editor.

<table>
<thead>
<tr>
<th>Method</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>When you define initial data for a new table dataset</td>
<td>See Define initial data for a new table dataset.</td>
</tr>
<tr>
<td>When you edit an existing table dataset.</td>
<td>See Manage table datasets.</td>
</tr>
<tr>
<td>When you extend a an existing dataset as a new table dataset</td>
<td>See Manage table datasets.</td>
</tr>
</tbody>
</table>

See Define initial data for a new table dataset if you need help with this step of the dataset creation workflow.

Table Editor views

The Table Editor allows you to edit your table in two views. These views are named Preview Rows and Summarize Fields.
The Preview Rows view

Preview Rows is the default Table Editor view. It displays your table dataset as a table, with fields as columns, values in cells, and sample events in rows. It displays a sample 50 events from your dataset. It does not represent the results from any particular time range.

You can edit your table by applying actions to it, either by making menu selections, or by making edits directly to the table.

In the context of the Table Editor, the Preview Rows view is not designed to be an editing tool rather than a search tool. It does not provide a time range picker. If you would like to see a table-formatted set of results from a specific time range, save your table and go to the Datasets listing page to open it in the Explorer view. In the Explorer view, View Results displays results from a search over a time range that you can define.

See Explore a dataset.

Alternatively, you can switch to the Summarize Fields view of the Table Editor. It has a time range picker that lets you view field statistics for specific time ranges.

The Summarize Fields view

Click Summarize Fields to see analytical details about the fields in the table. You can see top value distributions, null value percentages, numeric value statistics, and more.
You can apply some menu actions and commands to your table while you are in the Summarize fields view. You can also apply actions through direct edits, such as moving columns, renaming fields, fixing field type mismatches, and editing field values.

**Using the time range picker**

When you are in the Summarize Fields view you can view field analytics for a specific range of time. The *time range picker* is near the top right side of the display.

The time range picker shows events from the last 24 hours by default. If your dataset has no events from the last 24 hours it will have no statistics when you open this view. To fix this, adjust the time range picker to a range where events are present.

The time range picker gives you a variety of time range definition options. You can choose a pre-set time range, or you can define a custom time range. For help with the time range picker, see Select time ranges to apply to your search in the *Search Manual*.

**Table element selection options**

Availability of menu actions depends on the table elements that you select. For example, some actions are only available when you select a field column.

You have the same selection options in the Preview Rows and Summarize Fields views.

<table>
<thead>
<tr>
<th>Element</th>
<th>Applies action to</th>
<th>How to select</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

290
### Table

<table>
<thead>
<tr>
<th>Table</th>
<th>Entire dataset</th>
<th>Click the asterisk header at the top of the leftmost column.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column</td>
<td>A field</td>
<td>Click on a column header.</td>
</tr>
</tbody>
</table>
| Multi-Column| Two or more fields                                 | • To select multiple nonadjacent columns, hold the CTRL or CMD key and click the header row of each column you wish to select. Deselect columns by clicking on them while holding CTRL or CMD.  
• To select a range of adjacent columns, click the header row of the first column, hold SHIFT, and click the header row of the last column. |
| Cell        | A field value                                      | Click a cell.                                                |
| Text        | A portion of text within a field value.            | Click and drag to select text. You can select text for text and IPv4 field types. |

### Field types

Each field belongs to a type. There are five field types.

Some actions and commands can only be applied to fields of specific types. For example, you can apply the **Round Values** and **Map Ranges** actions only to numeric fields.

<table>
<thead>
<tr>
<th>Type</th>
<th>Icon</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td></td>
<td>A field whose values are text strings. It can include a mix of text and numbers.</td>
</tr>
<tr>
<td>Number</td>
<td></td>
<td>A field whose values are purely numerical. Does not include IPv4 addresses.</td>
</tr>
<tr>
<td>Boolean</td>
<td></td>
<td>A field whose values are either true or false. Alternate value pairs such as 1 and 0 or Yes and No can also be used.</td>
</tr>
<tr>
<td>IPv4</td>
<td></td>
<td>A field whose value is an IPv4 address such as 192.0.2.1.</td>
</tr>
<tr>
<td>Epoch Time</td>
<td></td>
<td>A field whose value is a timestamp.</td>
</tr>
</tbody>
</table>

The Table Editor automatically assigns types to fields when you define initial data for a dataset. It can also assign types to fields when you add fields to those
datasets. If a field is assigned the wrong type, you can change the type either by direct table edit, or by using the **Edit** action menu.

See Apply actions through direct table edits.

**Apply actions through menu selections**

You can apply actions to your table or elements of your table by making selections from the action menus just above it. Many of these actions can only be performed while you are in the Preview Rows view, but some can be performed in either view.

**Action menus**

The actions and commands that you can apply to your table are categorized into the following menus.

<table>
<thead>
<tr>
<th>Menu</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit</td>
<td>Contains basic editorial actions. Change field types, rename fields, move and delete fields.</td>
</tr>
<tr>
<td>Sort</td>
<td>Sort rows by the values of a selected field.</td>
</tr>
<tr>
<td>Filter</td>
<td>Provides actions that let you filter rows out of your dataset.</td>
</tr>
<tr>
<td>Clean</td>
<td>Features actions that fix or change field values.</td>
</tr>
<tr>
<td>Summarize</td>
<td>Perform statistical aggregations on your dataset.</td>
</tr>
<tr>
<td>Add new</td>
<td>Gives you different ways to add fields to your dataset.</td>
</tr>
</tbody>
</table>

**Apply actions through direct table edits**

You can make edits to your table dataset by clicking on it. Move field columns, change field names, replace field values, and fix field type mismatches.

The following steps apply to both Table Editor views.
### Move a field column

You can drag-and-drop field columns to new positions in your table.

1. Select the column that you want to move.
2. Click on the column header cell and drag the column to a new location in your table. When you do this, the column header cell moves with your mouse cursor.
3. Release the mouse button to drop the column in its new location.

This action is not recorded by the Table Editor in the command history sidebar.

### Change a field name

You can change a field name by double-clicking on it.

1. Double-click on the column header cell that contains the name of the field that you want to change.
2. Enter the new field name. Field names cannot be blank, start with an underscore, or contain quotes, backslashes, or spaces.
3. Click outside of the cell to complete the field name change.

The Table Editor records this change in the command history sidebar as a Rename field action.

### Replace field values

Select a field value and replace every instance of it in its column with a new value. For example, if your dataset has an action field with a value of `addtocart`, you can replace that value with `add to cart`.

You can use this method to fill null or empty field values.

You cannot make field value replacements on an event by event basis. When you use this method to replace a value in one event in your dataset, that value is changed for that field throughout your dataset.

For example, if you have an event where the city field has a value of `New York`, you cannot change that value to `Los Angeles` just for that one event. If you change it to `Los Angeles`, every instance of `New York` in the city column also changes to `Los Angeles`.

293
1. Double-click on a cell that contains the field value that you want to change.
2. Edit the value or replace it entirely.
3. Click outside of the cell to complete the field replacement. Every instance of the field value in the field's column will be changed.

The Table Editor records this change in the command history sidebar as a Replace value action.

Fix field type mismatches

Sometimes fields have type mismatches. For example, a string field that has a lot of values with numbers in them might be mistyped as a numeric field. You can give a field the correct type by clicking on the type symbol in its column header cell.

You cannot change the type of the _time or _raw fields.

1. Find the column header cell of the mistyped field and hover over its type icon. The cursor changes to a pointing finger.
2. Click on the type icon.
3. Select the type that is most appropriate for the field.

This action is not recorded by the Table Editor in the command history sidebar.

Use the command history sidebar

The command history sidebar keeps track of the commands you apply as you apply them. You can click on a command record to reopen its command editor and change the values entered there.

When you click on a command that is not the most recent command applied, the Table Editor shows you how the table looked at that point in the command history.

You can edit the details of any command record in the command history. You can also delete any command in the history by clicking the X on its record. When you edit or delete a command record, you potentially can break commands that follow it. If this happens, the command history sidebar will notify you.

When you edit or delete a command that is not the most recent command applied, you can break commands that follow it. If this happens the command history sidebar will notify you.
Click SPL to see the search processing language behind your commands. When you have SPL selected you can click Open in Search to run a search using this SPL in the Search & Reporting view.

Save a new table

When you finish editing a table dataset you can click Save As to save it as a new table dataset.

Prerequisites

- Do not create table datasets with the same name

Steps

1. Click Save As to save your table.
2. Give your dataset a unique Name.
3. (Optional) Enter or update the Table ID. This value can only contain letters, numbers and underscores. It cannot be changed later.
4. (Optional) Add a dataset Description. Table dataset descriptions are visible in two places:
   - The Dataset listing page, when you expand the table dataset row.
   - The Explorer view of the table dataset, under the dataset name.
   You can edit the description through the Datasets page or the Explorer view by selecting Manage > Edit description.
5. Click Save to save your changes.

After you save a new table, you can choose one of three options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continue Editing</td>
<td>Returns you to the Table Editor, where you can keep editing the dataset.</td>
</tr>
<tr>
<td>Explore Dataset</td>
<td>Opens the dataset in the Explorer view.</td>
</tr>
<tr>
<td>Done</td>
<td>Takes you to the Datasets listing page.</td>
</tr>
</tbody>
</table>

Do not create table datasets with the same name

When you create table datasets, always give them unique names. If you have more than one table dataset with the same name in your system you risk experiencing object name collision issues that are difficult to resolve.
For example, say you have two table datasets named Store Sales, and you share one at the global level, but leave the other one private. If you then extend the global Store Sales dataset, the dataset that is created through that extension will display the table from the private Store Sales dataset instead.

**Dataset extension**

Dataset extension is a way to create a search, report, dataset, or other object that is built upon a reference to an existing dataset. This reference means that the object always refers to the original dataset for its foundational data. If the definition of the original dataset changes, those changes are passed down to any datasets that extend it.

Dataset extension is not the same as dataset cloning. When you clone a dataset, you create a distinct, individual dataset that is identical to the original dataset but not otherwise connected to it. When you extend a dataset, you create a dataset, report, dashboard panel, or alert that is bound to the original dataset through its reference to that dataset.

**Example of extending a dataset as a report**

For example, you have a dataset named Alpha. If you select *Explore > Investigate in Search* on the Datasets listing page for the Alpha dataset, you go to the Search view and run a search that displays the contents of Alpha. This search string uses the `from` command to reference Alpha. You can optionally modify the search string with additional Splunk Search Processing Language (SPL).

If you save this search string as a report named Beta, it will still have the reference back to Alpha. This means that if someone decides to make a change to Alpha, that change cascades down to the Beta report. This might cause problems in the Beta report.

For example, you might modify the search string of the Beta report with lookups and eval expressions that use fields passed down from the Alpha dataset in their definitions. If someone deletes those fields from the Alpha dataset, those lookups and eval expressions will break in the Beta report, because they require fields that no longer exist.
Dataset extension chains

If you have the Splunk Datasets Add-on installed, you can extend any dataset as a table dataset. This means that you can have chains of extended datasets. For example you can extend Dataset Alpha as dataset Beta, and then extend dataset Beta as dataset Gamma, and so on. Any change to Alpha will propagate down through the other datasets in the chain.

The Splunk Datasets Add-on enables you to understand dataset extension chains from the end of the chain, but not from the start. So to use the example in the preceding paragraph, if you are on dataset Gamma, you can see that it extends Beta, which in turn extends Alpha. But if you are looking at Alpha, you have no way of knowing which datasets were extended from it.

To learn which datasets a dataset extends

Locate the dataset in the Datasets listing page and expand its row. If it extends one or more datasets, you will find an Extends line item with the extended datasets listed from top to bottom. For example, here is the detail information for Gamma, showing that it extends Alpha and Beta.

You can also find this information on the viewing page for a dataset. Click More Info to see what datasets the dataset that you are viewing extends.

Use a naming convention for extended datasets

When you are working with a dataset, it is difficult to know what datasets are extended from it. For example, a person working with the Alpha dataset has no way of knowing that it is extended by the Beta and Gamma datasets.

You can manage this by using a naming convention to indicate when a dataset is extended from another. For example, if you extend a dataset from dataset Alpha, you can name it Alpha.Beta. Later, if you extend two datasets from Alpha.Beta, you can name those datasets Alpha.Beta.Gamma and Alpha.Beta.Epsilon. This naming methodology is similar to that of data model datasets, where the dataset name indicates where it lives in a greater hierarchy of data model datasets.
When you extend a dataset you can update its description to indicate that it has been extended. Identify the knowledge objects that have been directly extended from it, not the full extension chain, if one exists. Add a sentence like this to the dataset description: "This dataset has been extended as a table dataset named <dataset_name> and a report named <report_name>."

**The from command**

Dataset extension is facilitated by the `from` command, whether you extend it by opening it in the Search view, or through the Table Editor.

When you open a dataset in the Search view, you see a search string that uses the `from` command to retrieve data from that dataset. For example, say you have a dataset named Buttercup_Games_Purchases. If, while on the Datasets listing page, you click **Explore in Search** for that dataset, the Splunk platform takes you to the Search view, where you see this search string:

```
| from datamodel:"Buttercup_Games_Purchases"
```

If you work with Splunk Cloud, or work with Splunk Enterprise and have installed the Splunk Datasets Add-on, you can extend any dataset as a table dataset. When you do this, the Table Editor uses the `from` command in the background. Click the **SPL** toggle in the command history sidebar to see how the Table Editor uses the `from` command.

This closeup of the command history sidebar in the Table Editor shows that the initial data for the table dataset is provided by a `from` command extension of the Buttercup Games Purchases dataset.
For more information, see `from` in the *Search Reference*.

**Extension and table acceleration**

If you want to accelerate a table that extends other tables, it needs to be shared, and the tables it extends must be shared as well.

You will not see acceleration benefits when you use `from` to extend an accelerated table.

You cannot accelerate a table that is extended from a lookup table file or lookup definition. Acceleration can only be applied to datasets that use purely **streaming commands**. Lookup dataset extension is not a streaming operation.

See Accelerate tables.

**Accelerate tables**

If you have a table dataset that contains a large amount of data, you can accelerate it so that searches, reports, and dashboard panels that use or extend it return results faster than they would otherwise.

With table acceleration, the Splunk software treats each table dataset as if it were a **data model** made up of a single root search **data model dataset**.

See Design data models for information about root search data model datasets.

**Things to know about table acceleration**

Before you accelerate your table datasets, there are some restrictions and best
practices to be aware of.

**The benefits of acceleration are only available to tables when you apply the `tstats` or `pivot` commands to them**

You see the acceleration benefits when you run a search that uses the `tstats` or `pivot` commands to reference a table. You also see acceleration benefits when you use the Pivot editor to create a report or dashboard panel that uses an accelerated table. You do not see acceleration benefits when you use a command such as `from` to reference an accelerated table.

**By default, only users whose roles have the `accelerate_datamodel` capability can accelerate table datasets**

Table acceleration can be resource-intensive, so it should be used conservatively by a limited number of Splunk users.

**You cannot enable acceleration for private tables**

You must share a table to make it eligible for acceleration. You must also share related knowledge objects, such as lookup tables and lookup definitions that your lookup fields are dependent upon, and in exactly the same way.

**You can apply table acceleration only to tables that use purely streaming search commands**

If you use the action menus to apply the `Sort`, `Limit Rows`, `Remove Duplicates` or `Stats` actions to your table, you cannot accelerate it.

**You cannot accelerate a table that is extended from a lookup file or lookup definition**

Lookup dataset extension involves search operations that are not streaming

This disqualifies it from being accelerated in the same way that non-streaming commands do. See Dataset extension.

**If you want to accelerate a table that is extended from other tables, you must share those tables as well**

The parent table or tables that a child table is extended from must be shared before you can accelerate the child table.
If you edit an accelerated table, the Splunk software rebuilds its acceleration summary when you save your changes.

When you change a dataset definition, its summary becomes invalid and must be replaced.

Table acceleration is most efficient if the table being accelerated specifies the indexes to be searched in its initial data search.

If you do not specify an index, the Splunk software searches all available indexes for the table and can create unnecessarily large acceleration summaries.

For details about how table acceleration works and tips on managing table acceleration summaries, see Accelerate data models.

**Accelerate a table dataset**

Access the table dataset acceleration settings through the Datasets listing page or the Explorer view of a dataset.

**Steps**

1. Select **Manage > Edit Acceleration** for the dataset you want to accelerate.
2. Select **Accelerate**.
3. Select a **Summary Range**.
   - Your choice depends on the range of time over which you plan to run searches, reports, or dashboard panels that use the accelerated table. For example, if you plan to run dashboard panels using the table over periods of time that fall within the previous seven days, choose **7 Days**.
   - If you require a different summary range than the ones supplied by the Summary Range field, you can configure it for your table in datamodels.conf.
4. Click **Save**.
   - When your table is accelerated, the symbol for the table has a yellow color.

**Inspect table acceleration metrics**

After you accelerate a table you can find its acceleration metrics on the Data Models management page. Expand the row for the accelerated table and review the information that appears under **ACCELERATION**.
<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Tells you whether the acceleration summary for the table is complete. When the summary is in <strong>Building</strong> status you also see what percentage of the summary is complete. Many table summaries are constantly updating with new data. This means that a summary that is <strong>Complete</strong> at one moment may be <strong>Building</strong> later.</td>
</tr>
<tr>
<td>Access Count</td>
<td>Shows you how many times the table summary has been accessed since it was created, and when the last access time was. Useful when you are trying to determine which accelerated tables are not being used frequently. Because table acceleration uses system resources, you might not want to accelerate tables that are not regularly accessed.</td>
</tr>
<tr>
<td>Size on Disk</td>
<td>Shows you how much disk space the table acceleration summary uses. Use this metric along with the <strong>Access Count</strong> to determine which summaries are unnecessary and can be deleted. If a table acceleration summary is using a large amount of disk space, consider reducing its summary range.</td>
</tr>
<tr>
<td>Summary Range</td>
<td>Presents the range of the table acceleration summary, in seconds, always relative to the present moment. You set this range when you enable acceleration for the table.</td>
</tr>
<tr>
<td>Buckets</td>
<td>Displays the number of index <strong>buckets</strong> spanned by the table acceleration summary.</td>
</tr>
</tbody>
</table>

Click **Rebuild** to rebuild the summary. You might want to do this if you suspect that there has been data loss due to a system crash or a similar mishap. Splunk Enterprise rebuilds summaries when you edit a table, or when you disable and reenable table acceleration.

Click **Update** to refresh the acceleration summary detail information.

Click **Edit** to open the Edit Acceleration dialog box to change the **Summary Range**, or to disable acceleration for the table.
**Accelerating data model datasets**

Data model datasets are accelerated at the data model level. You can access the Data Models management page by selecting **Settings > Data Models**.

For more information, see Accelerate data models.
Build a data model

About data models

Data models drive the pivot tool. Data models enable users of Pivot to create compelling reports and dashboards without designing the searches that generate them. Data models can have other uses, especially for Splunk app developers.

Splunk knowledge managers design and maintain data models. These knowledge managers understand the format and semantics of their indexed data and are familiar with the Splunk search language. In building a typical data model, knowledge managers use knowledge object types such as lookups, transactions, search-time field extractions, and calculated fields.

What is a data model?

A data model is a hierarchically structured search-time mapping of semantic knowledge about one or more datasets. It encodes the domain knowledge necessary to build a variety of specialized searches of those datasets. These specialized searches are used by Splunk software to generate reports for Pivot users.

When a Pivot user designs a pivot report, she selects the data model that represents the category of event data that she wants to work with, such as Web Intelligence or Email Logs. Then she selects a dataset within that data model that represents the specific dataset on which she wants to report. Data models are composed of datasets, which can be arranged in hierarchical structures of parent and child datasets. Each child dataset represents a subset of the dataset covered by its parent dataset.

If you are familiar with relational database design, think of data models as analogs to database schemas. When you plug them into the Pivot Editor, they let you generate statistical tables, charts, and visualizations based on column and row configurations that you select.

To create an effective data model, you must understand your data sources and your data semantics. This information can affect your data model architecture--the manner in which the datasets that make up the data model are organized.
For example, if your dataset is based on the contents of a table-based data format, such as a .csv file, the resulting data model is flat, with a single top-level root dataset that encapsulates the fields represented by the columns of the table. The root dataset may have child dataset beneath it. But these child dataset do not contain additional fields beyond the set of fields that the child datasets inherit from the root dataset.

Meanwhile, a data model derived from a heterogeneous system log might have several root datasets (events, searches, and transactions). Each of these root datasets can be the first dataset in a hierarchy of datasets with nested parent and child relationships. Each child dataset in a dataset hierarchy can have new fields in addition to the fields they inherit from ancestor datasets.

Data model datasets can get their fields from custom field extractions that you have defined. Data model datasets can get additional fields at search time through regular-expression-based field extractions, lookups, and eval expressions.

The fields that data models use are divided into the categories described above (auto-extracted, eval expression, regular expression) and more (lookup, geo IP). See Dataset fields.

Data models are a category of knowledge object and are fully permissionable. A data model's permissions cover all of its data model datasets.

See Manage data models.

**Data models generate searches**

When you consider what data models are and how they work it can also be helpful to think of them as a collection of structured information that generates different kinds of searches. Each dataset within a data model can be used to generate a search that returns a particular dataset.

We go into more detail about this relationship between data models, data model datasets, and searches in the following subsections.

- **Dataset constraints** determine the first part of the search through:
  - Simple search filters (Root event datasets and all child datasets).
  - Complex search strings (Root search datasets).
  - transaction definitions (Root transaction datasets).
- When you select a dataset for Pivot, the unhidden fields you define for that dataset comprise the list of fields that you choose from in Pivot when you
decide what you want to report on. The fields you select are added to the search that the dataset generates. The fields can include calculated fields, user-defined field extractions, and fields added to your data by lookups.

The last parts of the dataset-generated-search are determined by your Pivot Editor selections. They add transforming commands to the search that aggregate the results as a statistical table. This table is then used by Pivot as the basis for charts and other types of visualizations.

For more information about how you use the Pivot Editor to create pivot tables, charts, and visualizations that are based on data model datasets, see Introduction to Pivot in the Pivot Manual.

Datasets

Data models are composed of one or more datasets. Here are some basic facts about data model datasets:

- **Each data model dataset corresponds to a set of data in an index.** You can apply data models to different indexes and get different datasets.
- **Datasets break down into four types.** These types are: Event datasets, search datasets, transaction datasets, and child datasets.
- **Datasets are hierarchical.** Datasets in data models can be arranged hierarchically in parent/child relationships. The top-level event, search, and transaction datasets in data models are collectively referred to as "root datasets."
- **Child datasets have inheritance.** Data model datasets are defined by characteristics that mostly break down into constraints and fields. Child datasets inherit constraints and fields from their parent datasets and have additional constraints and fields of their own.

We'll dive into more detail about these and other aspects of data model datasets in the following subsections.

- **Child datasets provide a way of filtering events from parent datasets**
  - Because a child dataset always provides an additional constraint on top of the constraints it has inherited from its parent dataset, the dataset it represents is always a subset of the dataset that its parent represents.


**Root datasets and data model dataset types**

The top-level datasets in data models are called root datasets. Data models can contain multiple root datasets of various types, and each of these root datasets can be a parent to more child datasets. This association of base and child datasets is a dataset tree. The overall set of data represented by a dataset tree is selected first by its root dataset and then refined and extended by its child datasets.

Root datasets can be defined by a search constraint, a search, or a transaction:

- **Root event datasets** are the most commonly-used type of root data model dataset. Each root event dataset broadly represents a type of event. For example, an *HTTP Access* root event dataset could correspond to access log events, while an *Error* event corresponds to events with error messages.

  Root event datasets are typically defined by a simple constraint. This constraint is what an experienced Splunk user might think of as the first portion of a search, before the pipe character, commands, and arguments are applied. For example, `status > 600 and sourcetype=access_* OR sourcetype=iis*` are possible event dataset definitions.

  See Dataset Constraints.

- **Root search datasets** use an arbitrary Splunk search to define the dataset that it represents. If you want to define a base dataset that includes one or more fields that aggregate over the entire dataset, you might need to use a root search dataset that has transforming commands in its search. For example: a system security dataset that has various system intrusion events broken out by category over time.

- **Root transaction datasets** let you create data models that represent transactions: groups of related events that span time. Transaction dataset definitions utilize fields that have already been added to the model via event or search dataset, which means that you can't create data models that are composed only of transaction datasets and their child datasets. Before you create a transaction dataset you must already have some event or search dataset trees in your model.

Child datasets of all three root dataset types--event, transaction, and search--are defined with simple constraints that narrow down the set of data that they inherit from their ancestor datasets.
Dataset types and data model acceleration

You can optionally use data model acceleration to speed up generation of pivot tables and charts. There are restrictions to this functionality that can have some bearing on how you construct your data model, if you think your users would benefit from data model acceleration.

To accelerate a data model, it must contain at least one root event dataset, or one root search dataset that only uses streaming commands. Acceleration only affects these dataset types and datasets that are children of those root datasets. You cannot accelerate root search datasets that use nonstreaming commands (including transforming commands), root transaction datasets, and children of those datasets. Data models can contain a mixture of accelerated and unaccelerated datasets.

See Manage data models.

See Command types in the Search Reference for more information about streaming commands and other command types.

Example of data model dataset hierarchies

The following example shows the first several datasets in a "Call Detail Records" data model. Four top-level root datasets are displayed: All Calls, All Switch Records, Conversations, and Outgoing Calls.
**All Calls** and **All Switch Records** are root event datasets that represent all of the calling records and all of the carrier switch records, respectively. Both of these root event datasets have child datasets that deal with subsets of the data owned by their parents. The **All Calls** root event dataset has child datasets that break down into different call classifications: Voice, SMS, Data, and Roaming. If you were a Pivot user who only wanted to report on aspects of cellphone data usage, you'd select the Data dataset. But if you wanted to create reports that compare the four call types, you'd choose the **All Calls** root event dataset instead.

**Conversations** and **Outgoing Calls** are root transaction datasets. They both represent transactions—groupings of related events that span a range of time. The "Conversations" dataset only contains call records of conversations between two or more people where the maximum pause between conversation call record events is less than two hours and the total length of the conversation is less than one day.

For details about defining different data model dataset types, see Design data models.

**Dataset constraints**

All data model datasets are defined by sets of **constraints**. Dataset constraints filter out events that aren't relevant to the dataset.

- **For a root event dataset or a child dataset of any type**, the constraint looks like a simple search, without additional pipes and search commands. For example, the constraint for **HTTP Request**, one of the root event dataset of the Web Intelligence data model, is `sourcetype=access_*`.
- **For a root search dataset**, the constraint is the dataset search string.
- **For a root transaction dataset**, the constraint is the transaction definition. Transaction dataset definitions must identify **Group Dataset** (either one or more event dataset, a search dataset, or a transaction dataset) and one or more **Group By** fields. They can also optionally include **Max Pause** and **Max Span** values.

Constraints are inherited by child datasets. Constraint inheritance ensures that each child dataset represents a subset of the data represented by its parent datasets. Your Pivot users can then use these child datasets to design reports with datasets that already have extraneous data prefiltered out.
In the following example, we will use a data model called Buttercup Games. Its Successful Purchases dataset is a child of the root event dataset HTTP Requests and is designed to contain only those events that represent successful customer purchase actions. Successful Purchases inherits constraints from HTTP Requests and another parent dataset named Purchases.

1. HTTP Requests starts by setting up a search that only finds webserver access events.

   `sourcetype=access_*`

2. The Purchases dataset further narrows the focus down to webserver access events that involve purchase actions.

   `action=purchase`

3. And finally, Successful Purchases adds a constraint that reduces the dataset event set to web access events that represent successful purchase events.

   `status=200`

When all the constraints are added together, the base search for the Successful Purchases dataset looks like this:

```
sourcetype=access_* action=purchase status=200
```

A Pivot user might use this dataset for reporting if they know that they only want to report on successful purchase actions.

For details about datasets and dataset constraints, see the topic Design data models.

**Dataset field types**

There are five types of data model dataset fields.
## Auto-extracted

A field extracted by the Splunk software at **index time** or **search time**. You can only add auto-extracted fields to root datasets. Child datasets can inherit them, but they cannot add new auto-extracted fields of their own. Auto-extracted fields divide into three groups.

<table>
<thead>
<tr>
<th>Group</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fields added by <strong>automatic key value field extraction</strong></td>
<td>These are fields that the Splunk software extracts automatically, like <code>uri</code> or <code>version</code>. This group includes fields indexed through structured data inputs, such as fields extracted from the headers of indexed CSV files. See Extract fields from files with structured data in <em>Getting Data In</em>.</td>
</tr>
<tr>
<td>Fields added by knowledge objects</td>
<td>Fields added to search results by <strong>field extractions</strong>, <strong>automatic lookups</strong>, and <strong>calculated field</strong> configurations can all appear in the list of auto-extracted fields.</td>
</tr>
<tr>
<td>Fields that you have manually added</td>
<td>You can manually add fields to the auto-extracted fields list. They might be rare fields that you do not currently see in the dataset, but may appear in it at some point in the future. This set of fields can include fields added to the dataset by generating commands such as <code>inputcsv</code> or <code>dbinspect</code>.</td>
</tr>
</tbody>
</table>

### Eval Expression

A field derived from an **eval expression** that you enter in the field definition. Eval expressions often involve one or more extracted fields.

### Lookup

A field that is added to the events in the dataset with the help of a **lookup** that you configure in the field definition. Lookups add fields from external data sources such as CSV files and scripts. When you define a lookup field you can use any lookup object in your system and associate it with any other field that has already been associated with that same dataset.

See *About lookups*.

### Regular Expression

This field type is extracted from the dataset event data using a regular expression that you provide in the field definition. A regular expression field
definition can use a regular expression that extracts multiple fields; each field will appear in the dataset field list as a separate regular expression field.

**Geo IP**

A specific type of *lookup* that adds geographical fields, such as latitude, longitude, country, and city to events in the dataset that have valid IP address fields. Useful for map-related visualizations.

See Design data models.

**Field categories**

The Data Model Editor groups data model dataset fields into three categories.

<table>
<thead>
<tr>
<th>Category</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited</td>
<td>All datasets have at least a few inherited fields. Child fields inherit fields from their parent dataset, and these inherited fields always appear in the Inherited category. Root event, search, and transaction datasets also have default fields that are categorized as inherited.</td>
</tr>
<tr>
<td>Extracted</td>
<td>Any auto-extracted field that you add to a dataset is listed in the &quot;Extracted&quot; field category.</td>
</tr>
<tr>
<td>Calculated</td>
<td>The Splunk software derives calculated fields through a calculation, lookup definition, or field-matching regular expression. When you add Eval Expression, Regular Expression, Lookup, and Geo IP field types to a dataset, they all appear in this field category.</td>
</tr>
</tbody>
</table>

The Data Model Editor lets you arrange the order of calculated fields. This is useful when you have a set of fields that must be processed in a specific order. For example, you can define an Eval Expression that adds a set of fields to events within the dataset. Then you can create a Lookup with a definition that uses one of the fields calculated by the eval expression. The lookup uses this definition to add another set of fields to the same events.

**Field inheritance**

All data model datasets have inherited fields.

A child dataset will automatically have all of the fields that belong to its parent. All of these inherited fields will appear in the child dataset's "Inherited" category, even if the fields were categorized otherwise in the parent dataset.
You can add additional fields to a child dataset. The Data Model Editor will categorize these datasets either as extracted fields or calculated fields depending on their field type.

You can design a relatively simple data model where all of the necessary fields for a dataset tree are defined in its root dataset, meaning that all of the child datasets in the tree have the exact same set of fields as that root dataset. In such a data model, the child datasets would be differentiated from the root dataset and from each other only by their constraints.

Root event, search, and transaction datasets also have inherited fields. These inherited fields are default fields that are extracted from every event, such as _time, host, source, and sourcetype.

You cannot delete inherited fields, and you cannot edit their definitions. The only way to edit or remove an inherited field belonging to a child dataset is to delete or edit the field from the parent dataset it originates from as an extracted or calculated field. If the field originates in a root dataset as an inherited field, you won't be able to delete it or edit it.

You can hide fields from Pivot users as an alternative to field deletion.

You can also determine whether inherited fields are optional for a dataset or required.

**Field purposes**

Fields serve several purposes for data models.

Their most obvious function is to provide the set of fields that Pivot users use to define and generate a pivot report. The set of fields that a Pivot user has access to is determined by the dataset the user chooses when she enters the Pivot Editor. You might add fields to a child dataset to provide fields to Pivot users that are specific to that dataset.

On the other hand, you can also design calculated fields whose only function is to set up the definition of other fields or constraints. This is why field listing order matters: Fields are processed in the order that they are listed in the Data Model Editor. This is why The Data Model Editor allows you to rearrange the listing order of calculated fields.

For example, you could design a chained set of three Eval Expression fields. The first two Eval Expression fields would create what are essentially calculated
**fields.** The third Eval Expression field would use those two calculated fields in its eval expression.

**Fields can be visible or hidden to Pivot users**

When you define a field you can determine whether it is visible or hidden for Pivot users. This can come in handy if each dataset in your data model has lots of fields but only a few fields per dataset are actually useful for Pivot users.

A field can be visible in some datasets and hidden in others. Hiding a field in a parent dataset does not cause it to be hidden in the child datasets that descend from it.

Fields are visible by default. Fields that have been hidden for a dataset are marked as such in the dataset's field list.

The determination of what fields to include in your model and which fields to expose for a particular dataset is something you do to make your datasets easier to use in Pivot. It's often helpful to your Pivot users if each dataset exposes only the data that is relevant to that dataset, to make it easier to build meaningful reports. This means, for example, that you can add fields to a root dataset that are hidden throughout the model except for a specific dataset elsewhere in the hierarchy, where their visibility makes sense in the context of that dataset and its particular dataset.

Consider the example mentioned in the previous subsection, where you have a set of three "chained" Eval Expression fields. You may want to hide the first two Eval Expression fields because they are just there as "inputs" to the third field. You would leave the third field visible because it's the final "output"--the field that matters for Pivot purposes.

**Fields can be required or optional for a dataset**

During the field design process you can also determine whether a field is required or optional. This can act as a filter for the event set represented by the dataset. If you say a field is required, you're saying that every event represented by the dataset must have that field. If you define a field as optional, the dataset may have events that do not have that field at all.

**Note:** As with field visibility (see above) a field can be required in some datasets and optional in others. Marking a field as required in a parent dataset will not automatically make that field required in the child datasets that descend from that parent dataset.
Fields are optional by default. Fields that have had their status changed to *required* for a dataset are marked as such in the dataset's field list.

**Manage data models**

The Data Models management page is where you go to create data models and maintain some of their "higher order" aspects such as permissions and acceleration. On this page you can:

- **Create a new data model** - It's as easy as clicking a button.
- **Set permissions** - Data models are knowledge objects and as such are permissionable. You use permissions to determine who can see and update the data model.
- **Enable data model acceleration** - This can speed up Pivot performance for data models that cover large datasets.
- **Clone data models** - Useful for quick creation of new data models that are based on existing data models, or to copy data models into other apps.
- **Upload and download data models** - Download a data model (export it outside of Splunk). Upload an exported data model into a different Splunk implementation.
- **Delete data models** - Remove data models that are no longer useful.

In this topic we'll discuss these aspects of data model management. When you need to define the dataset hierarchies that make up a data model, you go to the Data Model Editor. See Design data models [http://docs.splunk.com/Documentation/Splunk/Minty/Search/Searchnormalization].

**Navigating to the Data Models management page**

The Data Models management page is essentially a listing page, similar to the Alerts, Reports, and Dashboards listing pages. It enables management of permissions and acceleration and also enables data model cloning and removal. It is different from the Select a Data Model page that you may see when you first enter Pivot (you'll only see it if you have more than one data model), as that page exists only to enable Pivot users to choose the data model they wish to use for pivot creation.

The Data Models management page lists all of the data models in your system in a paginated table. This table can be filtered by app, owner, and name. It can also display all data models that are visible to users of a selected app or just show

315
those data models that were actually created within the app.

If you use Splunk Cloud, or if you use Splunk Enterprise and have installed the Splunk Datasets Add-on, you may also see table datasets in the Data Models management page.

See About datasets for more information about table datasets.

There are two ways to get to the Data Models management page. You can use the Settings list, or you can get there through the Datasets listing page and Data Model Editor.

**Through the Settings list**

Navigate to Settings > Data Models.

**Through the Datasets listing page**

1. In the Search & Reporting app, open the Datasets listing page.
2. Locate a data model dataset.
3. (Optional) Click the name of the data model dataset to view it in the dataset viewing page.
4. Select Manage > Edit Data Model for that dataset.
5. On the Data Model Editor, click All Data Models to go to the Data Models management page.

**Create a new data model**

**Prerequisites**

You can only create data models if your permissions enable you to do so. Your role must have the ability to write to at least one app. If your role has insufficient permissions the New Data Model button will not appear.

See Enable roles to create data models.

**Steps**

1. Navigate to the Data Models management page.
2. Click New Data Model to create a new data model.
3. Enter the data model Title.
   
   The Title field can accept any character except asterisks. It can also accept blank spaces between characters.
The data model **ID** field fills in as you enter the title. Do not update it. The data model **ID** must be a unique identifier for the data model. It can only contain letters, numbers, and underscores. Spaces between characters are also not allowed. After you click **Create** you cannot change the **ID** value.

4. (Optional) Enter the data model **Description**.

5. (Optional) Change the 'App value if you want the data model to belong to a different app context. **App** displays app context that you are in currently.

6. Click **Create** to open the new data model in the Data Model Editor, where you can begin adding and defining the datasets that make up the data model.

When you first enter the Data Model Editor for a new data model it will not have any datasets. To define the data model's first dataset, click **Add Dataset** and select a dataset type. For more information about dataset definition, see the following sections on adding field, search, transaction, and child datasets.

For all the details on the Data Model Editor and the work of creating data model datasets, see Design data models.

**Enable roles to create data models**

By default only users with the **admin** or **power** role can create data models. For other users, the ability to create a data model is tied to whether their roles have "write" access to an app. To grant another role write access to an app, follow these steps.

**Steps**

1. Click the **App** dropdown at the top of the page and select **Manage Apps** to go to the Apps page.
On the Apps page, find the app that you want to grant data model creation permissions for and click Permissions.

3. On the Permissions page for the app, select Write for the roles that should be able to create data models for the app.

4. Click Save to save your changes.

Giving roles the ability to create data models can have other implications.

See Disable or delete knowledge objects.

**About data model permissions**

Data models are knowledge objects, and as such the ability to view and edit them is governed by role-based permissions. When you first create a data model it is private to you, which means that no other user can view it on the Select a Data Model page or Data Models management page or update it in any way.

If you want to accelerate a data model, you need to share it first. You cannot accelerate private data models. See Enable data model acceleration.

**Align data model permissions with those of related knowledge objects**

When you share a data model the knowledge objects associated with that data model (such as lookups or field extractions) must have the same permissions. Otherwise, people may encounter errors when they use the data model.

For example, if your data model is shared to all users of the Search app but uses a lookup table and lookup definition that is only shared with users of the Search app that have the Admin role, everything will work fine for Admin role users, but all other users will get errors when they try to use the data model in Pivot. The solution is either to restrict the data model to Admin users or to share the lookup table and lookup definition to all users of the Search app.

**Edit the permissions for a data model**

**Prerequisites**

- Manage knowledge object permissions

**Steps**

1. Go to the Data Models management page.
2. Locate the data model that you want to edit permissions for. Use one of the following options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Additional steps for this option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Edit &gt; Edit Permissions.</td>
<td>None</td>
</tr>
<tr>
<td>Expand the row for the dataset.</td>
<td>Click Edit for permissions.</td>
</tr>
</tbody>
</table>

3. Edit the dataset permissions and click **Save** to save your changes.

This brings up the **Edit Permissions** dialog, which you can use to share private data models with others, and to determine the access levels that various roles have to the data models.

**Manage data model acceleration**

Accelerated data models can return search results faster than they do when they ordinarily would. After you enable acceleration for a data model, you can inspect its metrics to ensure it is being accelerated correctly. If you determine that there are problems, you can rebuild the data summary for the data model.

**Enable data model acceleration**

After you enable acceleration for a data model, pivots, reports, and dashboard panels that use that data model can return results faster than they did before.

Data model acceleration builds a data summary for a data model at the indexer level. This summary can be made up of several smaller summaries distributed across your indexers.

After the data summary is built, searches that use accelerated data model datasets run against the summary rather than the full array of _raw data. This can speed up data model search completion times by a significant amount.

While data model acceleration is useful for speeding up searches on extremely large datasets, it has a few caveats.

- **After you accelerate a data model, you cannot edit it.** To make changes to an accelerated data model, you must disable its acceleration. Reaccelerating the data model can be resource-intensive, so it’s best to avoid disabling acceleration if you can.
- **Data model acceleration is applied only to root event datasets, root search datasets that restrict their command usage to streaming**
commands, and their child datasets. The Splunk platform cannot apply acceleration to dataset hierarchies based on root transaction datasets or root search datasets that use nonstreaming commands. Searches that use those unaccelerated datasets fall back to _raw data.

- **Data model acceleration is most efficient if the root event datasets or root search datasets include the indexes to be searched in their initial constraint search.** Otherwise, all available indexes for the data model are searched.

**Prerequisites**

- A role with the accelerate_datamodel capability, such as the admin role. Data model acceleration can be resource-intensive, so it should be used conservatively by a limited number of users.
- The data model must be shared with other users before you can accelerate it, and any knowledge objects that the data model is dependent on must be shared as well. See About data model permissions.
- See Accelerate data models.
- See Command types in the Search Reference for more information about streaming, generating, and transforming commands.
- See Specify time modifiers in your search in the Search Manual or learn about setting fixed UNIX time dates if you intend to enter a Custom Summary Range.

**Steps**

1. In Splunk Web, go to the Data Models management page.
2. Find the data model you want to accelerate and open its acceleration controls. Use one of the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Additional steps for this option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigate to the Data Models management page.</td>
<td>Find the model you want to accelerate and select <strong>Edit &gt; Edit Acceleration</strong>.</td>
</tr>
<tr>
<td>Navigate to the Data Models management page.</td>
<td>Expand the row of the data model you want to accelerate and click <strong>Add for ACCELERATION</strong>.</td>
</tr>
<tr>
<td>Open the Data Model Editor for a data model.</td>
<td>Select <strong>Edit &gt; Edit Acceleration</strong>.</td>
</tr>
</tbody>
</table>

3. Select **Accelerate** to enable acceleration for the data model.
4. Select a **Summary Range** of **1 Day**, **7 Days**, **1 Month**, **3 Months**, **1 Year**, **All Time**, or **Custom** depending on the range of time over which you plan to run searches that use the accelerated datasets within the data model. For example, if you only plan to run searches with this data model over periods of time within the last seven days, choose **7 Days**.

Select **Custom** to provide a custom earliest time range. You can use relative time notation, or you can provide a fixed date in Unix epoch time format.

Smaller time ranges result in smaller summaries that require less time to build and take up less space on disc.

5. Click **Save**.

After your data model is accelerated, the icon for the model on the Data Models management page is yellow instead of gray.

**Inspect data model acceleration metrics**

After a data model is accelerated, you can find information about the model's acceleration on the Data Models management page. Expand the row for the accelerated data model and review the information that appears under **ACCELERATION**.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status</strong></td>
<td>Tells you whether the acceleration summary for the data model is complete. If it is in building status it will tell you what percentage of the summary is complete. Data model summaries can constantly update with new data. Just because a summary is complete now doesn't mean it won't be building later.</td>
</tr>
<tr>
<td><strong>Access Count</strong></td>
<td>Tells you how many times the data model summary has been accessed since it was created, and when the last access time was. This metric can help you determine which data models are</td>
</tr>
</tbody>
</table>
infrequently used. Because data model acceleration uses system resources, you should restrict acceleration to data models that are accessed frequently.

| **Size on Disk** | Shows you how much space the data model's acceleration summary takes up in terms of storage. You can use this metric along with the Access Count to determine which summaries are an unnecessary load on your system and ought to be deleted. If the acceleration summary for your data model is taking up a large amount of space on disk, you might also consider reducing its summary range. |
| **Summary Range** | The range of the data model, in seconds, always relative to the present moment. You set this range up when you define acceleration for the data model. |
| **Buckets** | The number of index buckets spanned by the data model acceleration summary. |
| **Updated** | Tells you when the summary was last updated with the results of a summarization search. |

Rebuild a summary for an accelerated data model

You may want to rebuild the summary for your data model if you suspect there has been data loss due to a system crash or similar mishap. When you rebuild your summary, Splunk software deletes the entire acceleration summary for this data model and rebuilds it. This can take a long time for larger summaries.

The Splunk platform automatically rebuilds summaries when you disable and then reenable acceleration for a summary. You might disable and reenable acceleration for a data model when you edit the data model, because the data model cannot be edited when it is in an accelerated state.

Prerequisites

- An accelerated data model.

Steps

1. In Splunk Web, go to the Data Models management page.
2. Find the accelerated data model that needs to have its summary rebuilt, and expand its row.
3. Click Rebuild. The summary will begin rebuilding.
4. (Optional) Check the Status of the summary to find out when it is complete.
**Update summary metrics for an accelerated data model**

Data model acceleration metrics are updated on a regular interval. If you do not want to wait for a scheduled update, you can get the metrics updated right away by clicking the **Update** button.

**Prerequisites**

- An accelerated data model.

**Steps**

1. In Splunk Web, go to the Data Models management page.
2. Expand the row of an accelerated data model to see its acceleration metrics.
3. Click **Update** to have the Splunk platform update the metrics it displays for the data model.

**Edit the advanced data model acceleration settings**

If you run into issues with summary creation for a data model, you may need to adjust its advanced data model acceleration settings. Use this table to

**Clone a data model**

Data model cloning is a way to quickly create a data model that is based on an existing data model. You can then edit it so it focuses on a different overall dataset or has a different dataset structure that divides up the dataset in a different way than the original.

**Steps**

1. Use one of the following options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Additional steps for this option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Go to the Data Models management page.</td>
<td>Locate the data model that you want to clone and select <strong>Edit &gt; Clone</strong>.</td>
</tr>
<tr>
<td>Open the Data Model Editor for the data model that you want to clone.</td>
<td>Select <strong>Edit &gt; Clone</strong>.</td>
</tr>
</tbody>
</table>

2. Enter a unique name for the cloned data model in **New Title**.

323
3. (Optional) Provide a **Description** for the new data model.
4. (Optional) If your permissions allow it, select **Clone** to give the cloned data model the same permissions as the data model it is cloned from.
5. Click **Clone** to create the data model clone.

You can edit the cloned data model in the **Data Model** management page, and the **Data Model Editor**, as described in Design data models.

**Upload and download data models**

You can use the download/upload functionality to export a data model from one Splunk deployment and upload it into another Splunk deployment. You can use this feature to back up important data models, or to collaborate on data models with other Splunk users by emailing them to those users. You might also use it to move data models between staging and production instances of Splunk.

You can manually move data model JSON files between Splunk deployments, but this is an unsupported procedure with many opportunities for error.

See Manual data model management.

**Download a data model**

Download a data model from the Data Model Editor. You can only download one data model at a time.

**Steps**

1. Open a data model in the Data Model Editor.
2. Click the **Download** button at the top right.

Splunk will download the JSON file for the data model to your designated download directory. If you haven’t designated this directory, you may see a dialog that asks you to identify the directory you want to save the file to.

The name of the downloaded JSON file will be the same as the data model's **ID**. You provide the **ID** only once, when you first create the data model. Unlike the data model **Title**, once the **ID** is saved with the creation of the model, you can’t change it.
You can see the **ID** for an existing data model when you view the model in the Data Model Editor. The **ID** appears near the top left corner of the Editor, under the model's title.

When you upload the data model you have an opportunity to give it a new **ID** that is different from the **ID** of the original data model.

**Upload a data model**

Upload a data model from the Data Models management page. You can only upload one data model at a time.

Splunk software validates any file that you try to upload. It cannot upload files that contain anything other than valid JSON data model code.

**Steps**

1. Navigate to the Data Models management page.
2. Click **Upload Data Model**.
3. Identify the JSON File that you want to upload.
   The **ID** field populates with the original ID of the data model.
4. (Optional) Change the data model ID to a new, unique value.
   Keep in mind that once you save the data model file to your system you will not be able to change this ID. You can still edit the data model title after you save it to your system.
5. Provide the name of the **App** that the data model belongs to.
6. (Optional) If your capabilities allow it, change the uploaded data model permissions from **Private** to **Shared in App**.
   - **Shared in App** indicates that the data model is shared with all users of the **App**.
   - If you select **Shared in App** you can also enable acceleration for the data model by selecting **Accelerate** and choosing a **Summary Range**.
7. Click **Upload** to upload the data model.
   The uploaded data model appears in the Data Model management page listing if it passes validation.

See About data model permissions.

See Enable data model acceleration.
Delete a data model

You can delete a data model from the Data Model Editor or the Data Models management page.

If your role grants you the ability to create data models, it should grant you the ability to delete them as well. For more information about this see Enable roles to create data models.

Delete a data model from the Data Model Editor

1. In the Search & Reporting app, click **Datasets** to open the Data model datasets listing page.
2. Locate a data model dataset that belongs to the dataset that you want to delete.
3. Select **Manage > Edit Dataset**.
4. In the Data Model Editor, select **Edit > Delete**.

Delete a data model from the Data Models management page

1. In the Search & Reporting app, click **Datasets** to open the Data model datasets listing page.
2. Locate a data model dataset that belongs to the dataset that you want to delete.
3. Select **Manage > Edit Dataset**.
4. Click **All Data Models**.
5. Locate the data model that you want to delete.
6. Select **Manage > Edit**.

Manual data model management

Splunk does not recommend that you manage data models manually by hand-moving their files or hand-coding data model files. You should create and edit data models in Splunk Web whenever possible. When you edit models in Splunk Web the Data Model Editor validates your changes. The Data Model Editor cannot validate changes in models created or edited by hand.

Data models are stored on disk as JSON files. They have associated configs in `datamodels.conf` and metadata in `local.meta` (for models that you create) and `default.meta` (for models delivered with the product).

Models that you create are stored in `<yourapp>/local/data/models`, while models delivered with the product can be found in...
You can manually move model files between Splunk implementations but it's far easier to use the Data Model Download/Upload feature in Splunk Web (described above). If you absolutely must move model files manually, take care to move their `datamodels.conf` stanzas and `local.meta` metadata when you do so.

The same goes for deleting data models. In general it's best to do it through Splunk Web so the appropriate cleanup is carried out.

**Design data models**

In Splunk Web, you use the Data Model Editor to design new data models and edit existing models. This topic shows you how to use the Data Model Editor to:

- Build out data model dataset hierarchies by adding root datasets and child datasets to data models.
- Define datasets (by providing constraints, search strings, or transaction definitions).
- Rename datasets.
- Delete datasets.

You can also use the Data Model Editor to create and edit dataset fields. For more information, see Define dataset fields.

**Note:** This topic will not spend much time explaining basic data model concepts. If you have not worked with Splunk data models, you may find it helpful to review the topic About data models. It provides a lot of background detail around what data models and data model datasets actually are and how they work.

For information about creating and managing new data models, see Manage data models in this manual. Aside from creating new data models via the Data Models management page, this topic will also show you how to manage data model permissions and acceleration.

**The Data Model Editor**

Data models are collections of data model datasets arranged in hierarchical structures. To design a new data model or redesign an existing data model, you go to the Data Model Editor. In the Data Model Editor, you can create datasets.
for your data model, define their constraints and **fields**, arrange them in logical dataset hierarchies, and maintain them.

You can only edit a specific data model if your permissions enable you to do so.

**Navigate to the Data Model Editor**

To open the Data Model Editor for an existing data model, choose one of the following options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Additional steps for this option</th>
</tr>
</thead>
<tbody>
<tr>
<td>From the <strong>Data Models</strong> page in <strong>Settings</strong>.</td>
<td>Find the data model you want to edit and select <strong>Edit &gt; Edit Datasets</strong>.</td>
</tr>
<tr>
<td>From the Datasets listing page</td>
<td>Find a data model dataset that you want to edit and select <strong>Manage &gt; Edit data model</strong>.</td>
</tr>
<tr>
<td>From the Pivot Editor</td>
<td>Click <strong>Edit dataset</strong> to edit the data model dataset that the Pivot editor is displaying.</td>
</tr>
</tbody>
</table>

**Add a root event dataset to a data model**

Data models are composed chiefly of dataset hierarchies built on root event dataset. Each root event dataset represents a set of data that is defined by a **constraint**: a simple search that filters out events that aren't relevant to the dataset. Constraints look like the first part of a search, before pipe characters and additional search commands are added.
Constraints for root event datasets are usually designed to return a fairly wide range of data. A large dataset gives you something to work with when you associate child event datasets with the root event dataset, as each child event dataset adds an additional constraint to the ones it inherits from its ancestor datasets, narrowing down the dataset that it represents.

For more information on how constraints work to narrow down datasets in a dataset hierarchy, see dataset constraints.

To add a root event dataset to your data model, click Add Dataset in the Data Model Editor and select Root Event. This takes you to the Add Event Dataset page.

Give the root event dataset a Dataset Name, Dataset ID, and one or more Constraints.

The Dataset Name field can accept any character except asterisks. It can also accept blank spaces between characters. It's what you'll see on the Choose a Dataset page and other places where data model datasets are listed.

The Dataset ID must be a unique identifier. It can contain only characters that are alphanumeric, underscores, or hyphens (a-z, A-Z, 0-9, _, or -). It cannot include spaces between characters. Once you save a Dataset ID value that value cannot be edited.

After you provide Constraints for the root event dataset you can click Preview to test whether the constraints you have supplied return the kinds of events you want.
Add a root search dataset to a data model

Root search datasets enable you to create dataset hierarchies where the base dataset is the result of an arbitrary search. You can use any SPL in the search string that defines a root search dataset.

You cannot accelerate root search datasets that use transforming searches. A transforming search uses transforming commands to define a base dataset where one or more fields aggregate over the entire dataset.

To add a root search dataset to your data model, click Add Dataset in the Data Model Editor and select Root Search. This takes you to the Add Search Dataset page.

Give the root search dataset a Dataset Name, Dataset ID, and search string. To preview the results of the search in the section at the bottom of the page, click the magnifying glass icon to run the search, or just hit return on your keyboard while your cursor is in the search bar.

The Dataset Name field can accept any character except asterisks. It can also accept blank spaces between characters. It's what you'll see on the Choose a Dataset page and other places where data model datasets are listed.

The Dataset ID must be a unique identifier for the dataset. It cannot contain spaces or any characters that aren't alphanumeric, underscores, or hyphens (a-z, A-Z, 0-9, _, or -). Spaces between characters are also not allowed. Once you save the Dataset ID, value you can't edit it.

For more information about designing search strings, see the Search Manual.
Don’t create a root search dataset for your search if the search is a simple
transaction search. Set it up as a root transaction dataset.

**Using transforming searches in a root search dataset**

You can create root search datasets for searches that do not map directly to Splunk events, as long as you understand that they cannot be accelerated. In other words, searches that involve input or output that is not in the format of an event. This includes searches that:

- Make use of transforming commands such as `stats`, `chart`, and `timechart`. Transforming commands organize the data they return into tables rather than event lists.
- Use other commands that do not return events.
- Pull in data from external non-Splunk sources using a command other than `lookup`. This data cannot be guaranteed to have default fields like `host`, `source`, `sourcetype`, or `_time` and therefore might not be event-mappable. An example would be using the `inputcsv` command to get information from an external `.csv` file.

**Add a root transaction dataset to a data model**

Root transaction datasets enable you to create dataset hierarchies that are based on a dataset made up of transaction events. A transaction event is actually a collection of conceptually-related events that spans time, such as all website access events related to a single customer hotel reservation session, or all events related to a firewall intrusion incident. When you define a root transaction dataset, you define the transaction that pulls out a set of transaction events.

Read up on transactions and the `transaction` command if you’re unfamiliar with how they work. Get started at About transactions, in the Search Manual. Get detail information on the `transaction` command at its entry in the Search Reference.

Root transaction datasets and their children do not benefit from data model acceleration.

To add a root transaction dataset to your data model, click Add Dataset in the Data Model Editor and select Root Transaction. This takes you to the Add Transaction Dataset page.
Root transaction dataset definitions require a **Dataset Name** and **Dataset ID** and at least one **Group Dataset**. The **Group by**, **Max Pause**, and **Max Span** fields are optional, but the transaction definition is incomplete until at least one of those three fields is defined.

The **Dataset Name** field can accept any character except asterisks. It can also accept blank spaces between characters. It's what you'll see on the Choose a dataset page and other places where data model datasets are listed.

The **Dataset ID** must be a unique identifier for the dataset. It cannot contain spaces or any characters that aren't alphanumeric, underscores, or hyphens (a-z, A-Z, 0-9, _, or -). Spaces between characters are also not allowed. Once you save the **Dataset ID** value you can't edit it.

All root transaction dataset definitions require one or more **Group Dataset** names to define the pool of data from which the transaction dataset will derive its transactions. There are restrictions on what you can add under **Group Dataset**, however. **Group Dataset** can contain one of the following three options:

- One or more event datasets (either root event datasets or child event datasets)
- One transaction dataset (either root or child)
- One search dataset (either root or child)

In addition, you are restricted to datasets that exist within the currently selected data model.

If you're familiar with how the `transaction` command works, you can think of the
**Group Datasets** as the way we provide the portion of the search string that appears *before* the transaction command. Take the example presented in the preceding screenshot, where we’ve added the *Apache Access Search* dataset to the definition of the root transaction dataset Web Session. *Apache Access Search* represents a set of successful webserver access events--its two constraints are status < 600 and sourcetype = access_* OR source = *.log. So the start of the transaction search that this root transaction dataset represents would be:

```
status < 600 sourcetype=access_* OR source=*.log | transaction...
```

Now we only have to define the rest of the *transaction* argument.

**Add a child dataset to a data model**

You can add child datasets to root datasets and other child datasets. A child dataset inherits all of the constraints and fields that belong to its parent dataset. A single dataset can be associated with multiple child datasets.

When you define a new child dataset, you give it one or more additional constraints, to further focus the dataset that the dataset represents. For example, if your Web Intelligence data model has a root event dataset called HTTP Request that captures all webserver access events, you could give it three child event datasets: HTTP Success, HTTP Error, and HTTP Redirect. Each child event dataset focuses on a specific subset of the HTTP Request dataset:

- **The child event dataset** *HTTP Success* uses the additional constraint status = 2* to focus on successful webserver access events.
- **HTTP Error** uses the additional constraint status = 4* to focus on failed webserver access events.
- **HTTP Redirect** uses the additional constraint status = 3* to focus on redirected webserver access events.

The addition of fields beyond those that are inherited from the parent dataset is optional. For more information about field definition, see Manage Dataset fields with the Data Model Editor.

To add a child dataset to your data model, select the parent dataset in the left-hand dataset hierarchy, click Add Dataset in the Data Model Editor, and select Child. This takes you to the Add Child Dataset page.
Give the child dataset a **Dataset Name** and **Dataset ID**.

The **Dataset Name** field can accept any character except asterisks. It can also accept blank spaces between characters. It's what you'll see on the Choose a Dataset page and other places where data model datasets are listed.

The **Dataset ID** must be a unique identifier for the dataset. It cannot contain spaces or any characters that aren't alphanumeric, underscores, or hyphens (a-z, A-Z, 0-9, _, or -). Spaces between characters are also not allowed. After you save the **Dataset ID** value you can't edit it.

After you define a **Constraint** for the child dataset you can click **Preview** to test whether the constraints you've supplied return the kinds of events you want.

**Set a tag whitelist for better data model search performance**

When the Splunk software processes a search that includes tags, it loads all of the tags defined in *tags.conf* by design. This means that data model searches that use tags can suffer from reduced performance.

You should set up a tag whitelist for data models that meet the following criteria:

- Data models that include tags in their constraint searches.
- Data models that are commonly referenced in searches that also include tags in their search strings.
Set up tag whitelists for data models

If you have .conf file access you can define tag whitelists for your data models by setting tags_whitelist in their datamodels.conf stanzas.

Set tags_whitelist with a comma-separated list of the tags that you want the Splunk software to process. The list should include all of the tags in the constraint searches for the data model and any additional tags that you commonly use in searches that reference the data model.

If you use the Splunk Common Information Model Add-on

Use the setup page for the Common Information Model (CIM) Add-on to edit the tag whitelists of CIM data models. See Set up the Splunk Common Information Model Add-on in Common Information Model Add-on Manual.

How tag whitelists work

When you run a search that references a data model with a tag whitelist, the Splunk software only loads the tags identified in that whitelist. This improves the performance of the search, because the tag whitelist prevents the search from loading all the tags in tags.conf.

The tags_whitelist setting does not validate to ensure that the tags it lists are present in the data model it is associated with.

If you do not configure a tag whitelist for a data model, the Splunk software attempts to optimize out unnecessary tags when you use that data model in searches. Data models that do not have tag fields in their constraint searches cannot use the tags_whitelist setting.

A tag whitelist example

You have a data model named Network_Traffic with constraint searches include the network and communicate tags. When you run a search against the Network_Traffic data model, the Splunk software processes both of those tags, but it also processes the other 234 tags that are configured in the tags.conf file for your deployment.

In addition, your most common searches against Network_Traffic include the destination tag. For example, a typical Network_Traffic search might be as follows:
In this case, you can set `tags_whitelist = network, communicate, destination` for the `Network_Traffic` stanza in `datamodels.conf`.

After you do this, any search you run against the `Network_Traffic` data model only loads the `network`, `communicate`, and `destination` tags. The rest of the tags in `tags.conf` are not factored into the search. This optimization should cause the search to complete faster than it would if you had not set `tags_whitelist` for `Network_Traffic`.

Some best practices for data model design

It can take some trial and error to determine the data model designs that work for you. Here are some tips that can get you off to a running start.

- **Use root event datasets and root search datasets that only use streaming commands.** This takes advantage of the benefits of data model acceleration.

- **When you define constraints for a root event dataset or define a search for a root search dataset that you will accelerate, include the index or indexes it is selecting from.** Data model acceleration efficiency and accuracy is improved when the data model is directed to search a specific index or set of indexes. If you do not specify indexes, the data model searches over all available indexes.

- **Minimize dataset hierarchy depth.** Constraint-based filtering is less efficient deeper down the tree.

- **Use field flags to selectively expose small groups of fields for each dataset.** You can expose and hide different fields for different datasets. A child field can expose an entirely different set of fields than those exposed by its parent. Your Pivot users will benefit from this selection by not having to deal with a bewildering array of fields whenever they set out to make a pivot chart or table. Instead they'll see only those fields that make sense in the context of the dataset they've chosen.

- **Reverse-engineer your existing dashboards and searches into data models.** This can be a way to quickly get started with data models. Dashboards built with pivot-derived panels are easier to maintain.

- **When designing a new data model, first try to understand what your Pivot users hope to be able to do with it.** Work backwards from there. The structure of your model should be determined by your users’ needs and expectations.
Define data model dataset fields

Define dataset fields

In this topic we talk about adding and editing fields of data model datasets. Dataset fields provide the set of fields that your Pivot users work with when they define and generate pivot reports.

Fields can be present within the dataset, or they can be derived and added to the dataset through the use of lookups and eval expressions.

You use the Data Model Editor to create and manage dataset fields. It enables you to:

- Create new fields.
- Update or delete existing fields that aren't inherited.
- Override certain settings for inherited fields.

You can also use the Data Model Editor to build out data model dataset hierarchies, define datasets (by providing constraints, search strings, or transaction definitions), rename datasets, and delete datasets. For more information about using the Data Model Editor to perform these tasks, see Design data models.

This topic will not cover the concepts behind dataset fields in detail. If you have not worked with data model fields up to this point, you should review the topic About data models.

For information about creating and managing new data models, see Manage data models. Aside from creating new data models via the Data Models management page, this topic also shows you how to manage data model permissions and acceleration.

Data model dataset field types

There are five types of data model dataset fields.

Auto-extracted

A field extracted by the Splunk software at index time or search time. You can only add auto-extracted fields to root datasets. Child datasets can inherit them,
but they cannot add new auto-extracted fields of their own. Auto-extracted fields divide into three groups.

<table>
<thead>
<tr>
<th>Group</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fields added by <strong>automatic key value field extraction</strong></td>
<td>These are fields that the Splunk software extracts automatically, like uri or version. This group includes fields indexed through structured data inputs, such as fields extracted from the headers of indexed CSV files. See Extract fields from files with structured data in Getting Data In.</td>
</tr>
<tr>
<td>Fields added by <strong>knowledge objects</strong></td>
<td>These are fields added to search results by <strong>field extractions</strong>, automatic <strong>lookups</strong>, and <strong>calculated fields</strong> that are configured in props.conf.</td>
</tr>
<tr>
<td>Fields that you have manually added</td>
<td>You can manually add fields to the auto-extracted fields list. They might be rare fields that you do not currently see in the dataset, but may appear in it at some point in the future. This set of fields can include fields added to the dataset by generating commands such as inputcsv or dbinspect.</td>
</tr>
</tbody>
</table>

**Eval Expression**

A field derived from an eval expression that you enter in the field definition. Eval expressions often involve one or more extracted fields.

**Lookup**

A field that is added to the events in the dataset with the help of a lookup that you configure in the field definition. Lookups add fields from external data sources such as CSV files and scripts. When you define a lookup field you can use any lookup object in your system and associate it with any other field that has already been associated with that same dataset.

See About lookups.

**Regular Expression**

This field type is extracted from the dataset event data using a regular expression that you provide in the field definition. A regular expression field definition can use a regular expression that extracts multiple fields; each field will appear in the dataset field list as a separate regular expression field.
**Geo IP**

A specific type of lookup that adds geographical fields, such as latitude, longitude, country, and city to events in the dataset that have valid IP address fields. Useful for map-related visualizations.

See Design data models.

**Field categories**

The Data Model Editor groups data model dataset fields into three categories.

<table>
<thead>
<tr>
<th>Category</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited</td>
<td>All datasets have at least a few inherited fields. Child fields inherit fields from their parent dataset, and these inherited fields always appear in the Inherited category. Root event, search, and transaction datasets also have default fields that are categorized as inherited.</td>
</tr>
<tr>
<td>Extracted</td>
<td>Any auto-extracted field that you add to a dataset is listed in the &quot;Extracted&quot; field category.</td>
</tr>
<tr>
<td>Calculated</td>
<td>The Splunk software derives calculated fields through a calculation, lookup definition, or field-matching regular expression. When you add Eval Expression, Regular Expression, Lookup, and Geo IP field types to a dataset, they all appear in this field category.</td>
</tr>
</tbody>
</table>

**Field order and field chaining**

The Data Model Editor lets you rearrange the order of calculated fields. This is useful when you have a set of fields that must be processed in a specific order, because fields are processed in descending order from the top of the list to the bottom.

For example, you can design an Eval Expression field that uses the values of two auto-extracted fields. Extracted fields precede calculated fields, so in this case the fields would be processed in the correct order without any work on your part. But you might also use the eval expression field as input for a lookup field. Because Eval Expression fields and Lookup fields are both categorized as calculated fields by the Data Model Editor, you would want to make sure that you order the calculated field list so that the Eval Expression field appears above the Lookup field.

So the order of these fields would be:
• Auto Extracted Field 1
• Auto Extracted Field 2
• Eval Expression Field (calculates a field with the values of the two Auto-Extracted fields)
• Lookup Field (uses the Eval Expression field as an input field)

**Marking fields as hidden or required**

All dataset fields are *shown* and *optional* by default.

• A *shown* field is visible and available to Pivot users when they are in the context of the dataset to which the field belongs. For example, say the `url` field is marked as shown for the HTTP Requests dataset. When a user enters Pivot and selects the HTTP Requests dataset, they can use the `url` field when they define a pivot report.

• An *optional* field is *not* required to be present in every event in the dataset represented by its dataset. This means that there potentially can be many events in the dataset that do not contain the field.

You can change these settings to *hidden* and *required*, respectively. When you do this the field will be marked as *hidden* and/or *required* in the dataset field list.

• A *hidden* field is *not* displayed to Pivot users when they select the dataset in a Pivot context. They will be unable to use it for the purpose of Pivot report definition.
  ♦ This setting lets you expose different subsets of fields for each dataset in your data model, even if all of the datasets inherit the same set of fields from a single parent dataset. This helps to ensure that your Pivot users only engage with fields that make sense given the context of the dataset represented by the dataset.
  ♦ You can hide field fields that are only being added to the dataset because they're used to define another field (see "Field order and field chaining," above). There may be no need for your Pivot users to engage with the first fields in a field chain.

• A *required* field *must* appear in every event represented by the dataset. This filters out any event that does not have the field. In effect this is another type of *constraint* on top of any formal constraints you've associated with the dataset.

These field settings are specific to each dataset in your data model. This means you can have the `ip_address` field set to *Required* in a parent dataset but still set as optional in the child datasets that descend from that parent dataset. Even if all of the datasets in a data model have the same fields (meaning the fields are set
in the topmost root dataset and then simply inherited to all the other datasets in the hierarchy), the fields that are marked hidden or required can be different from dataset to dataset in that data model.

**Note:** There is one exception to your ability to provide different "shown/hidden" and "optional/required" settings for the same field across different datasets in a data model. **You cannot update these settings for inherited fields that are categorized as "Calculated" fields in the parent dataset in which they first appear.** For this kind of field you can only change the setting by updating the fields in that parent dataset. Your changes will be replicated through the child dataset that descend from that parent dataset.

You can set these values for extracted and calculated fields when you first define them. You can also edit field names or types after they've been defined.

1. Click **Override** for a field in the Inherited category or **Edit** for a field in the Extracted and Calculated categories.
2. Change the value of the **Flag** field to the appropriate value.
3. Click **Save** to save your changes.

With the **Bulk Edit** list you can change the "shown/hidden" and "optional/required" values for multiple fields at once.

1. Select the fields you want to edit.
2. Click **Bulk Edit** and select either **Optional**, **Required**, **Hidden**, or **Shown**.

If you select either **Required** or **Hidden** the appropriate fields update to display the selected status for the selected fields. You cannot update these values for inherited fields that are categorized as calculated fields in the parent dataset in which they first appear. See the **Note** above for more information.

**Enter or update field names and types**

The Data Model Editor lets you give fields in the **Extracted** and **Calculated** categories a display **Name** of your choice. It also lets you determine the **Type** for such fields, even in cases where a **Type** value has been automatically assigned to the field.

Splunk software automatically assigns a type to auto-extracted fields. If an auto-extracted field's **Type** value is assigned incorrectly, you can provide the correct one. For example, based on available values for an auto-extracted field, Splunk software may decide it is a **Number** type field when you know that it is in
fact a *String* type. You can change the *Type* value to *String* if this is the case.

Changing the display *Name* of an auto-extracted field won't change how the associated field is named in the index--it just renames it in the context of this data model.

1. Click **Edit** for the field whose *Name* or *Type* you would like to update.
2. Update the *Name* or change the *Type*. *Name* values cannot contain asterisk characters.
3. Click **Save** to save your changes.

Use the **Bulk Edit** list to give multiple fields the same *Type* value.

1. Select the fields you want to edit.
2. Click **Bulk Edit** and select either *Boolean*, *IPv4*, *Number*, or *String*.
   - You cannot change the *Type* value for inherited fields. If you select any inherited fields the *Type* values in the **Bulk Edit** list will be unavailable.

All of the selected fields should have their *Type* value updated to the value you choose.

**Add an auto-extracted field**

You can add an auto-extracted field to any root dataset in your data model.

1. In the Data Model Editor, open the root dataset you'd like to add an auto-extracted field to.
2. Click **Add Field** and select *Auto-extracted* to define an auto-extracted field.

   The Add Auto-Extracted Field dialog appears. It includes a list of fields that can be added to your data model datasets.

3. Select the fields you would like to add to your data model by marking their checkboxes.

   You can select the checkbox in the header to select all fields in the list.

   If you look at the list and don’t find the fields you are expecting, try changing the event sample size, which is set to the *First 1000 events* by default. A larger event sample may contain rare fields that didn’t turn up in the first thousand events. For example, you could choose a sample size like the *First 10,000 events* or the *Last 7 days*.

4. (Optional) **Rename** the auto-extracted field.

   If you use Rename, do not include asterisk characters in the new field name.

5. (Optional) Correct the auto-extracted field **Type**.

6. (Optional) Update the auto-extracted field’s status (*Optional*, *Required*, *Hidden*, or *Hidden and Required*) as necessary.

7. Click **Save** to add the selected fields to your root dataset.

   **Note:** You cannot add auto-extracted fields to child datasets. Child datasets inherit auto-extracted fields from the root dataset at the top of their dataset hierarchy.

The list of fields displayed by the Add Auto-Extracted Field dialog includes:

- Fields that are extracted automatically, like *uri* or *version*. This includes fields indexed through structured data inputs, such as fields extracted from the headers of indexed CSV files.
- **Field extractions, lookups, or calculated fields** that you have defined in Settings or configured in *props.conf*.

Expand a field row for a field to see its top ten sample values.

**Manually add a field to the set of auto-extracted fields**

While building a data model you may find that you are missing certain auto-extracted fields. They could be missing for a variety of reasons. For example:
• You may be building your data model prior to indexing the data that will make up its dataset.
• You are indexing data, but certain rare fields that you expect to see eventually haven’t been indexed yet.
• You are utilizing a generating search command like input.csv that adds fields that don’t display in this list.

You can manually add auto-extracted fields to a root dataset.

**Note:** Before adding fields manually, try increasing the event sample size as described in the procedure above to pull in rare fields that aren’t found in the first thousand events.

1. Click **Add by name** in the top right-hand corner of the Add Auto-Extracted Field dialog.
   
   This adds a row to the field table. Note that in the example at the top of this topic a row has been added for a manually added ISBN field.

2. In that row, manually identify the **Field** name, **Type**, and status for an auto-extracted field.

3. Click **Add by name** again to add additional field rows.

4. Click the **X** in the top right-hand corner of an added row to remove it.

5. Click **Save** to save your changes.

   Fields that you’ve added to the table are added to your root dataset as **Extracted** in the **Extracted category**, along with any selected auto-extracted fields.

### Add an eval expression field

You can add eval expression fields to any dataset in your data model. Use eval expressions to create fields and add them to events in a manner similar to that of **calculated fields**.
1. In the Data Model Editor, open the dataset that you would like to add a field to.

2. Click **Add Field**. Select *Eval Expression* to define an eval expression field. The Add Fields with an Eval Expression dialog appears.

3. Enter the **Eval Expression** that defines the field value.
   The Eval Expression text area should just contain the `<eval-expression>` portion of the `eval` syntax. There’s no need to type the full syntax used in Search (`eval <eval-field>='<eval-expression>'`).

4. Under **Field** enter the **Field Name** and **Display Name**.
   The **Field Name** is the name in your dataset. The **Display Name** is the field name that your Pivot users see when they create pivots.
   **Note:** The **Field Name** cannot include whitespace, single quotes, double quotes, curly braces, or asterisks. The field **Display Name** cannot contain asterisks.

5. Define the field **Type** and set its **Flag**.
   For more information about the **Flag** values, see the subsection on marking fields as hidden or required in Define dataset fields.

6. (Optional) Click **Preview** to verify that the eval expression is working as expected.
   You should see events in table format with the new eval field(s) included as columns. For example, if you’re working with an event-based dataset and you’ve added an eval field named `gb`, the preview event table should show a column labeled `gb` to the right of the first column `_time`.
   The preview pane has two tabs. **Events** is the default tab. It presents the events in table format. The new eval field should appear to the right of the first column (the `_time` column).
   If you do not see values in this column, or you see the same value repeated in the events at the top of the list, it could mean that more values appear later in the sample. Select the **Values** tab to review...
the distribution of eval field values among the selected event sample. You can also change the Sample value to increase the number of events in the preview sample—this can sometimes uncover especially rare values of the field created by the eval expression.

In the example below, the three real-time searches only appeared in the value distribution when Sample was expanded from First 1,000 events to First 10,000 events.

7. Click Save to save your changes and return to the Data Model Editor.

For more information about the eval command and the formatting of eval expressions, see the eval page as well as the topic Evaluation functions in the Search Reference.

Eval expressions can utilize fields that have already been defined or calculated, which means you can chain fields together. Fields are processed in the order that they are listed from top to bottom. This means that you must place prerequisite fields above the eval expression fields that uses those fields in its eval expression. In other words, if you have a calculation B that depends on another calculation A, make sure that calculation A comes before calculation B in the field order. For more information see the subsection on field order and chaining in Define dataset fields.

You can use fields of any type in an eval expression field definition. For example, you could create an eval expression field that uses an auto-extracted field and another eval expression field in its eval expression. It will work as long as those fields are listed above the one you're creating.

When you create an eval expression field that uses the values of other fields in its definition, you can optionally "hide" those other fields by setting their Flag to Hidden. This ensures that only the final eval expression value is available to your Pivot users.
Add a lookup field

You can add a lookup field to any dataset in your data model. This is a field that is added to the data model through a lookup. A lookup matches fields in events to fields in a lookup table and then adds corresponding fields from that lookup table to those same events.

To create a lookup field, you must have a lookup definition defined in Settings > Lookups > Lookup definitions. The lookup definition specifies the location of the lookup table and identifies the matching fields as well as the fields that are returned to the events.

For more information about lookup types and creation, see About lookups.

Any lookup table files and lookup definitions that you use in your lookup field definition must have the same permissions as the data model. If the data model is shared globally to all apps, but the lookup table file or definition is private, the lookup field will not work. A data model and the lookup table files and lookup definitions that it is associated with should all have the same permission level.

1. In the Data Model Editor, open the dataset you’d like to add a lookup field to.
2. Click Add Field and select Lookup.
   This takes you to the Add Fields with a Lookup page.
3. Under Lookup Table, select the lookup table that you intend to match an input field to. All of the values in the Lookup Table list are lookup definitions that were previously defined in Settings.

   When you select a valid lookup table, the Input and Output sections of the page are revealed and populated. The Output section should display a list of all of the columns in the selected Lookup Table.
4. Under Input, define your lookup input fields. Choose a Field in Lookup (a field from the Lookup Table that you’ve chosen) and a corresponding Field from the dataset you’re editing.

   The Input lookup table field/value combination is the key that selects rows in the lookup table. For each row that this input key selects, you can bring in output field values from that row and add them to matching events.

   For example, your dataset may have a productId field in your lookup table that matches an auto-extracted Product Id field in your dataset event data. The lookup table field and the dataset field should have the same (or very similar) value sets. In other words, if you have a row in your lookup
table where productId has a value of PD3Z002, there should be events in your dataset where the Product ID = PD3Z002. Those matching events will be updated with output field/value combinations from the row where productId has a value of PD3Z002. See "Example of a lookup field setup," below, for a detailed step-by-step explanation of this process.

In cases where multiple lookup table rows are matched by a particular input key, field values from the first matching row are returned. To narrow down the set of rows that are matched, you can optionally define multiple pairs of input fields. For a row to be selected, all of these input keys must match. You cannot reuse Field in Lookup values when you have multiple inputs.

5. Under Output, determine which fields from the lookup will be added to eligible events in your dataset as new lookup fields. You should find a list of fields here, pulled from the columns in the lookup table that you’ve chosen. Start by selecting the fields that you would like to add to your events. Any lookup fields that you’ve designated as inputs will be unavailable. You must define at least one output field in order for the lookup field definition to be valid.

If you do not find any fields here there may be a problem with the designated Lookup Table.

6. Under Field Name, provide the field name that the lookup field should have in your data. Field Name values cannot include whitespace, single quotes, double quotes, curly braces, or asterisks.

7. Under Display Name provide the display name for the lookup field in the Data Model Editor and in Pivot. Display Name values cannot include asterisk characters.

8. Set appropriate Type and Flags values for each lookup field that you define.

For more information about the Type field, see the subsection "Marking fields as hidden or required" in the Define dataset fields topic.

9. (Optional) Click Preview to verify that the output fields are being added to qualifying events. Qualifying events are events whose input field values match up with input field values in the lookup table). See "Preview lookup fields," below, for more information.

10. If you’re satisfied that the lookup is working as expected, click Save to save your fields and return to the Data Model Builder. The new lookup fields will be added to the bottom of the dataset field list.
Preview lookup fields

After you set up your lookup field, you can click **Preview** to see whether the lookup fields are being added to qualifying events (events where the designated input field values match up with corresponding input field values in the lookup table). Splunk Web displays the results in two or more tabbed pages.

The first tab shows a sample of the events returned by the underlying search. New lookup fields should appear to the right of the first column (the `_time` column). If you do not see any values in the lookup field columns in the first few pages it could indicate that these values are very rare. You can check on this by looking at the remaining preview tab(s).

Splunk Web displays a tab for each lookup field you select in the **Output** section. Each field tab provides a quick summary of the value distribution in the chosen sample of events. It’s set up as a top values list, organized by **Count** and percentage.
Example of a lookup field setup

Let's say the following things are true:

- **You have a data model dataset with an auto-extracted field called **Product ID** and another auto-extracted field named **Product Name**. You would like to use a lookup table to add a new field to your dataset that provides the product price.

- **You have a .csv file called product_lookup**. This table includes several fields related to products, including productId and product_name (which have very similar value sets to the similarly-named fields in your dataset), as well as price, which is the field in the lookup table that you want to add to your dataset as a lookup field.

- **You know that there are a few products that have the same **Product Name** but different **Product ID** values and prices**. This means you can't set up a lookup definition that depends solely on Product Name as the input field, because it will try to apply the same price value from the lookup table to two or more products. You'll have to design a lookup field definition that uses both Product Name and Product ID as input fields, matching each combination of values in your matching events to rows in the lookup table that have the same name/ID combinations.

If this is the case, here's what you do to get price properly added to your dataset as an field.

1. In Settings, create a CSV lookup definition that points at the product_lookup.csv lookup file. Call this lookup definition product_lookup.
2. Select Settings > Data Models and open the Data Model Editor for the dataset you want to add the lookup field to.
3. Click Add Field and select Lookup.
   The Edit Fields with a Lookup page opens.
4. Under Lookup Table select product_lookup.
   All of the fields tracked in the lookup table will appear under Output.
5. Under Input, define two Field in Lookup/Field in Dataset pairs. The first pair should have a Field in Lookup value of ProductId and a Field in Dataset value of Product ID. The second pair should have a Field in Lookup value of product_name and a Field in Dataset value of Product Name.
   The first pair matches the lookup table's productId field with your dataset's Product ID field. The second pair matches the lookup table's product_name field with your dataset's Product Name field. Notice that when you do this, under Output the rows for the
productID and product_name fields become unavailable.

6. Under Output, select the checkbox for the price field.
   This setting specifies that you want to add it to the events in your dataset that have matching input fields.

7. Give the price field a Display Name of Price.
   The price field should already have a Type value of Number.

8. Click Preview to test whether price is being added to your events.
   The preview events appear in table format, and the price field is the second column after the timestamp.

9. If the price field shows up as expected in the preview results, click Save to save the lookup field.

Now your Pivot users will be able to use Price as a field option when building Pivot reports and dashboards.

Add a regular expression field

You can add a regular expression field to any dataset in your data model. Regular expression fields turn the named groups in regular expression strings into separate data model fields. You can arrange for the regular expression to extract fields from the _raw event text as well as specific field values.

1. In the Data Model Editor, open the dataset you’d like to add a regular expression field to.
   For an overview of the Data Model Editor, see Design data models.

2. Click Add Field and select Regular Expression.
   This takes you to the Add Fields with a Regular Expression page.

3. Under Extract From select the field that you want to extract from.
   The Extract From list should include all of the fields currently found in your dataset, with the addition of _raw. If your regular expression
is designed to extract one or more fields from values of a specific field, choose that field from the Extract From list. On the other hand, if your regular expression is designed to parse the entire event string, choose _raw from the Extract From list.

4. Provide a Regular Expression.
   The regular expression must have at least one named group. Each named expression in the regular expression is extracted as a separate field. Field names cannot include whitespace, single quotes, double quotes, curly braces, or asterisks. After you provide a regular expression, the named group(s) appear under Field(s).
   
   Note: Regular expression fields currently do not support sed mode or sed expressions.

5. (Optional) Provide different Display Name values for the field(s).
   Field Display Name values cannot include asterisk characters.

6. (Optional) Correct field Type values.
   They will be given String by default.

7. (Optional) Change field Flag values to whatever is appropriate for your needs.

8. (Optional) Click Preview to get a look at how well the fields are represented in the dataset.

   For more information about previewing fields, see "Preview regular expression field representation," below.

9. Click Save to save your changes.
   You will be returned to the Data Model Editor. The regular expression fields will be added to the list of calculated dataset fields.

For a primer on regular expression syntax and usage, see Regular-Expressions.info. You can test your regex by using it in a search with the rex search command. Splunk also maintains a list of useful third-party tools for writing and testing regular expressions.

**Preview regular expression field representation**

When you click Preview after defining one or more field extraction fields, Splunk software runs the regular expression against the datasets in your dataset that have the Extract From field you've selected (or against raw data if you're extracting from _raw) and shows you the results. The preview results appear underneath the setup fields, in a set of four or more tabbed pages. Each of these tabs shows you information taken from a sample of events in the dataset. You can determine how this sample is determined by selecting an option from the
Sample list, such as *First 1000 events* or *Last 24 hours*. You can also determine how many events appear per page (default is 20).

If the preview doesn't return any events it could indicate that you need to adjust the regular expression, or that you have selected the wrong Extract From field.

**The All tab**

The All tab gives you a quick sense of how prevalent events that match the regular expression are in the event data. You can see an example of the All tab in action in the screen capture near the top of this topic.

It shows you an unfiltered sample of the events that have the Extract From field in their data. For example, if the Extract From field you've selected is `uri_path` this tab displays only events that have a `uri_path` value.

The first column indicates whether the event matched the regular expression or not. Events that match have green checkmarks. Non-matching events have red "x" marks.

The second column displays the value of the Extract From field in the event. If the Extract From field is `_raw`, the entire event string is displayed. The remaining columns display the field values extracted by the regular expression, if any.

**The Match and Non-Match tabs**

The Match and Non-Match tabs are similar to the All tab except that they are filtered to display either just events that *match* the regular expression or just events that *do not match* the regular expression. These tabs help you get a better sense of the field distribution in the sample, especially if the majority of events in the sample fall in either the matching or non-matching event set.
The field tab(s)

Each field named in the regular expression gets its own tab. A field tab provides a quick summary of the value distribution in the chosen sample of events. It's set up as a top values list, organized by **Count** and percentage. If you don't see the values you're expecting, or if the value distribution you are seeing seems off to you, this can be an indication that you need to fine-tune your regular expression.

You can also increase the sample size to find rare field values or values that appear further back in the past. In the example below, setting **Sample** to *First 10,000 events* uncovered a number of values for the *path* field that do not appear when only the first 1,000 events are sampled.

The top value tables in field tabs are drilldown-enabled. You can click on a row to see all of the events represented by that row. For example, if you are looking at the *path* field and you see that there are 6 events with the path `/numa/`, you can click on the `/numa/` row to go to a list that displays the 6 events where `path="/numa/"`.

Add a geo IP field

You can add a Geo IP field to any dataset in your data model that already has a field with a **Type** of *ipv4* in its field list. The *ipv4* field must appear *above* the location for the Geo IP field, and it cannot already be in use for a different Geo IP field calculation.

The Geo IP field is a type of lookup. It reads the IP address values in your dataset's events and can add the related longitude, latitude, city, region, and country values to those events.

1. In the Data Model Editor, open the dataset you'd like to add a field to.
2. Click **Add Field** and select *Geo IP* to define a Geo IP field. The “Add Geo Fields with an IP Lookup” page opens.
3. Choose the IP field that you want to match, if more than one exists for the selected dataset.
4. Select the fields that you want to add to your dataset.
5. (Optional) Rename selected fields by changing their Display Name. Display names cannot include asterisk characters.
6. (Optional) Click Preview to verify that the GeoIP field is correctly updating your events with the GeoIP fields that you have selected.
   You should see events in table format with the new GeoIP field(s) included as columns. For example, if you’re working with an event-based dataset and you’ve selected the City, Region, and Country GeoIP fields, the preview event table should display City, Region, and Country columns to the right of the first column (_time).
   The preview pane has two tabs. Events is the default tab. It presents the events in table format. Select the Values tab to review the distribution of GeoIP field values among your events. If you’re not seeing the range of values you’re expecting, try increasing the preview event sample. By default this sample is set to the first thousand events. You might increase it by setting the Sample value to First 10,000 events or Last 7 days.

7. Click Save to save your changes.
   You will be returned to the Data Model Editor. The Geo IP fields that you have defined will be added to the dataset’s set of Calculated fields.
   **Note:** Geo IP fields are added to your dataset as required fields, and their Type values are predetermined. You cannot change these values.
Use data summaries to accelerate searches

Overview of summary-based search acceleration

Splunk Enterprise is capable of generating reports on massive amounts of data. However, the amount of time it takes to compute such reports is directly proportional to the number of events they summarize. Plainly put, it can take a lot of time to report on very large datasets. If you only have to do this on an occasional basis, the length of time may not be an issue. But running such reports on a regular schedule (or using them as the basis for panels in popular dashboards) is impractical—and this impracticality only increases exponentially as more and more users in your organization run similar reports.

To efficiently report on large volumes of data, you need to create data summaries that are populated by the results of background runs of the search upon which the report is based. When you next run the report against data that has been summarized in this manner, it should complete significantly faster, because the summaries are much smaller than the original events from which they were generated.

There are three data summary creation methods:

- **Report acceleration** - Uses automatically-created summaries to speed up completion times for certain kinds of reports.
- **Data model acceleration** - Uses automatically-created summaries to speed up completion times for pivot-based reports and dashboard panels.
- **Summary indexing** - Enables acceleration of searches and reports through the manual creation of separate summary indexes that exist separately from your main indexes.

Report acceleration

**Report acceleration** is used to accelerate individual reports. It’s easy to set up for any transforming search or report that runs over a large dataset.

In early versions of Splunk software, summary indexing was used to accelerate reports. Report acceleration is preferable over summary indexing for the following reasons:

- **Kicking off report acceleration is as easy as clicking a checkbox and selecting a time range.** Everything after that happens behind the scenes.
Subsequent runs of accelerated reports should complete faster as long as they're run (at least partially) within their selected time ranges. For summary indexing you need to design a search to populate the index that includes special search commands; you may need to create the summary index as well.

- **Splunk software automatically shares report acceleration summaries with similar searches.** Say an employee named Mary sets up report acceleration for a report, which leads to Splunk software building a summary for it. Then, a few days later, Joe designs a report that is nearly identical to Mary’s report, with a few variations. When Joe turns on report acceleration for the report and saves it, Splunk software automatically assigns it to the summary that was already built for Mary’s report, which means that Joe won’t need to wait for the summary to be built.

- **Report acceleration features automatic backfill.** If for some reason you have a data interruption, Splunk software can detect this and automatically update or rebuild your summaries as appropriate.

- **Report acceleration summaries are stored alongside the buckets in your indexes.** Summary indexes, on the other hand, reside on the search head. Storing summaries in indexes at the bucket level enables Splunk Enterprise to easily handle the dilemma of late-arriving events—something that can force full rebuilds of summary indexes. Because summaries can simultaneously span both hot and warm buckets, they can summarize late-arriving data, because such data can only be added to hot buckets.

It’s important to note that not all searches qualify for report acceleration. Only searches that utilize **transforming commands**—searches that transform their results into statistical tables and charts—are eligible. In addition, any commands used in the search before the transforming command must be **streaming commands**. This limitation is related to the fact that the summaries are built at the index level rather than the search head.

In Splunk Web, you can enable report acceleration for an eligible search when you save it as a report. You can enable report acceleration for an eligible existing report by:

- On the Reports page, expanding a row for a report and clicking **Edit** to open the **Edit Acceleration dialog**. If your report qualifies for acceleration and your permissions allow for report acceleration, the Edit Acceleration dialog will display a checkbox labeled Accelerate Report. Select it. The Summary Range field should appear. Select the range of time over which you plan to run the report, then click **Save**.
- in **Settings > Searches and reports** opening the detail page for a report, clicking **Accelerate this search** and setting a **Summary range**.
See Accelerate reports.

You use the Report acceleration summaries page in System to review and manage the summaries created through report acceleration.

See Manage report acceleration. This topic also explains how summaries work and includes examples of qualifying and non-qualifying searches.

**When should I use report acceleration?**

Report acceleration is good for just about any slow-completing report that has 100k or more hot bucket events and which meets the qualifying conditions outlined above.

**Data model acceleration**

You use data model acceleration to accelerate all of the fields defined in a data model. When a data model is accelerated, any pivot or report generated by that data model should complete much quicker than it would without the acceleration, even if the data model represents a significantly large dataset.

There are two types of data model acceleration, ad hoc and persistent. Ad hoc acceleration applies to a single dataset, is run over all time, and exists for the duration of a user’s pivot session, while persistent acceleration is turned on by an admin, happens in the background, and can be scoped to shorter time ranges such as a week or a month. Persistent acceleration is used any time a search is run against a dataset in an acceleration-enabled data model.

Data model acceleration makes use of Splunk’s high performance analytics store (HPAS) technology, which, in a manner similar to that of report acceleration, builds summaries alongside the buckets in your indexes. Also like report acceleration, persistent data model acceleration is easy to enable; you just click a checkbox for the data model you want to accelerate and select a summary range. Once you do this, Splunk software starts building a summary that spans the indicated range. When the summary is complete, any pivot, report, or dashboard panel that uses an accelerated data model dataset will run against the summary rather than the full array of _raw whenever possible, and result return time should be improved by a significant amount.

There are restrictions for persistent data model acceleration.

- **Persistent data model acceleration only applies to event dataset hierarchies and search dataset hierarchies based on root search**
datasets that only include streaming commands). Dataset hierarchies based on root search datasets that nonstreaming commands and root transaction datasets cannot be accelerated.

- All data model datasets can benefit from "ad hoc" data model acceleration. See the subsection on this below.
- **Once a data model is persistently accelerated it cannot be edited.** After you enable acceleration for a data model, the only way to edit it is to disable its acceleration.
- **By default only users with admin permissions can persistently accelerate data models.**
- **Data models that are private cannot be persistently accelerated.** You must share a data model with users of at least one app to make it eligible for acceleration.

In Splunk Web, you can enable data model acceleration for an eligible data model on the Data Models management page, which you can access in a variety of ways (including navigating to Settings > Data Models).

For more information about enabling persistent data model acceleration, see Manage data models.

For technical background information on data model acceleration and how the high performance analytics store works behind the scenes, see Accelerate data models.

**Ad hoc data model acceleration**

Ad hoc data model acceleration is a process that runs behind the scenes for all data model datasets that are not "persistently" accelerated beforehand. Unlike persistent data model acceleration, ad hoc data model acceleration applies to all dataset types, including root search datasets, root transaction datasets, and their children.

Whenever you build a pivot based on an dataset that isn't already accelerated, Splunk software will use ad hoc data model acceleration to build a temporary acceleration summary in a dispatch directory that exists only while you define the pivot in the Pivot Editor. The result is that as you fine-tune a particular pivot in the Pivot Editor you'll find that the pivot performance improves, returning results faster than it did when you first entered the editor.

This isn't as good as persistent data model acceleration, where summaries for the data model datasets are maintained on an ongoing basis, ensuring speedy performance from the moment you enter the Pivot Editor--but it's still helpful.
See Accelerate data models.

**When should I use persistent data model acceleration?**

If you are struggling with slow-completing pivots in the Pivot Editor and the source datasets for those pivots belongs to a topmost root event dataset hierarchy, you should consider enabling acceleration for that data model. It will ensure that the pivots based on those datasets return results faster than they would otherwise.

Furthermore, any report or dashboard panel that references a persistently accelerated data model dataset will also get this acceleration benefit (this will not happen with ad hoc data model acceleration).

**Report acceleration versus data model acceleration**

In general, data model acceleration is faster than report acceleration. However, there are specific kinds of searches that allow report acceleration to come out ahead of data model acceleration.

The more aggregating your transforming search is, the faster it can be. Report acceleration is especially fast when it runs with a search that aggregates down to one item per index bucket. For example, if you get a couple of billion events per day and you just want a monthly count and average, your return report acceleration will be better at this than data model acceleration. In this case you would be maxing out the aggregation capabilities of report acceleration.

Report acceleration and data model acceleration go about accelerating searches in similar ways. They both automatically preprocess events on the indexers, and they both create bucket-level acceleration summaries. But the general advantage for data model acceleration lies in how its summaries differ from those created by report acceleration.

Report acceleration is designed to create summaries that include precalculated statistics. Data model acceleration, on the other hand, builds its summaries in a format that is much more efficient to read, enabling Splunk software to calculate the statistics on demand without giving up performance. So if the search is a relatively complicated one you'll be better off going with data model acceleration.

**Summary indexing**

**Summary indexing** is a method you can use to speed up long-running reports that do not qualify for report acceleration, such as reports that use search
commands that are not **streamable** before the transforming command. It's similar to report acceleration in that it involves populating a data summary with the results of a search, but in this case the data summary is actually a special **summary index** that is built and stored on the search head. This summary index is populated by a scheduled report that is *based on* the report that you'd like to accelerate and which has **Enable** selected for summary indexing in **Settings > Searches and Reports**.

For example, if the report you want to accelerate uses a **transforming command**, you can populate its summary index with a report that swaps the transforming command with a similar "si-" prefix summary indexing transforming command: `sichart, sitimechart, sistats, sitop, and sirare`.

There are two topics on summary indexing setup.

- **Use summary indexing for increased reporting efficiency** shows you the easy way of setting up summary indexes, with scheduled searches that use `si-` commands.
- **Configure summary indexes** covers the tricky and difficult method of summary index setup with `addinfo`, `collect`, and `overlap` commands. You should only use this latter method if you're comfortable setting up searches that take aggregated statistics into account.

Summary indexing volume is not counted against your license, even if you have multiple summary indexes.

**When should I use summary indexing?**

If the report you're using qualifies for report acceleration, it's almost always preferable to use that method of speeding up the performance of large data volume searches.

You might want to use summary indexing instead of report acceleration if:

- The primary report you want to accelerate includes nonstreamable commands before a transforming command (just as with report acceleration, reports that populate summary indexes must involve transforming commands).
- You would like to run any report against a particular summary index, simply by including `index=<summary_index_name>` in your search string. (Under report acceleration, Splunk software automatically decides whether a report can run against a specific data summary.)
• Your raw data rolls more frequently than your reporting window (e.g. your retention policy is 6 months but you want to power a panel in a dashboard from data for the last year). Summary indexes generally take up less space than the events they aggregate and can be retained separately and for greater durations.

**Batch mode search**

Batch mode search is a feature that improves the performance and reliability of transforming searches. For transforming searches that don’t require the events to be time-ordered, running in batch mode means that the search executes bucket-by-bucket (in batches), rather than over time. In certain reporting cases, this means that the transforming search can complete faster. Additionally, batch mode search improves the reliability for long-running distributed searches, which can fail when an indexer goes down while the search is running. In this case, Splunk software attempts to reconnect to the missing peer and retry the search.

Transforming searches that meet the criteria for batch mode search include:

- Generating and transforming searches (stats, chart, etc.) that do not include the localize or transaction commands in the search.
- Searches that are not real-time and not summarizing searches.
- Non-distributed searches that are not stateful streaming. (A streamstats search is an example of a stateful streaming search.)

Batch mode search is invoked from the configuration file, in the [search] stanza of limits.conf. Use the search inspector to determine whether or not a transforming search is running in batch mode.

See Configure batch mode search.

**Manage report acceleration**

Report acceleration is the easiest way to speed up transforming searches and reports that take a long time to complete because they have to cover a large volume of data. You enable acceleration for a transforming search when you save it as a report. You can also accelerate report-based dashboard panels that use a transforming search.

This topic covers various aspects of report acceleration in more detail. It includes:
• A quick guide to enabling automatic acceleration for a transforming report.
• Examples of qualifying and nonqualifying reports (only specific kinds of reports qualify for report acceleration).
• Details on how report acceleration summaries are created and maintained
• An overview of the Report acceleration summaries page in Settings, which you can use to review and maintain the data summaries that are used for automatic report acceleration.

Restrictions on report acceleration

You cannot accelerate a report if:

• You created it through Pivot. Pivot reports are accelerated via data model acceleration. See Manage data models.
• Your permissions do not enable you to accelerate searches. You cannot accelerate reports if your role does not have the schedule_search and accelerate_search capabilities.
• Your role does not have write permissions for the report.
• The search that the report is based upon is disqualified for acceleration. For more information, see How reports qualify for acceleration.

In addition, be careful when accelerating reports whose base searches include tags, event types, search macros, and other knowledge objects whose definitions can change independently of the report after the report is accelerated. If this happens, the accelerated report can return invalid results.

If you suspect that your accelerated report is returning invalid results, you can verify its summary to see if the data contained in the summary is consistent. See Verify a summary.

Enabling report acceleration

You can enable report acceleration when you create a report, or later, after the report is created.

For a more thorough description of this procedure, see Create and edit reports, in the Reporting Manual.

Enabling report acceleration when you create a report

You can enable report acceleration for qualifying reports.

Prerequisites
• How reports qualify for acceleration

Steps

1. In Search, run a search that qualifies for report acceleration.
2. Select Save As > Report.
3. Give your report a Name and optionally, a Description.
4. Click Save to save the search as a report.
5. Select Acceleration.
   You can only accelerate the report if the report qualifies for acceleration and your permissions allow you to accelerate reports. To be able to accelerate reports your role has to have the schedule_search and accelerate_search capabilities.
7. Select a Summary Range.
   Base your selection on the range of time over which you plan to run the report. For example, if you only plan to run the report over periods of time within the last seven days, choose 7 Days.
8. Click Save to save your acceleration settings.

Enabling report acceleration for an existing report

You can enable acceleration for a qualifying existing report.

Prerequisites

• How reports qualify for acceleration

Steps

1. On the Reports listing page, find a report that you want to accelerate.
2. Expand the report row by clicking on the > symbol in the first column.
   The Acceleration line item displays the acceleration status for the report. Its value will be Disabled if it is not accelerated.
3. Click Edit
   You can only accelerate the report if the report qualifies for acceleration and your permissions allow you to accelerate reports. To be able to accelerate reports your role has to have the schedule_search and accelerate_search capabilities.
5. Select a Summary Range.
   Base your selection on the range of time over which you plan to run the report. For example, if you only plan to run the report over periods of time...
within the last seven days, choose 7 Days.
6. Click Save to save your acceleration settings.

Alternatively, you can enable report acceleration for an existing report at Settings > Searches, Reports, and Alerts.

**After you enable acceleration for a report**

When you enable acceleration for your report, the Splunk software begins building a report acceleration summary for the report if it determines that the report would benefit from summarization. To find out whether your report summary is being constructed, go to Settings > Report Acceleration Summaries. If the Summary Status is stuck at 0% complete for an extended amount of time, the summary is not being built.

See Conditions under which Splunk software cannot build or update a summary.

Once the summary is built, future runs of an accelerated report should complete faster than they did before. See the subtopics below for more information on summaries and how they work.

**Note:** Report acceleration only works for reports that have Search Mode set to Smart or Fast. If you select the Verbose search mode for a report that benefits from report acceleration, it will run as slow as it would if no summary existed for it. Search Mode does not affect searches powering dashboard panels.

For more information about the Search Mode settings, see Search modes in the Search Manual.

**How reports qualify for acceleration**

For a report to qualify for acceleration its search must meet three criteria:

- The search string must use a transforming command (such as chart, timechart, stats, and top).
- If the search string has any commands before the first transforming command, they must be streamable.
- The search cannot use event sampling.

**Note:** You can use non-streaming commands after the first transforming command and still have the report qualify for automatic acceleration. It's just non-streaming commands before the first transforming command that disqualify the report.
For more information about event sampling, see Event sampling in the Search Manual.

**Examples of qualifying search strings**

Here are examples of search strings that qualify for report acceleration:

```
index=_internal | stats count by sourcetype

index=_audit search=* | rex field=search "'(.*?)'" | chart count by user

search test foo bar | bin _time span=1d | stats count by _time x y

index=_audit | lookup usertogroup user OUTPUT group | top searches by group
```

**Examples of nonqualifying search strings**

And here are examples of search strings that do not qualify for report acceleration:

**Reason the following search string fails:** This is a simple event search, with no transforming command.

```
index=_internal metrics group=per_source_thruput
```

**Reason the following search string fails:** `eventstats` is not a transforming command.

```
index=_internal sourcetype=splunkd *thruput | eventstats avg(kb) as avgkb by group
```

**Reason the following search string fails:** `transaction` is not a streaming command. Other non-streaming commands include `dedup`, `head`, `tail`, and any other search command that is not on the list of streaming commands.

```
index=_internal | transaction user maxspan=30m | timechart avg(duration) by user
```

**Search strings that qualify for report acceleration but won’t get much out of it**

In addition, you can have reports that technically qualify for report acceleration, but which may not be helped much by it. This is often the case with reports with
high data cardinality—something you'll find when there are two or more transforming commands in the search string and the first transforming command generates many (50k+) output rows. For example:

```
index=* | stats count by id | stats avg(count) as avg, count as distinct_ids
```

**Set report acceleration summary time ranges**

Report acceleration summaries span an approximate range of time. You determine this time range when you choose a value from the **Summary Range** list. At times, a report acceleration summary can have a store of data that that slightly exceeds its summary range, but the summary never fails to meet that range, except while it is first being created.

For example, if you set a summary range of **7 days** for an accelerated report, a data summary that approximately covers the past seven days is created. Every ten minutes, a search is run to ensure that the summary always covers the selected range. These maintenance searches add new summary data and and remove older summary data that passes out of the range.

When you then run the accelerated report over a range that falls within the past 7 days, the report searches its summary rather than the source index (the index the report originally searched). In most cases the summary has far less data than the source index, and this—along with the fact that the report summary contains precomputed results for portions of the search pipeline—means that the report should complete faster than it did on its initial run.

When you run the accelerated report over a period of time that is only partially covered by its summary, the report does not complete quite as fast because the Splunk software has to go to the source index for the portion of the report time range that does not fall within the summary range.

If the **Summary Range** setting for a report is **7 Days** and you run the report over the last 9 days, the Splunk software only gets acceleration benefits for the portion of the report that covers the past 7 days. The portion of the report that runs over days 8 and 9 will run at normal speed.

Keep this in mind when you set the **Summary Range** value. If you always plan to run a report over time ranges that exceed the past 7 days, but don't extend further out than 30 days, you should select a **Summary Range** of **1 month** when you set up report acceleration for that report.
How the Splunk platform builds report acceleration summaries

After you enable acceleration for an eligible report, Splunk software determines whether it will build a summary for the report. A summary for an eligible report is generated only when the number of events in the hot bucket covered by the chosen Summary Range is equal to or greater than 100,000. For more information, see the subtopic below titled "Conditions under which the Splunk platform cannot build or update a summary."

When Splunk software determines that it will build a summary for the report, it begins running the report to populate the summary with data. When the summary is complete, the report is run every ten minutes to keep the summary up to date. Each update ensures that the entire configured time range is covered without a significant gap in data. This method of summary building also ensures that late-arriving data will be summarized without complication.

**Report acceleration summaries can take time to build and maintain**

It can take some time to build a report summary. The creation time depends on the number of events involved, the overall summary range, and the length of the summary timespans (chunks) in the summary.

You can track progress toward summary completion on the Report Acceleration Summaries page in Settings. On the main page you can check the Summary Status to see what percentage of the summary is complete.

**Note:** Just like ordinary scheduled reports, the reports that automatically populate report acceleration summaries on a regular schedule are managed by the report scheduler. By default, the report scheduler is allowed to allocate up to 25% of its total search bandwidth for report acceleration summary creation.

The report scheduler also runs reports that populate report acceleration summaries at the lowest priority. If these "auto-summarization" reports have a scheduling conflict with user-defined alerts, summary-index reports, and regular scheduled reports, the user-defined reports always get run first. This means that you may run into situations where a summary is not created or updated because reports with a higher priority are running.

For more information about the search scheduler see the topic "Configure the priority of scheduled reports," in the Reporting Manual.
Use parallel summarization to speed up creation and maintenance of report summaries

If you feel that the summaries for some of your accelerated reports are building or updating too slowly, you can turn on parallel summarization for those reports to speed the process up. To do this you add a parameter in `savedsearches.conf` for the report or reports in question.

Under parallel summarization, multiple search jobs are run concurrently to build a report acceleration summary. It also runs the same number of concurrent searches on a 10 minute schedule to maintain those summary files. Parallel summarization decreases the amount of time it takes for report acceleration summaries to be built and maintained.

There is a cost for this improvement in summarization search performance. The concurrent searches count against the total number of concurrent search jobs that your Splunk deployment can run, which means that they can cause increased indexer resource usage.

1. Open the `savedsearches.conf` file that contains the report that you want to update summarization settings for.

2. Locate the stanza for the report.

3. Add `auto_summarize.max_concurrent = 2` if that parameter is not present in the stanza.

4. Save your changes.

If you turn on parallel summarization for some reports and find that your overall search performance is impacted, either because you have too many searches running at once or your concurrent search limit is reached, you can easily restore the `auto_summarize.max_concurrent` value of your accelerated reports back to 1.

In general we do not recommend increasing `auto_summarize.max_concurrent` to a value higher than 2. However, if your Splunk deployment has the capacity for a large amount of search concurrency, you can try setting `auto_summarize.max_concurrent` to 3 or higher for selected accelerated reports.

See:

- "Accomodate many simultaneous searches" in the Capacity Planning Manual for information about the impact of concurrent searches on search
performance.
• "Configure the priority of scheduled reports" for more information about
how the the concurrent search limit for your implementation is determined.

**Summary data is divided into chunks with regular timespans**

As Splunk software builds and maintains the summary, it breaks the data up into
chunks to ensure statistical accuracy, according to a "timespan" determined
automatically, based on the overall summary range. For example, when the
summary range for a report is 1 month, a timespan of 1d (one day) might be
selected.

A summary timespan represents the smallest time range for which the summary
contains statistically accurate data. If you are running a report against a summary
that has a one hour timespan, the time range you choose for the report should be
evenly divisible by that timespan, if you want the report to use the summarized
data. When you are dealing with a 1h timespan, a report that runs over the past
24 hours would work fine, but a report running over the past 90 minutes might not
be able to use the summarized data.

**Summaries can have multiple timespans**

Report acceleration summaries might be assigned multiple timespans if
necessary to make them as searchable as possible. For example, a summary
with a summary range of 3 months can have timespans of 1mon and 1d. In
addition, extra timespans might be assigned when the summary spans more than
one index bucket and the buckets cover very different amounts of time. For
example, if a summary spans two buckets, and the first bucket spans two months
and the next bucket spans 40 minutes, the summary will have chunks with 1d
and 1m timespans.

**You can manually set summary timespans (but we don’t recommend it)**

You can set summary timespans manually at the report level in
savedsearches.conf by changing the value of the auto_summarize.timespan
parameter. If you do set your summary timespans manually, keep in mind that
very small timespans can result in extremely slow summary creation times,
especially if the summary range is long. On the other hand, large timespans can
result in quick-building summaries that cannot not be used by reports with short
time ranges. In almost all cases, for optimal performance and usability it’s best to
let Splunk software determine summary timespans.
The way that Splunk software gathers data for accelerated reports can result in a lot of files over a very short amount of time

Because report acceleration summaries gather information for multiple timespans, many files can be created for the same summary over a short amount of time. If file and folder management is an issue for you, this is something to be aware of.

For every accelerated report and search head combination in your system, you get:

- 2 files (data + info) for each 1-day span
- 2 files (data + info) for each 1-hour span
- 2 files (data + info) for each 10-minute span
- SOMETIMES: 2 files (data + info) for each 1-minute span

So if you have an accelerated report with a 30-day range and a 10 minute granularity, the result is:

\[(30 \times 1 + 30 \times 24 + 30 \times 144) \times 2 = 10,140 \text{ files}\]

If you switch to a 1 minute granularity, the result is:

\[(30 \times 1 + 30 \times 24 + 30 \times 144 + 30 \times 1440) \times 2 = 96,540 \text{ files}\]

If you use Deployment Monitor, which ships with 12 accelerated reports by default, an immediate backfill could generate between 122k and 1.2 million files on each indexer in $SPLUNK_HOME/var/lib/splunk/_internaldb/summary, for each search-head on which it is enabled.

Where report acceleration summaries are created and stored

The Splunk software creates report acceleration summaries on the indexer, parallel to the bucket or buckets that cover the range of time over which the summary spans. For example, for the "index1" index, they reside under $SPLUNK_HOME/var/lib/splunk/index1/summary.

Data model acceleration summaries are stored in the same manner, but in directories labeled datamodel_summary instead of summary.

By default, indexer clusters do not replicate report acceleration and data model acceleration summaries. This means that only primary bucket copies will have associated summaries.
If your peer nodes are running version 6.4 or higher, you can configure the cluster master node so that your indexer clusters replicate report acceleration summaries. All searchable bucket copies will then have associated summaries. This is the recommended behavior.


**Configure size-based retention for report acceleration summaries**

Do you set size-based retention limits for your indexes so they do not take up too much disk storage space? By default, report acceleration summaries can theoretically take up an unlimited amount of disk space. This can be a problem if you're also locking down the maximum data size of your indexes or index volumes. The good news is that you can optionally configure similar retention limits for your report acceleration summaries.

**Note:** Although report acceleration summaries are unbounded in size by default, they are tied to raw data in your warm and hot index buckets and will age along with it. When events pass out of the hot/warm buckets into cold buckets, they are likewise removed from the related summaries.

**Important:** Before attempting to configure size-based retention for your report acceleration summaries, you should first understand how to use volumes to configure limits on index size across indexes, as many of the principles are the same. For more information, see "Configure index size" in Managing Indexers and Clusters.

By default, report acceleration summaries live alongside the hot and warm buckets in your index at `homePath/../summary/`. In other words, if in `indexes.conf` the `homePath` for the hot and warm buckets in your index is:

```bash
homePath = /opt/splunk/var/lib/splunk/index1/db
```

Then summaries that map to buckets in that index will be created at:

```bash
homePath/opt/splunk/var/lib/splunk/index1/summary
```

Here are the steps you take to set up size-based retention for the summaries in that index. All of the configurations described are made within `indexes.conf`. 
1. Review your volume definitions and identify a volume (or volumes) that will be the home for your report acceleration summary data.

   If the right volume doesn’t exist, create it.

   If you want to piggyback on a preexisting volume that controls your indexed raw data, you might have that volume reference the filesystem that hosts your hot and warm bucket directories, because your report acceleration summaries will live alongside it.

   However, you could also place your report acceleration summaries in their own filesystem if you want. The only rule here is: You can only reference one filesystem per volume, but you can reference multiple volumes per filesystem.

2. For the volume that will be the home for your report acceleration data, add the `maxVolumeDataSizeMB` parameter to set the volume’s maximum size.

   This lets you manage size-based retention for report acceleration summary data across your indexes.

3. Update your index definitions.

   Set the `summaryHomePath` for each index that deals with summary data. Ensure that the path is referencing the summary data volume that you identified in Step 1.

   `summaryHomePath` overrides the default path for the summaries. Its value should compliment the `homePath` for the hot and warm buckets in the indexes. For example, here’s the `summaryHomePath` that compliments the `homePath` value identified above:

   ```
   summaryHomePath = /opt/splunk/var/lib/splunk/index1/summary
   ```

   This example configuration shows data size limits being set up on a global, per-volume, and per-index basis.

   # Global settings

   # Inheritable by all indexes: No hot/warm bucket can exceed 1 TB.
   # Individual indexes can override this setting. The global
   # summaryHomePath setting indicates that all indexes that do not
explicitly
# define a summaryHomePath value will write report acceleration
summaries
# to the small_indexes # volume.
[global]
homePath.maxDataSizeMB = 1000000
summaryHomePath = volume:small_indexes/$_index_name/summary

#############################################################################
# Volume definitions
#############################################################################

# This volume is designed to contain up to 100GB of summary data and
other
# low-volume information.
[volume:small_indexes]
path = /mnt/small_indexes
maxVolumeDataSizeMB = 100000

# This volume handles everything else. It can contain up to 50
# terabytes of data.
[volume:large_indexes]
path = /mnt/large_indexes
maxVolumeDataSizeMB = 5000000

#############################################################################
# Index definitions
#############################################################################

# The report_acceleration and rare_data indexes together are limited to
100GB, per the
# small_indexes volume.
[report_acceleration]
homePath = volume:small_indexes/report_acceleration/db
coldPath = volume:small_indexes/report_acceleration/colddb
thawedPath = $SPLUNK_DB/summary/thaweddb
summaryHomePath = volume:small_indexes/report_acceleration/summary
maxHotBuckets = 2

[rare_data]
homePath = volume:small_indexes/rare_data/db
coldPath = volume:small_indexes/rare_data/colddb
thawedPath = $SPLUNK_DB/rare_data/thaweddb
summaryHomePath = volume:small_indexes/rare_data/summary
maxHotBuckets = 2

# Splunk constrains the main index and any other large volume indexes
that
# share the large_indexes volume to 50TB, separately from the 100GB of
the
# small_indexes volume. Note that these indexes both use summaryHomePath
to
# direct summary data to the small_indexes volume.

[main]
homePath = volume:large_indexes/main/db
coldPath = volume:large_indexes/main/colddb
thawedPath = $SPLUNK_DB/main/thaweddb
summaryHomePath = volume:small_indexes/main/summary
maxDataSize = auto_high_volume
maxHotBuckets = 10

# Some indexes reference the large_indexes volume with summaryHomePath,
# which means their summaries are created in that volume. Others do not
# explicitly reference a summaryHomePath, which means that the Splunk
# platform
# directs their summaries to the small_indexes volume, per the [global]
# stanza.
[idx1_large_vol]
homePath=volume:large_indexes/idx1_large_vol/db
coldPath=volume:large_indexes/idx1_large_vol/colddb
homePath=$SPLUNK_DB/idx1_large/thaweddb
summaryHomePath = volume:large_indexes/idx1_large_vol/summary
maxDataSize = auto_high_volume
maxHotBuckets = 10
frozenTimePeriodInSecs = 2592000

[other_data]
homePath=volume:large_indexes/other_data/db
coldPath=volume:large_indexes/other_data/colddb
homePath=$SPLUNK_DB/other_data/thaweddb
maxDataSize = auto_high_volume
maxHotBuckets = 10

When a report acceleration summary volume reaches its size limit, the Splunk
volume manager removes the oldest summary in the volume to make room.
When the volume manager removes a summary, it places a marker file inside its
Corresponding bucket. This marker file tells the summary generator not to rebuild
the summary.

Data model acceleration summaries have a default volume called
_splunk_summaries that is referenced by all indexes for the purpose of data
model acceleration summary size-based retention. By default this volume has no
maxVolumeDataSizeMB setting, meaning it has infinite retention.

You can use this preexisting volume to manage data model acceleration
summaries and report acceleration summaries in one place. You would need to:

- have the summaryHomePath reference for your report acceleration
  summaries reference the _splunk_summaries volume.
- set a maxVolumeDataSizeMB value for _splunk_summaries.
For more information about size-based retention for data model acceleration summaries, see "Accelerate data models" in this manual.

**Multiple reports for a single summary**

A single report summary can be associated with multiple searches when the searches meet the following two conditions.

- The searches are identical up to and including the first reporting command.
- The searches are associated with the same app.

Searches that meet the first condition, but which belong to different apps, cannot share the same summary.

For example, these two reports use the same report acceleration summary.

```
sourcetype=access_* status=2* | stats count by price
```

```
sourcetype=access_* status=2* | stats count by price | eval discount = price/2
```

These two reports use different report acceleration summaries.

```
sourcetype=access_* status=2* | stats count by price
```

```
sourcetype=access_* status=2* | timechart by price
```

Two reports that are identical except for syntax differences that do not cause one to output different results than the other can also use the same summary.

These two searches use the same report acceleration summary.

```
sourcetype = access_* status=2* | fields - clientip, bytes | stats count by price
```

```
sourcetype = access_* status=2* | fields - bytes, clientip | stats count by price
```

You can also run non-saved searches against the summary, as long as the basic search matches the populating saved search up to the first reporting command and the search time range fits within the summary span.
You can see which searches are associated with your summaries by navigating to Manager > Report Acceleration Summaries. See "Use the Report Acceleration Summaries Page" in this topic.

Conditions under which the Splunk platform cannot build or update a summary

Splunk software cannot build a summary for a report when either of the following conditions exist.

- The number of events in the hot bucket covered by the chosen Summary Range is less than than 100k. When this condition exists you see a Summary Status warning that says Not enough data to summarize.
- Splunk software estimates that the completed summary will exceed 10% of the total bucket size in your deployment. When it makes this estimation, it suspends the summary for 24 hours. You will see a Summary Status of Suspended.

You can see the Summary Status for a summary in Settings > Report Acceleration Summaries.

If you define a summary and the Splunk software does not create it because these conditions exist, the software checks periodically to see if conditions improve. When these conditions are resolved, Splunk software begins creating or updating the summary.

How can you tell if a report is using its summary?

The obvious clue that a report is using its summary is if you run it and find that its report performance has improved (it completes faster than it did before).

But if that's not enough, or if you aren't sure if there's a performance improvement, you can View search job properties in the Search Manual for a debug message that indicates whether the report is using a specific report acceleration summary. Here's an example:

```
DEBUG: [thething] Using summaries for search,
summary_id=246B0E5B-A8A2-484E-840C-78CB43595A84_search_admin_b7a7b033b6a72b45,
maxtimespan=
```

In this example, that last string of numbers, b7a7b033b6a72b45, corresponds to the Summary ID displayed on the Report Acceleration Summaries page.
Use the Report Acceleration Summaries page

You can review the your report acceleration summaries and even manage various aspects of them with the Report Acceleration Summaries page in Settings. Go to **Settings > Report Acceleration Summaries.**

The main Report Acceleration Summaries page enables you to see basic information about the summaries that you have permission to view.

The **Summary ID** and **Normalized Summary ID** columns display the unique hashes that assigned to those summaries. The IDs are derived from the remote search string for the report. They are used as part of the directory name that is created for the summary files. Click a summary ID or normalized summary ID to view summary details and perform summary management actions. For more information about this detail view, see the subtopic "Review summary details," below.

The **Reports Using Summary** column lists the saved reports that are associated with each of your summaries. It indicates that each report associated with a particular summary will get report acceleration benefits from that summary. Click on a report title to drill down to the detail page for that report.

Check **Summarization Load** to get an idea of the effort that Splunk software has to put into updating the summary. It's calculated by dividing the number of seconds it takes to run the populating report by the interval of the populating report. So if the report runs every 10 minutes (600 seconds) and takes 30 seconds to run, the summarization load is 0.05. If the summarization load is high and the **Access Count** for the summary shows that the summary is rarely used or hasn't been used in a long time, you might consider deleting the summary to reduce the strain on your system.
The **Summary Status** column reports on the general state of the summary and tells you when it was last updated with new data. Possible status values are *Complete, Pending, Suspended, Not enough data to summarize*, or the percentage of the summary that is complete at the moment. If you want to update a summary to the present moment, click its summary ID to go to its detail page and click **Update** to kick off a new summary-populating report.

If the **Summary Status** is *Pending* it means that the summary may be slightly outdated and the search head is about to schedule a new update job for it.

If the **Summary Status** is *Suspended* it means that the report is not worth summarizing because it creates a summary that is too large. Splunk software projects the size of the summary that a report can create. If it determines that a summary will be larger than 10% of the index buckets it spans, it suspends that summary for 24 hours. There's no point to creating a summary, for example, if the summary contains 90% of the data in the full index.

You cannot override summary suspension, but you can adjust the length of time that summaries are suspended by changing the value of the `auto_summarize.suspend_period` attribute in `savedsearches.conf`.

If the **Summary Status** reads *Not enough data to summarize*, it means that Splunk software is not currently generating or updating a summary because the reports associated with it are returning less than 100k events from the **hot buckets** covered by the summary range. For more information, see the subtopic above titled "Conditions under which the Splunk platform cannot build or update a summary."

**Review summary details**

You use the summary details page to view detail information about a specific summary and to initiate actions for that summary. You get to this page by clicking a **Summary ID** on the Report Acceleration Summaries page in Settings.
Summary Status

Under Summary Status you'll see basic status information for the summary. It mirrors the Summary Status listed on the Report Acceleration Summaries page (see above) but also provides information about the verification status of the summary.

If you want to update a summary to the present moment click the Update button under Actions to kick off a new summary-populating report.

No verification status will appear here if you've never initiated verification for the summary. After you initiate verification this status shows the verification percentage complete. Otherwise this status shows the results of the last attempt at summary verification; the possible values are Verified and Failed to verify, with an indication of how far back in the past this attempt took place.

For more information about summary verification, see "Verify a summary," below.

Reports using the summary

The Reports Using This Summary section lists the reports that are associated with the summary, along with their owner and home app. Click on a report title to drill down to the detail page for that report. Similar reports (reports with search strings that all transform the same root search with different transforming commands, for example) can use the same summary.
**Summary details**

The **Details** section provides a set of metrics about the summary.

**Summarization Load** and **Access Count** are mirrored from the main **Report Acceleration Summaries** page. See the subtopic "Use the Report Acceleration Summaries page," above, for more information.

**Size on Disk** shows you how much space the summary takes up in terms of storage. You can use this metric along with the **Summarization Load** and **Access Count** to determine which summaries ought to be deleted.

**Note:** If the **Size** value stays at **0.00MB** it means that Splunk software is not currently generating this summary because the reports associated with it either don't have enough events. At least 100k hot bucket events are required. It is also possible that the projected summary size is over 10% of the bucket that the report is associated with. Splunk software periodically checks this report and automatically creates a summary when the report meets the criteria for summary creation.

**Summary range** is the range of time spanned by the summary, always relative to the present moment. You set this up when you define the report that populates the summary. For more information, see the subtopic "Set report acceleration summary time ranges," above.

**Timespans** displays the size of the data chunks that make up the summary. A summary timespan represents the smallest time range for which the summary contains statistically accurate data. So if you are running a report against a summary that has a one hour timespan, the time range you choose for the report should be evenly divisible by that timespan if you want to get good results. So if you are dealing with a **1h** timespan, a report over the past 24 hours would work fine, but a report over the past 90 minutes might be problematic. See the subsection "How the Splunk platform builds summaries," above, for more information.

**Buckets** shows you how many index **buckets** the summary spans, and **Chunks** tells you how many data chunks comprise the summary. Both of these metrics are informational for the most part, though they may aid with troubleshooting issues you may be encountering with your summary.
Verify a summary

At some point you may find that an accelerated report seems to be returning results that don’t fit with the results the report returned when it was first created. This can happen when certain aspects of the report change without your knowledge, such as a change in the definition of a tag, event type, or field extraction rule used by the report.

If you suspect that this has happened with one of your accelerated reports, go to the detail page for the summary with which the report is associated. You can run a verification process that examines a subset of the summary and verifies that all of the examined data is consistent. If it finds that the data is inconsistent, it notifies you that the verification has failed.

For example, say you have a report that uses an event type, netsecurity, which is associated with a specific kind of network security event. You enable acceleration for this report, and Splunk software builds a summary for it. At some later point, the definition of the event type netsecurity is changed, so it finds an entirely different set of events, which means your summary is now being populated by a different set of data than it was before. You notice that the results being returned by the accelerated report seem to be different, so you run the verification process on it from the Report acceleration summaries page in Settings. The summary fails verification, so you begin investigating the root report to find out what happened.

Ideally the verification process should only have to look at a subset of the summary data in order to save time; a full verification of the entire summary will take as long to complete as the building of the summary itself. But in some cases a more thorough verification is required.

Clicking Verify opens a Verify Summary dialog box. Verify Summary provides two verification options:
• A *Fast verification*, which is set to quickly verify a small subset of the summary data at the cost of thoroughness.
• A *Thorough verification*, which is set to thoroughly review the summary data at the cost of speed.

In both cases, the estimated verification time is provided.

After you click **Start** to kick off the verification process you can follow its progress on the detail page for your summary under **Summary Status**. When the verification process completes, this is where you’ll be notified whether it succeeded or failed. Either way you can click the verification status to see details about what happened.

When verification fails, the **Verification Failed** dialog can tell you what went wrong:

During the verification process, hot buckets and buckets that are in the process of building are skipped.

When a summary fails verification you can review the root search string (or strings) to see if it can be fixed to provide correct results. Once the report is working, click **Rebuild** to rebuild the summary so it is entirely consistent. Or, if you’re fine with the report as-is, just rebuild the report. And if you’d rather start over from scratch, delete the summary and start over with an entirely new report.

**Update, rebuild, and delete summaries**

Click **Update** if the **Summary Status** shows that the summary has not been updated in some time and you would like to make it current. **Update** kicks off a standard summary update report to pull in events so that it is not missing data from the last few hours (for example).

**Note:** When a summary’s **Summary Status** is **Suspended**, you cannot use **Update** to bring it current.
Click **Rebuild** to rebuild the index from scratch. You may want to do this in situations where you suspect there has been data loss due to a system crash or similar mishap, or if it failed verification and you've either fixed the underlying report(s) or have decided that the summary is ok with the data it is currently bringing in.

Click **Delete** to remove the summary from the system (and not regenerate summaries in the future). You may want to do this if the summary is used infrequently and is taking up space that could better be used for something else. You can use the Searches and Reports page in Settings to reenable report acceleration for the report or reports associated with the summary.

### Accelerate data models

Data model acceleration is a tool that you can use to speed up data models that represent extremely large datasets. After acceleration, pivots based on accelerated data model datasets complete quicker than they did before, as do reports and dashboard panels that are based on those pivots.

Data model acceleration does this with the help of the High Performance Analytics Store functionality, which builds data summaries behind the scenes in a manner similar to that of **report acceleration**. Like report acceleration summaries, data model acceleration summaries are easy to enable and disable, and are stored on your indexers parallel to the index buckets that contain the events that are being summarized.

This topic covers:

- The differences between data model acceleration, report acceleration, and summary indexing.
- How you enable persistent acceleration for data models.
- How Splunk software builds data model acceleration summaries.
- How you can query accelerated data model acceleration summaries with the **tstats** command.
- Advanced configurations for persistently accelerated data models.

This topic also explains ad hoc data model acceleration. Splunk software applies ad hoc data model acceleration whenever you build a pivot with an unaccelerated dataset. It is even applied to transaction-based datasets and search-based datasets that use transforming commands, which can't be accelerated in a persistent fashion. However, any acceleration benefits you obtain are lost the moment you leave the Pivot Editor or switch datasets during a
session with the Pivot Editor. These disadvantages do not apply to "persistently" accelerated datasets, which will always load with acceleration whenever they're accessed via Pivot. In addition, unlike "persistent" data model acceleration, ad hoc acceleration is not applied to reports or dashboard panels built with Pivot.

**How data model acceleration differs from report acceleration and summary indexing**

This is how data model acceleration differs from report acceleration and summary indexing:

- Report acceleration and summary indexing *speed up individual searches*, on a report by report basis. They do this by building collections of precomputed search result aggregates.
- Data model acceleration speeds up reporting for the entire set of **fields** that you define in a data model and which you and your Pivot users want to report on. In effect it accelerates the dataset represented by that collection of fields rather than a particular search against that dataset.

**What is a high-performance analytics store?**

Data model acceleration summaries are composed of multiple **time-series index files**, which have the `.tsidx` file extension. Each `.tsidx` file contains records of the indexed field::value combos in the selected dataset and all of the index locations of those field::value combos. It's these `.tsidx` files that make up the high-performance analytics store. Collectively, the `.tsidx` files are optimized to accelerate a range of analytical searches involving the set of fields defined in the accelerated data model.

An accelerated data model's high-performance analytics store spans a summary range. This is a range of time that you select when you enable acceleration for the data model. When you run a pivot on an accelerated dataset, the pivot's time range must fall at least partly within this summary range in order to get an acceleration benefit. For example, if you have a data model that accelerates the last month of data but you create a pivot using one of this data model's dataset that runs over the past year, the pivot will initially only get acceleration benefits for the portion of the search that runs over the past month.

The `.tsidx` files that make up a high-performance analytics store for a single data model are always distributed across one or more of your indexers. This is because Splunk software creates `.tsidx` files on the indexer, parallel to the buckets that contain the events referenced in the file and which cover the range
of time that the summary spans.

The high-performance analytics store created through persistent data model acceleration is different from the summaries created through ad hoc data model acceleration. Ad hoc summaries are always created in a dispatch directory at the search head.

See About ad hoc data model acceleration.

**Enable persistent acceleration for a data model**

See Managing Data Models to learn how to enable data model acceleration.

**Data model acceleration caveats**

There are a number of restrictions on the kinds of data model datasets that can be accelerated.

- **Datasets can only be accelerated if they contain at least one root event hierarchy or one root search hierarchy that only includes streaming commands.** Dataset hierarchies based on root search datasets that include nonstreaming commands and root transaction datasets are not accelerated.
  - Pivots that use unaccelerated datasets fall back to _raw data, which means that they initially run slower. However, they can receive some acceleration benefit from ad hoc data model acceleration. See About ad hoc data model acceleration.
- **Data model acceleration is most efficient if the root event datasets and root search datasets being accelerated include in their initial constraint search the index(es) that Splunk software should search over.** A single high-performance analytics store can span across several indexes in multiple indexers. If you know that all of the data that you want to pivot on resides in a particular index or set of indexes, you can speed things up by telling Splunk software where to look. Otherwise the Splunk software wastes time accelerating data that is not of use to you.

**After you enable acceleration for a data model**

After you enable persistent acceleration for your data model, the Splunk software begins building a data model acceleration summary for the data model that spans the summary range that you’ve specified. Splunk software creates the .tsidx files for the summary in indexes that contain events that have the fields specified in the data model. It stores the .tsidx files parallel to their
corresponding index buckets in a manner identical to that of report acceleration summaries.

After the Splunk software builds the data model acceleration summary, it runs scheduled searches on a 5 minute interval to keep it updated. Every 30 minutes, the Splunk software removes old, outdated .tsidx summary files. You can adjust these intervals in datamodels.conf and limits.conf, respectively.

A few facts about data model acceleration summaries:

- Each bucket in each index in a Splunk deployment can have one or more data model acceleration summary .tsidx files, one for each accelerated data model for which it has relevant data. These summaries are created as data is collected.
- Summaries are restricted to a particular search head (or search head pool ID) to account for different extractions that may produce different results for the same search string.
- You can only accelerate data models that you have shared to all users of an app or shared globally to all users of your Splunk deployment. You cannot accelerate data models that are private. This prevents individual users from taking up disk space with private data model acceleration summaries.

If necessary, you can configure the location of data model acceleration summaries via indexes.conf.

**About the summary range**

Data model acceleration summary ranges span an approximate range of time. At times, a data model acceleration summary can have a store of data that slightly exceeds its summary range, but the summary never fails to meet that range, except during the period when it is first being built.

When Splunk software finishes building a data model acceleration summary, its data model summarization process ensures that the summary always covers its summary range. The process periodically removes older summary data that passes out of the summary range.

If you have a pivot that is associated with an accelerated data model dataset, that pivot completes fastest when you run it over a time range that falls within the summary range of the data model. The pivot runs against the data model acceleration summary rather than the source index _raw data. The summary has far less data than the source index, which means that the pivot completes faster.
than it does on its initial run.

If you run the same pivot over a time range that is only partially covered by the summary range, the pivot is slower to complete. Splunk software has to run at least part of the pivot search over the source index _raw data in the index, which means it must parse through a larger set of events. So it is best to set the Summary Range for a data model wide enough that it captures all of the searches you plan to run against it.

**Note:** There are advanced settings related to Summary Range that you can use if you have a large Splunk deployment that involves multi-terrabyte datasets. This can lead to situations where the search required to build the initial data model acceleration summary runs too long and/or is resource intensive. For more information, see the subtopic Advanced configurations for persistently accelerated data models.

**Summary range example**

You create a data model and accelerate it with a Summary Range of 7 days. Splunk software builds a summary for your data model that approximately spans the past 7 days and then maintains it over time, periodically updating it with new data and removing data that is older than seven days.

You run a pivot over a time range that falls within the last week, and it should complete fairly quickly. But if you run the same pivot over the last 3 to 10 days it will not complete as quickly, even though this search also covers 7 days of data. Only the part of the search that runs over the last 3 to 7 days benefits by running against the data model acceleration summary. The portion of the search that runs over the last 8 to 10 days runs over raw data and is not accelerated. In cases like this, Splunk software returns the accelerated results from summaries first, and then fills in the gaps at a slower speed.

Keep this in mind when you set the Summary Range value. If you always plan to run a report over time ranges that exceed the past 7 days, but don't extend further out than 30 days, you should select a Summary Range of 1 month when you set up data model acceleration for that report.

**How the Splunk platform builds data model acceleration summaries**

When you enable acceleration for a data model, Splunk software builds the initial set of .tsidx file summaries for the data model and then runs scheduled searches in the background every 5 minutes to keep those summaries up to
Each update ensures that the entire configured time range is covered without a significant gap in data. This method of summary building also ensures that late-arriving data is summarized without complication.

**Parallel summarization**

Data model acceleration summaries utilize parallel summarization by default. This means that Splunk software runs three concurrent search jobs to build `.tsidx` summary files instead of one. It also runs up to three concurrent searches on a five minute schedule to maintain those summary files. Parallel summarization decreases the amount of time it takes to build and maintain data model acceleration summaries.

There is a cost for this improvement in summarization search performance. The concurrent searches count against the total number of concurrent search jobs that your Splunk deployment can run, which means that they can cause increased indexer resource usage.

If your Splunk implementation does not use search head clustering and you find that the searches that build and maintain your data model acceleration summaries cause your implementation to reach or exceed concurrent search limits, consider lowering the parallel summarization setting.

If you have `.conf` file access, you can reduce the parallel summarization setting for a data model by editing its `datamodels.conf` stanza.

1. Open the `datamodels.conf` file in your Splunk deployment that has the data model that you want to update summarization settings for.
2. Locate the stanza for the data model.
3. Set `acceleration.max_concurrent = 2`. You can set it to 1 if 2 is too high.
   
   If `acceleration.max_concurrent` is not present in the stanza, add it.
4. Save your changes.

We do not recommend increasing `acceleration.max_concurrent` to a value higher than 3. However, if your Splunk deployment has the capacity for a large amount of search concurrency, you can try setting `acceleration.max_concurrent` to a higher value for selected accelerated data models.

**Additional information**

- See Accomodate many simultaneous searches in the *Capacity Planning Manual* for information about the impact of concurrent searches on search performance.
• See Configure the priority of scheduled reports for more information about how the concurrent search limit for your Splunk deployment is determined.
• See Search head clustering architecture in Distributed Search for more information about how a search head cluster handles concurrent search limits.

**Review summary creation metrics**

The speed of summary creation depends on the amount of events involved and the size of the summary range. You can track progress towards summary completion on the Data Models management page. Find the accelerated data model that you want to inspect, expand its row, and review the information that appears under **ACCELERATION**.

**Status** tells you whether the acceleration summary for the data model is complete. If it is in *Building* status it will tell you what percentage of the summary is complete. Data model acceleration summaries are constantly updating with new data. A summary that is "complete" now will return to "building" status later when it updates with new data.

When the Splunk software calculates the acceleration status for a data model, it bases its calculations on the **Schedule Window** that you have set for for the data model. However, if you have set a backfill relative time range for the data model, that time range is used to calculate acceleration status.

You might set up a backfill time range for a data model when the search that populates the data model acceleration summaries takes an especially long time to run. See Advanced configurations for persistently accelerated data models.

**Verify that the Splunk platform is scheduling summary update searches**

You can verify that Splunk software is scheduling searches to update your data model acceleration summaries. In `log.cfg`, set `category.SavedSplunker=DEBUG` and then watch `scheduler.log` for events like:

```
04-24-2013 11:12:02.357 -0700 DEBUG SavedSplunker - Added 1 scheduled
```
searches for accelerated datamodels to the end of ready-to-run list

When the data model definition changes and your summaries have not been updated to match it

When you change the definition of an accelerated data model, it takes time for Splunk software to update its summaries so that they reflect this change. In the meantime, when you run Pivot searches (or tstats searches) that use the data model, it does not use the summaries that are older than the new definition, by default. This ensures that the output you get from Pivot for the data model always reflects your current configuration.

If you know that the old data is "good enough" you can take advantage of an advanced performance feature that lets the data model return summary data that has not yet been updated to match the current definition of the data model, using a setting called allow_old_summaries, which is set to false by default.

- **On a search by search basis:** When running tstats searches that select from an accelerated data model, set the argument allow_old_summaries=t.
- **For your entire Splunk deployment:** Go to limits.conf and change the allow_old_summaries parameter to true.

**Data model acceleration summary size on disk**

You can use the data model metrics on the Data Models management page to track the total size of a data model's summary on disk. Summaries do take up space, and sometimes a significant amount of it, so it's important that you avoid overuse of data model acceleration. For example, you may want to reserve data model acceleration for data models whose pivots are heavily used in dashboard panels.

The amount of space that a data model takes up is related to the number of events that you are collecting for the summary range you have chosen. It can also be negatively affected if the data model includes fields with high cardinality (that have a large set of unique values), such as a Name field.

If you are particularly size constrained you may want to test the amount of space a data model acceleration summary will take up by enabling acceleration for a small Summary Range first, and then moving to a larger range if you think you can afford it.
Where the Splunk platform creates and stores data model acceleration summaries

By default, Splunk software creates each data model acceleration summary on the indexer, parallel to the bucket or buckets that cover the range of time over which the summary spans, whether the buckets that fall within that range are hot, warm, or cold. If a bucket within the summary range moves to frozen status, Splunk software removes the summary information that corresponds with the bucket when it deletes or archives the data within the bucket.

By default, data model acceleration summaries reside in a predefined volume titled `_splunk_summaries` at the following path:

```
$SPLUNK_DB/<index_name>/datamodel_summary/<bucket_id>/<search_head_or_pool_id>/DM_<datamodel_app>_<datamodel_name>
```

This volume initially has no maximum size specification, which means that it has infinite retention.

Also by default, the `tstatsHomePath` parameter is specified only once as a global setting in `indexes.conf`. Its path is inherited by all indexes. In `etc/system/default/indexes.conf`:

```
[global]
[....]
tstatsHomePath =
volume:_splunk_summaries/$_index_name/datamodel_summary
[....]
```

You can optionally:

- Override this default file path by providing an alternate volume and file path as a value for the `tstatsHomePath` parameter.
- Set different `tstatsHomePath` values for specific indexes.
- Add size limits to any volume (including `_splunk_summaries`) by setting a `maxVolumeDataSizeMB` parameter in the volume configuration.

See the size-based retention example at Configure size-based retention for data model acceleration summaries.

For more information about index buckets and their aging process, see How the indexer stores indexes in the *Managing Indexers and Clusters of Indexers* manual.
How clusters handle data model acceleration summaries

By default, Indexer clusters do not replicate data model acceleration summaries. This means that only primary bucket copies have associated summaries. Under this default setup, if primacy gets reassigned from the original copy of a bucket to another (for example, because the peer holding the primary copy fails), the data model summary does not move to the peer with new primary copy. Therefore, it becomes unavailable. It does not become available again until the next time Splunk software attempts to update the data model summary, finds that it is missing, and regenerates it.

If your peer nodes are running version 6.4 or higher, you can configure the cluster master node so that your indexer clusters replicate data model acceleration summaries. All searchable bucket copies will then have associated summaries. This is the recommended behavior.


Configure size-based retention for data model acceleration summaries

Do you set size-based retention limits for your indexes so they do not take up too much disk storage space? By default, data model acceleration summaries can take up an unlimited amount of disk space. This can be a problem if you are also locking down the maximum data size of your indexes or index volumes. However, you can optionally configure similar retention limits for your data model acceleration summaries.

Although data model acceleration summaries are unbounded in size by default, they are tied to raw data in your index buckets and age along with it. When summarized events pass out of cold buckets into frozen buckets, those events are removed from the related summaries.

Important: Before you attempt to configure size-based retention for your data model acceleration summaries, you should understand how to use volumes to configure limits on index size across indexes. For more information, see "Configure index size" in the Managing Indexers and Clusters of Indexers manual.

Here are the steps you take to set up size-based retention for data model acceleration summaries. All of the configurations described are made within indexes.conf.

393
1. (Optional) If you want to have data model acceleration summary results go into volumes other than `_splunk_summaries`, create them. If you want to use a preexisting volume that controls your indexed raw data, have that volume reference the filesystem that hosts your bucket directories, because your data model acceleration summaries will live alongside it. You can also place your data model acceleration summaries in their own filesystem if you want. You can only reference one filesystem per volume, but you can reference multiple volumes per filesystem.

2. Add `maxVolumeDataSizeMB` parameters to the volume or volumes that will be the home for your data model acceleration summary data, such as `_splunk_summaries`. This lets you manage size-based retention for data model acceleration summary data across your indexes. When a data model acceleration summary volume reaches its maximum size, Splunk software volume manager removes the oldest summary in the volume to make room. It leaves a "done" file behind. The presence of this "done" file prevents Splunk software from rebuilding the summary.

3. Update your index definitions. Set a `tstatsHomePath` for each index that deals with data model acceleration summary data. If you selected an alternate volume than `_splunk_summaries` in Step 1, ensure that the path references that volume. If you defined multiple volumes for your data model acceleration summaries, make sure that the `tstatsHomePath` settings for your indexes point to the appropriate volumes.

You can configure size-based retention for report acceleration summaries in much the same way that you do for data model acceleration summaries. The primary difference is that there is no default volume for report acceleration summaries. For more information about managing size-based retention of report acceleration summaries, see "Manage report acceleration" in this manual.

**Example configuration for data model acceleration size-based retention**

This example configuration sets up data size limits for data model acceleration summaries on the `_splunk_summaries` volume, on a default, per-volume, and per-index basis.

```
```

394
# Default settings

# When you do not provide the tstatsHomePath value for an index, 
# the index inherits the default volume, which gives the index a data 
# size limit of 1TB.

[default]
maxDataSize = 1000000
tstatsHomePath =
volume:_splunk_summaries/$_index_name/datamodel_summary

# Volume definitions

# Indexes with tstatsHomePath values pointing at this partition have 
# a data size limit of 100GB.

[volume:_splunk_summaries]
path = $SPLUNK_DB
maxVolumeDataSizeMB = 100000

# Index definitions

[main]
homePath   = $SPLUNK_DB/defaultdb/db
coldPath   = $SPLUNK_DB/defaultdb/colddb
thawedPath = $SPLUNK_DB/defaultdb/thaweddb
maxMemMB = 20
maxConcurrentOptimizes = 6
maxHotIdleSecs = 86400
maxHotBuckets = 10
maxDataSize = auto_high_volume

[history]
homePath   = $SPLUNK_DB/historydb/db
coldPath   = $SPLUNK_DB/historydb/colddb
thawedPath = $SPLUNK_DB/historydb/thaweddb
tstatsHomePath = volume:_splunk_summaries/historydb/datamodel_summary
maxDataSize = 10
frozenTimePeriodInSecs = 604800

[dm_acceleration]
homePath   = $SPLUNK_DB/dm_accelerationdb/db
coldPath   = $SPLUNK_DB/dm_accelerationdb/colddb
thawedPath = $SPLUNK_DB/dm_accelerationdb/thaweddb

[_internal]
homePath   = $SPLUNK_DB/_internaldb/db
coldPath   = $SPLUNK_DB/_internaldb/colddb
thawedPath = $SPLUNK_DB/_internaldb/thaweddb
Query data model acceleration summaries

You can query the high-performance analytics store for a specific accelerated data model in Search with the `tstats` command.

`tstats` can sort through the full set of `.tsidx` file summaries that belong to your accelerated data model even when they are distributed among multiple indexes.

This can be a way to quickly run a stats-based search against a particular data model just to see if it’s capturing the data you expect for the summary range you’ve selected.

To do this, you identify the data model using `FROM datamodel=<datamodel-name>`:

```
| tstats avg(foo) FROM datamodel=buttercup_games WHERE bar=value2 baz>5
```

The above query returns the average of the field `foo` in the "Buttercup Games" data model acceleration summaries, specifically where `bar` is `value2` and the value of `baz` is greater than 5.

**Note:** You don’t have to specify the app of the data model as Splunk software takes this from the search context (the app you are in). However you cannot query an accelerated data model in App B from App A unless the data model in App B is shared globally.

**Using the summariesonly argument**

The `summariesonly` argument of the `tstats` command enables you to get specific information about data model acceleration summaries.

This example uses the `summariesonly` argument to get the time range of the summary for an accelerated data model named `mydm`.

```
| tstats summariesonly=t min(_time) as min, max(_time) as max from datamodel=mydm | eval prettymin=strftime(min, "%c") | eval prettymax=strftime(max, "%c")
```

This example uses summariesonly in conjunction with `timechart` to reveal what data has been summarized over a selected time range for an accelerated data model titled `mydm`.

```
| tstats summariesonly=t prestats=t count from datamodel=mydm by _time span=1h | timechart span=1h count
```
For more about the `tstats` command, including the usage of `tstats` to query normal indexed data, see the entry for `tstats` in the Search Reference.

**Enable multi-eval to improve data model acceleration**

Searches against root-event datasets within data models iterate through many eval commands, which can be an expensive operation to complete during data model acceleration. You can improve the data model search efficiency by enabling multi-eval calculations for search in `limits.conf`.

```
enable_datamodel_meval = <bool>
* Enable concatenation of successively occurring evals into a single comma-separated eval during the generation of data model searches.
* default: true
```

If you disabled automatic rebuilds for any accelerated data model, you will need to rebuild that data model manually after enabling multi-eval calculations. For more information about rebuilding data models, see Manage data models.

**Advanced configurations for persistently accelerated data models**

There are a few situations that may require you to set up advanced configurations for your persistently accelerated data models in `datamodels.conf`.

* **When summary-populating searches take too long to run**

If your Splunk deployment processes an extremely large amount of data on a regular basis you may find that the initial creation of persistent data model acceleration summaries is resource intensive. The searches that build these summaries may run too long, causing them to fail to summarize incoming events. To deal with this situation, Splunk software gives you two configuration parameters, both in `datamodels.conf`. These parameters are `acceleration.max_time` and `acceleration.backfill_time`.

**Important:** Most Splunk users do not need to adjust these settings. The default `max_time` setting of 1 hour should ensure that long-running summary creation searches do not impede the addition of new events to the summary. We advise that you not change these advanced summary range configurations unless you know it is the only solution to your summary creation issues.
Change the maximum period of time that a summary-populating search can run

The max_time causes summary populating searches to quit after a specified amount of time has passed. After a summary-populating search stops, Splunk software runs a search to catch all of the events that have come in since the initial summary-populating search began, and then it continues adding the summary where the last summary-populating search left off. The max_time parameter is set to 3600 seconds (60 minutes) by default, a setting that should ensure proper summary creation for the majority of Splunk deployments.

For example: You have enabled acceleration for a data model, and you want its summary to retain events for the past three months. Because your organization indexes large amounts of data, the search that initially creates this summary should take about four hours to complete. Unfortunately you can't let the search run interrupted for that amount of time because it might fail to index some of the new events that come in while that four-hour search is in process.

The max_time parameter stops the search after an hour, and another search takes its place to pull in the new events that have come in during that time. It then continues running to add events from the last three months to the summary. This second search also stops after an hour and the process repeats until the summary is complete.

Note: The max_time parameter is an approximate time limit. After the 60 minutes elapses, Splunk software has to finish summarizing the current bucket before kicking off the summary search. This prevents wasted work.

Set a backfill time range that is shorter than the summary time range

If you are indexing a tremendous amount of data with your Splunk deployment and you don't want to adjust the max_time range for a slow-running summary-populating search, you have an alternative option: the acceleration.backfill_time parameter.

The acceleration.backfill_time parameter creates a second "backfill time range" that you set within the summary range. Splunk software builds a partial summary that initially only covers this shorter time range. After that, the summary expands with each new event summarized until it reaches the limit of the larger summary time range. At that point the full summary is complete and events that age out of the summary range are no longer retained.

For example, say you want to set your Summary Range to 1 Month but you know that your system would be taxed by a search that built a summary for that
time range. To deal with this, you set `acceleration.backfill_time = -7d` to run a search that creates a partial summary that initially just covers the past week. After that limit is reached, Splunk software would only add new events to the summary, causing the range of time covered by the summary to expand. But the full summary would still only retain events for one month, so once the partial summary expands to the full **Summary Range** of the past month, it starts dropping its oldest events, just like an ordinary data model acceleration summary does.

*When you do not want persistently accelerated data models to be rebuilt automatically*

By default Splunk software automatically rebuilds persistently accelerated data models whenever it finds that those models are outdated. Data models can become outdated when the current data model search does not match the version of the data model search that was stored when the data model was created.

This can happen if the JSON file for an accelerated model is edited on disk without first disabling the model's acceleration. It can also happen when changes are made to knowledge objects that are interdependent with the data model search. For example, if the data model constraint search references an event type, and the definition of that event type changes, the constraint search will return different results than it did before the change. When the Splunk software detects this change, it will rebuild the data model.

In rare cases you might want to disable this feature for specific accelerated data models, so that those data models are not automatically rebuilt when they become out of date. Instead it will be up to admins to initiate the rebuilds manually. Admins can manually rebuild a data model through the Data Model Manager page, by expanding the row for the affected data model and clicking **Rebuild**.

See Manage data models.

To disable automatic rebuilds for a specific persistently accelerated data model, open `datamodels.conf`, find the stanza for the data model, and set `acceleration.manual_rebuilds = true`

*About ad hoc data model acceleration*

Even when you're building a pivot that is based on a data model dataset that is not accelerated in a persistent fashion, that pivot can benefit from what we call
"ad hoc" data model acceleration. In these cases, Splunk software builds a summary in a search head dispatch directory when you work with a dataset to build a pivot in the Pivot Editor.

The search head begins building the ad-hoc data model acceleration summary after you select a dataset and enter the pivot editor. You can follow the progress of the ad hoc summary construction with the progress bar:

When the progress bar reads Complete, the ad hoc summary is built, and the search head uses it to return pivot results faster going forward. But this summary only lasts while you work with the dataset in the Pivot Editor. If you leave the editor and return, or switch to another dataset and then return to the first one, the search head will need to rebuild the ad hoc summary.

Ad hoc data model acceleration summaries complete faster when they collect data for a shorter range of time. You can change this range for root datasets and their children by resetting the time Filter in the Pivot Editor. See "About ad hoc data model acceleration summary time ranges," below, for more information.

Ad hoc data model acceleration works for all dataset types, including root search datasets that include transforming commands and root transaction datasets. Its main disadvantage against persistent data model acceleration is that with persistent data model acceleration, the summary is always there, keeping pivot performance speedy, until acceleration is disabled for the data model. With ad hoc data model acceleration, you have to wait for the summary to be rebuilt each time you enter the Pivot Editor.

About ad hoc data model acceleration summary time ranges

The search head always tries to make ad hoc data model acceleration summaries fit the range set by the time Filter in the Pivot Editor. When you first enter the Pivot Editor for a dataset, the pivot time range is set to All Time. If your dataset represents a large dataset this can mean that the initial pivot will complete slowly as it builds the ad hoc summary behind the scenes.

When you give the pivot a time range other than All Time, the search head builds an ad hoc summary that fits that range as efficiently as possible. For any given data model dataset, the search head completes an ad hoc summary for a pivot with a short time range quicker than it completes when that same pivot has a
longer time range.

The search head only rebuilds the ad hoc summary from start to finish if you replace the current time range with a new time range that has a different "latest" time. This is because the search head builds each ad hoc summary backwards, from its latest time to its earliest time. If you keep the latest time the same but change the earliest time the search head at most will work to collect any extra data that is required.

Root search datasets and their child datasets are a special case here as they do not have time range filters in Pivot (they do not extract _time as a field). Pivots based on these datasets always build summaries for all of the events returned by the search. However, you can design the root search dataset's search string so it includes "earliest" and "latest" dates, which restricts the dataset represented by the root search dataset and its children.

**How ad hoc data model acceleration differs from persistent data model acceleration**

Here's a summary of the ways in which ad hoc data model acceleration differs from persistent data model acceleration:

- **Ad hoc data model acceleration takes place on the search head rather than the indexer.** This enables it to accelerate all three dataset types (event, search, and transaction).
- **Splunk software creates ad hoc data model acceleration summaries in dispatch directories at the search head.** It creates and stores persistent data model acceleration summaries in your indexes alongside index buckets.
- **Splunk software deletes ad hoc data model acceleration summaries when you leave the Pivot Editor or change the dataset you are working on while you are in the Pivot Editor.** When you return to the Pivot Editor for the same dataset, the search head must rebuild the ad hoc summary. You cannot preserve ad hoc data model acceleration summaries for later use.
  - Pivot job IDs are retained in the pivot URL, so if you quickly use the back button after leaving Pivot (or return to the pivot job with a permalink) you may be able to use the ad-hoc summary for that job without waiting for a rebuild. The search head deletes ad hoc data model acceleration summaries from the dispatch directory a few minutes after you leave Pivot or switch to a different model within Pivot.
• **Ad hoc acceleration does not apply to reports or dashboard panels that are based on pivots.** If you want pivot-based reports and dashboard panels to benefit from data model acceleration, base them on datasets from persistently accelerated event dataset hierarchies.

• **Ad hoc data model acceleration can potentially create more load on your search head than persistent data model acceleration creates on your indexers.** This is because the search head creates a separate ad hoc data model acceleration summary for each user that accesses a specific data model dataset in Pivot that is not persistently accelerated. On the other hand, summaries for persistently accelerated data model datasets are shared by each user of the associated data model. This data model acceleration summary reuse results in less work for your indexers.

**Use summary indexing for increased reporting efficiency**

Use **summary indexing** to efficiently report on large volumes of data. With summary indexing, you set up a frequently-running search that extracts the precise information you want. Each time this search is run, its results are saved into a summary index that you designate. You can then run searches and reports on this significantly smaller (and thus seemingly "faster") summary index. And what's more, these reports will be statistically accurate because of the frequency of the index-populating search (for example, if you want to manually run searches that cover the past seven days, you might run them on a summary index that is updated on an hourly basis).

Summary indexing allows the cost of a computationally expensive report to be spread over time. In the example we've been discussing, the hourly search to populate the summary index with the previous hour's worth of data would take a fraction of a minute. Generating the complete report without the benefit of summary indexing would take approximately 168 (7 days * 24 hrs/day) times longer.

Perhaps an even more important advantage of summary indexing is its ability to amortize costs over different reports, as well as for the same report over a different but overlapping time range. The same summary data generated on a Tuesday can be used for a report of the previous 7 days done on the Wednesday, Thursday, or the following Monday. It could also be used for a monthly report that needed the average response size per day.
Note: Summary indexing volume is not counted against your license, even if you have multiple summary indexes.

Summary indexing use cases

Example #1 - Run reports over long time ranges for large datasets more efficiently: You're using Splunk Enterprise at a company that indexes tens of millions of events--or more--per day. You want to set up a dashboard for your employees that, among other things, displays a report that shows the number of page views and visitors each of your Web sites had over the past 30 days, broken out by site.

You could run this report on your primary data volume, but its runtime would be quite long, because Splunk software has to sort through a huge number of events that are totally unrelated to web traffic in order to extract the desired data. But that's not all--the fact that the report is included in a popular dashboard means it'll be run frequently, and this could significantly extend its average runtime, leading to a lot of frustrated users.

But if you use summary indexing, you can set up a saved search that collects website page view and visitor information into a designated summary index on a weekly, daily, or even hourly basis. You'll then run your month-end report on this smaller summary index, and the report should complete far faster than it would otherwise because it is searching on a smaller and better-focused dataset.

Example #2 - Building rolling reports: Say you want to run a report that shows a running count of an aggregated statistic over a long period of time--a running count of downloads of a file from a Web site you manage, for example.

First, schedule a saved search to return the total number of downloads over a specified slice of time. Then, use summary indexing to save the results of that search into a summary index. You can then run a report any time you want on the data in the summary index to obtain the latest count of the total number of downloads.

For another view, you can watch this Splunk developer video about the theory and practice of summary indexing.

Use the summary indexing reporting commands

If you are new to summary indexing, use the summary indexing reporting commands (`sichart`, `sitimechart`, `sistats`, `sitop`, and `sirare`) when you define the search that will populate the summary index. If you use these commands you
can use the same search string that you use for the search that you eventually run on the summary index, with the exception that you use regular reporting commands in the latter search.

**Note:** You do not have to use the si- summary index search commands if you are proficient with the "old-school" way of creating summary-index-populating searches. If you create summary indexes using those methods and they work for you there's no need to update them. In fact, they may be more efficient: there are performance impacts related to the use of the si- commands, because they create slightly larger indexes than the "manual" method does.

In most cases the impact is insignificant, but you may notice a difference if the summary indexes you are creating are themselves fairly large. You may also notice performance issues if you're setting up several searches to report against an index populated by an si- command search.

See the following section if you're interested in designing summary indexes without the help of si- search commands.

**Define index-populating searches without the special commands**

In previous versions of Splunk Enterprise you had to be very careful about how you designed the searches that you used to populate your summary index, especially if the search you wanted to run on the finished summary index involved aggregate statistics, because it meant that you had to carefully set up the "index-populating" search in a way that did not provide incorrect results. For example, if you wanted to run a search on the finished summary index that gave you average response times broken out by server, you'd want to set up a summary-index-populating search that:

- is scheduled to run on a more frequent basis than the search you plan to run against the summary index
- samples a larger amount of data than the search you plan to run against the summary index.
- contains additional search commands that ensure that the index-populating search is generating a weighted average (only necessary if you are looking for an average in the first place).

The summary index reporting commands take care of the last two points for you--they automatically determine the adjustments that need to be made so that your summary index is populated with data that does not produce statistically inaccurate results. However, you still should arrange for the summary-index-populating search to run on a more frequent basis than the
search that you later run against the summary index.

Interested in setting up summary indexes without the si- commands? Find out about the addinfo, collect, and overlap commands, learn how to devise searches that provide weighted averages, and review an example of summary index configuration via savedsearches.conf in the topic "Configure summary indexes," in this manual.

**Summary indexing reporting command usage example**

Let's say you've been running the following search, with a time range of the past year:

```
eventtype=firewall | top src_ip
```

This search gives you the top source ips for the past year, but it takes forever to run because it scans across your entire index each time.

What you need to do is create a summary index that is composed of the top source IPs from the "firewall" event type. You can use the following search to build that summary index. You would schedule it to run on a daily basis, collecting the top `src_ip` values for only the previous 24 hours each time. The results of each daily search are added to an index named "summary":

```
eventtype=firewall | sitop src_ip
```

**Note:** Summary-index-populating searches are statistically more accurate if you schedule them to run and sample information on a more frequent basis than the searches you plan to run against the finished summary index. So in this example, because we plan to run searches that cover a timespan of a year, we set up a summary-index-populating search that samples information on a daily basis.

**Important:** When you define summary-index-populating searches, do not pipe other search operators after the main summary indexing reporting command. In other words, don't include additional `| eval` commands and the like. Save the extra search operators for the searches you run against the summary indexes, not the search you use to populate it.

**Important:** The results from a summary-indexing optimized search are stored in a special format that cannot be modified before the final transformation is performed. This means that if you populate a summary index with ...

```
... | sistats <args>
```

...the only valid retrieval of the data is: ...

```
index=<summary> source=<saved search name> | stats <args>
```

The search against the summary index cannot
create or modify fields before the \| stats \<args\> command.

Now, let's say you save this search with the name "Summary - firewall top src_ip" (all saved summary-index-populating searches should have names that identify them as such). After your summary index is populated with results, search and report against that summary index using a search that specifies the summary index and the name of the search that you used to populate it. For example, this is the search you would use to get the top source_ips over the past year:

```
index=summary search_name="summary - firewall top src_ip" |top src_ip
```

Because this search specifies the search name, it filters out other data that have been placed in the summary index by other summary indexing searches. This search should run fairly quickly, even if the time range is a year or more.

**Note:** If you are running a search against a summary index that queries for events with a specific sourcetype value, be aware that you need to use `orig_sourcetype` instead. So instead of running a search against a summary index like \(...|stats timechart avg(ip) by sourcetype, use ...|stats timechart avg(ip) by orig_sourcetype.\)

Why do you have to do this? When events are gathered into a summary index, their sourcetype values are changed to "stash" and moves the original sourcetype values to `orig_sourcetype`.

### Set up summary index searches in Splunk Web

In Splunk Web, you can enable summary indexing for scheduled searches and identify the summary indexes that they populate.

**Prerequisites**

Review the following topics.

- Create and edit reports in the *Reporting Manual*
- Schedule reports in the *Reporting Manual*
- Manage summary index gaps
- Create custom indexes in the *Managing Indexers and Clusters of Indexers* manual

**Steps**
1. Create and save a report that you want to use to populate a summary index. The search string for the report should use the "si-" summary reporting commands.

2. Select **Settings > Searches, Reports, and Alerts**.

3. Locate the report that you just created and select **Edit > Edit Schedule** for it.

4. Schedule the report to run on an appropriate interval for its summary index. Make sure that it is scheduled so that it has no data gaps or overlaps. The summary-index-populating report should have an interval that is smaller than the time range of the searches you plan to run against the summary index. This practice helps to ensure that you get statistically accurate results from the searches that you run against the summary index.

   For example, if you plan to run searches against a summary index that return results for the last week, you should populate that summary index with the results of a report that runs on an hourly interval, returning results for the last hour. If you want to run searches against a summary index over the past year of data, arrange for the summary index to collect data on a daily basis for the past day.

5. Click **Next** and then click **Save**.

   You do not need to enable schedule actions for reports that populate summary indexes.

6. Select **Edit > Edit Summary Indexing** for the report you just scheduled.

7. Select **Enable Summary Indexing**.

8. Select a summary index. The default summary index is named **summary**. The list only displays indexes to which you have permission to write.

   It is a best practice to have summary indexes that are dedicated to different types of data. Consider creating a custom index if your search returns data that does not match the data in the available set of indexes.

9. (Optional) Use **Add Fields** to add one or more field/value pairs to the summary index definition.

   The Splunk software annotates events added to the summary index by this search with the field/value pairs that you supply. This enables you to search on these events. For example, you could add the name of the report that populates the summary index (\texttt{report=summary\_firewall\_top\_src\_ip}) to the events in your summary. Later, if you want to restrict a search of the summary index to events added by this search, you can add \texttt{report=summary\_firewall\_top\_src\_ip} to its SPL.
Schedule the populating report to avoid data gaps and overlaps

To minimize data gaps and overlaps you should be sure to set appropriate intervals and delays in the schedules of reports you use to populate summary indexes.

Gaps in a summary index are periods of time when a summary index fails to index events. Gaps can occur if:

- **The summary-index-populating report takes too long to run and runs past the next scheduled run time.** For example, if you were to schedule the report that populates the summary to run every 5 minutes when that report typically takes around 7 minutes to run, you would have problems, because the search won’t run again when it's still running a preceding report.
- **You have forced the summary-index-populating report to use real-time scheduling.** You do this by mistakenly changing the report definition in savedsearches.conf so that the realtime_schedule attribute is set to 1, enabling real-time scheduling. This setting can result in data collection gaps if you are concurrently running several reports. When you define a summary-index-populating scheduled report in Splunk Web by selecting Enable for summary indexing and saving the report, realtime_schedule is set to 0 to ensure that the report never skips a scheduled run. For more information see Configure the priority of scheduled reports, in the Reporting Manual.
- **splunkd goes down.** If Splunk Enterprise can’t index events, you will have gaps in your summary indexes.

Overlaps are events in a summary index (from the same report) that share the same timestamp. Overlapping events skew reports and statistics created from summary indexes. Overlaps can occur if you set the time range of a report to be longer than the frequency of the schedule of the report. In other words, don’t arrange for a report that runs hourly to gather data for the past 90 minutes.

For information about detecting and fixing overlapping data and gaps in data, see "Manage summary index gaps and overlaps" in this manual.

How summary indexing works

In Splunk Web, summary indexing is an alert option for scheduled saved searches. When you run a saved search with summary indexing turned on, its search results are temporarily stored in a file as follows:
MD5 hashes of search names are used to cover situations where the search name is overlong.

From the file, Splunk software uses the addinfo command to add general information about the current search and the fields you specify during configuration to each result. Splunk Enterprise then indexes the resulting event data in the summary index that you've designated for it (index=summary by default).

Note: Use the addinfo command to add fields containing general information about the current search to the search results going into a summary index. General information added about the search helps you run reports on results you place in a summary index.

Summary indexing of data without timestamps

To set the time for summary index events, Splunk software uses the following information, in this order of precedence:

1. The _time value of the event being summarized.
2. The earliest (or minimum) time of the scheduled search that populates the summary index. For example, if the summary-index-populating search covers the two minutes preceding each launch of its search, its earliest time is -2m.
3. The current system time (in the case of an "all time" search, where no "earliest" value is specified).

In the majority of cases, your events will have timestamps, so the first method of discerning the summary index timestamp holds. But if you are summarizing data that doesn't contain an _time field (such as data from a lookup), the resulting events will have the timestamp of the earliest time of the summary-index-populating search.

For example, if you summarize the lookup "asset_table" every night at midnight, and the asset table does not contain an _time column, tonight's summary will have an _time value equal to the earliest time of the search. If I have set the time range of the search to be between -24h and +0s, each summarized event will have an _time value of now()-86400 (that's the start time of the search minus 86,400 seconds, or 24 hours). This means that every event without an _time field value that is found by this summary-index-populating search will be given the exact same _time value: the search's earliest time.
The best practice for summarizing data without a time stamp is to manually create an `_time` value as part of your search. Following on from the example above:

```
|inputlookup asset_table | eval _time=now()
```

**Fields added to summary-indexed data by the si*- summary indexing commands**

**Caution:** Use of these fields and their encoded data by any search commands other than the `si*` summary indexing commands is unsupported. The format and content of these fields can change at any time without warning.

When you run searches with the `si*` commands in order to populate a summary index, Splunk software adds a set of special fields to the summary index data that all begin with `psrsvd`, such as `psrsvd_ct_bytes` and `psrsvd_v` and so on. When you run a search against the summary index with reporting commands like `chart`, `timechart`, and `stats`, the `psrsvd*` fields are used to calculate results for tables and charts that are statistically correct. `psrsvd` stands for "prestats reserved."

Most `psrsvd` types present information about a specific field in the original (pre-summary indexing) file in the dataset, although some `psrsvd` types are not scoped to a single field. The general pattern is `psrsvd_[type]_[fieldname]`. For example, `psrsvd_ct_bytes` presents count information for the `bytes` field.

Here is a list of the available `psrsvd` types:

- `ct` = count
- `gc` = group count (the count for a stats "grouping," not scoped to a single field.
- `nc` = numerical count (number of numerical values)
- `nn` = minimum numerical value
- `nx` = maximum numerical value
- `rd` = rdigest of values (values a the number of times they appear)
- `sm` = sum
- `sn` = minimum lexicographical value
- `ss` = sum of squares
- `sx` = maximum lexicographical value
- `v` = version (not scoped to a single field)
- `vm` = value map (all distinct values for the field and the number of times they appear)
- `vt` = value type (contains the precision of the associated field)
**Lexicographical order**

Lexicographical order sorts items based on the values used to encode the items in computer memory. In Splunk software, this is almost always UTF-8 encoding, which is a superset of ASCII.

- Numbers are sorted before letters. Numbers are sorted based on the first digit. For example, the numbers 10, 9, 70, 100 are sorted lexicographically as 10, 100, 70, 9.
- Uppercase letters are sorted before lowercase letters.
- Symbols are not standard. Some symbols are sorted before numeric values. Other symbols are sorted before or after letters.

**Manage summary index gaps**

The accuracy of your summary index searches can be compromised if the summary indexes involved have gaps in their collected data.

Gaps in summary index data can come about for a number of reasons:

- **A summary index initially only contains events from the point that you start data collection**: Don't lose sight of the fact that summary indexes won't have data from before the summary index collection start date--unless you arrange to put it in there yourself with the backfill script.
- **Splunk deployment outages**: If your Splunk deployment goes down for a significant amount of time, there's a good chance you'll get gaps in your summary index data, depending on when the searches that populate the index are scheduled to run.
- **Searches that run longer than their scheduled intervals**: If the search you're using to populate the scheduled search runs longer than the interval that you've scheduled it to run on, then you're likely to end up with gaps because Splunk software won't run a scheduled search again when a preceding search is still running. For example, if you were to schedule the index-populating search to run every five minutes, you'll have a gap in the index data collection if the search ever takes more than five minutes to run.

**Note**: For general information about creating and maintaining summary indexes, see "Use summary indexing for increased reporting efficiency" in the Knowledge Manager manual.
Use the backfill script to add other data or fill summary index gaps

If you have Splunk Enterprise, you can use the `fill_summary_index.py` script, which backfills gaps in summary index collection by running the saved searches that populate the summary index as they would have been executed at their regularly scheduled times for a given time range. In other words, even though your new summary index only started collecting data at the start of this week, if necessary you can use `fill_summary_index.py` to fill the summary index with data from the past month.

In addition, when you run `fill_summary_index.py` you can specify an App and schedule backfill actions for a list of summary index searches associated with that App, or simply choose to backfill all saved searches associated with the App.

When you enter the `fill_summary_index.py` commands through the CLI, you must provide the backfill time range by indicating an "earliest time" and "latest time" for the backfill operation. You can indicate the precise times either by using relative time identifiers (such as `-3d@d` for "3 days ago at midnight") or by using UTC epoch numbers. The script automatically computes the times during this range when the summary index search would have been run.

**NOTE:** To ensure that the `fill_summary_index.py` script only executes summary index searches at times that correspond to missing data, you must use `-dedup true` when you invoke it.

The `fill_summary_index.py` script requires that you provide necessary authentication (username and password). If you know the valid Splunk Enterprise key when you invoke the script, you can pass it in via the `-sk` option.

The script is designed to prompt you for any required information that you fail to provide in the command line, including the names of the summary index searches, the authentication information, and the time range.

**Examples of fill_summary_index.py invocation**

If this is your situation:

You need to backfill all of the summary index searches for the splunkdotcom App for the past month--but you also need to skip any searches that already have data in the summary index:

Then you'd enter this into the CLI:
If this is your situation:

You need to backfill the `my_daily_search` summary index search for the past year, running no more than 8 concurrent searches at any given time (to reduce impact on performance while the system collects the backfill data). You do not want the script to skip searches that already have data in the summary index. The `my_daily_search` summary index search is owned by the "admin" role.

Then you'd enter this into the CLI:

```
./splunk cmd python fill_summary_index.py -app search -name
my_daily_search -et -y -lt now -j 8 -owner admin -auth admin:changeme
```

**Note:** You need to specify the `-owner` option for searches that are owned by a specific user or role.

**What to do if fill_summary_index.py is interrupted while running**

If `fill_summary_index.py` is interrupted, look for a log directory in the app that you are invoking the process from, such as Search. In that directory you should find an empty temp file named `fsidx*lock`.

Delete this temp file and you should be able to restart `fill_summary_index.py`.

**fill_summary_index.py usage and commands**

In the CLI, start by entering:

```
python fill_summary_index.py
```

...and add the required and optional fields from the table below.

**Note:** `<boolean>` options accept the values 1, t, true, or yes for "true" and 0, f, false, or no for "false."

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-et &lt;string&gt;</td>
<td>Earliest time (required). Either a UTC time or a relative time string.</td>
</tr>
<tr>
<td>-lt &lt;string&gt;</td>
<td>Latest time (required). Either a UTC time or a relative time string.</td>
</tr>
</tbody>
</table>

413
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-app &lt;string&gt;</td>
<td>The app context to use (defaults to None).</td>
</tr>
<tr>
<td>-name &lt;string&gt;</td>
<td>Specify a single saved search name. Can specify multiple times to provide multiple names. Use the wildcard symbol (&quot;**&quot;) to specify all enabled, scheduled saved searches that have a summary index action.</td>
</tr>
<tr>
<td>-names &lt;string&gt;</td>
<td>Specify a comma separated list of saved search names.</td>
</tr>
<tr>
<td>-namefile &lt;filename&gt;</td>
<td>Specify a file with a list of saved search names, one per line. Lines beginning with a # are considered comments and ignored.</td>
</tr>
<tr>
<td>-owner &lt;string&gt;</td>
<td>The user context to use (defaults to &quot;None&quot;).</td>
</tr>
<tr>
<td>-index &lt;string&gt;</td>
<td>Identifies the summary index that the saved search populates. If the index is not provided, the backfill script tries to determine it automatically. If this attempt at auto index detection fails, the index defaults to &quot;summary&quot;.</td>
</tr>
<tr>
<td>-auth &lt;string&gt;</td>
<td>The authentication string expects either &lt;username&gt; or &lt;username&gt;:&lt;password&gt;. If only a username is provided, the script requests the password interactively.</td>
</tr>
<tr>
<td>-sleep &lt;float&gt;</td>
<td>Number of seconds to sleep between each search. Default is 5 seconds.</td>
</tr>
<tr>
<td>-j &lt;int&gt;</td>
<td>Maximum number of concurrent searches to run (default is 1).</td>
</tr>
<tr>
<td>-dedup &lt;boolean&gt;</td>
<td>When this option is set to true, the script does not run saved searches for a scheduled timespan if data already exists in the summary index for that timespan. This option is set to false by default.</td>
</tr>
<tr>
<td>Note: This option has no connection to the dedup command in the search language. The script does not have the ability to perform event-level data analysis. It cannot determine whether certain events are duplicates of others.</td>
<td></td>
</tr>
<tr>
<td>-nolocal &lt;boolean&gt;</td>
<td>Specifies that the summary indexes are not on the search head but are on the indexes instead, if you are working with a distributed environment. To be used in conjunction with -dedup.</td>
</tr>
<tr>
<td>-showprogress &lt;boolean&gt;</td>
<td>When this option is set to true, the script periodically shows the done progress for each currently running search that it spawns. If this option is unused, its default is false.</td>
</tr>
</tbody>
</table>
### Configure summary indexes

For a general overview of summary indexing and instructions for setting up summary indexing through Splunk Web, see the topic "Use summary indexing for increased reporting efficiency" in the Knowledge Manager manual.

You can't manually configure a summary index for a saved report in `savedsearches.conf` until it is set up as a scheduled report that runs on a regular interval, triggers each time it is run, and has the **Enable summary indexing** alert option selected.

In addition, you need to enter the name of the summary index that the report will populate. You do this through the detail page for the report in **Settings > Searches and Reports** after selecting **Enable summary indexing**. The **Summary** index is the default summary index (the index that Splunk Enterprise uses if you do not indicate another one).

If you plan to run a variety of summary index reports you may need to create additional summary indexes. For information about creating new indexes, see "Create custom indexes" in the Managing Indexers and Clusters manual. It's a good idea to create indexes that are dedicated to the collection of summary data.

Summary indexing volume is not counted against your license, even if you have several summary indexes. In the event of a license violation, summary indexing
will halt like any other non-internal search behavior.

**Note:** If you enter the name of an index that does not exist, Splunk Enterprise runs the report on the schedule you've defined, but it does not save the report data to a summary index.

For more information about creating and managing reports, see "Create and edit reports" in this manual.

For more information about defining a report that can populate a summary index, see the subtopic on setting up summary index reports in Splunk Web in "Use summary indexing for in increased reporting efficiency," in this manual.

**Note:** When you define the report that you'll use to build your index, most of the time you should use the summary indexing transforming commands in the report's search string. These commands are prefixed with "si-": `sichart`, `sitimechart`, `sistats`, `sitop`, and `sirare`. The reports you create with them should be versions of the report that you'll eventually use to query the completed summary index.

The summary index transforming commands automatically take into account the issues that are covered in "Considerations for summary index report definition" below, such as scheduling shorter time ranges for the populating report, and setting the populating report to take a larger sample. You only have to worry about these issues if the report you are using to build your index does not include summary index transforming commands.

If you do not use the summary index transforming commands, you can use the `addinfo` and `collect` search commands to create a report that Splunk Enterprise saves and schedules, and which populates a pre-created summary index. For more information about that method, see "Manually populate the summary index" in this topic.

**Customize summary indexing for a scheduled report**

When you use Splunk Web to enable summary indexing for a scheduled and summary-index-enabled report, Splunk Enterprise automatically generates a stanza in `$SPLUNK_HOME/etc/system/local/savedsearches.conf`. You can customize summary indexing for the report by editing this stanza.

If you've used Splunk Web to save and schedule a report, but haven't used Splunk Web to enable the summary index for the report, you can easily enable summary indexing for the report through `savedsearches.conf` as long as you...
have a new index for it to populate. For more information about manual index configuration, see the topic "About managing indexes" in the Managing Indexers and Clusters manual.

```
[<name>]  
action.summary_index = 0 | 1  
action.summary_index._name = <index>  
action.summary_index.<field> = <value>
```

- **[<name>]**: Splunk Enterprise names the stanza based on the name of the scheduled report that you enabled for summary indexing.
- **action.summary_index = 0 | 1**: Set to 1 to enable summary indexing. Set to 0 to disable summary indexing.
- **action.summary_index._name = <index>**: This displays the name of the summary index populated by this report. If you've created a specific summary index for this report, enter its name in `<index>`. Defaults to summary, the summary index that is delivered with Splunk Enterprise.
- **action.summary_index.<field> = <value>**: Specify a field/value pair to add to every event that gets summary indexed by this report. You can define multiple field/value pairs for a single summary index report.

This field/value pair acts as a "tag" of sorts that makes it easier for you to identify the events that go into the summary index when you are running reports against the greater population of event data. This key is optional but we recommend that you never set up a summary index without at least one field/value pair.

For example, add the name of the report that is populating the summary index (action.summary_index.report = summary_firewall_top_src_ip), or the name of the index that the report populates (action.summary_index.index = search).

**Search commands useful to summary indexing**

Summary indexing utilizes a set of specialized transforming commands which you need to use if you are manually creating your summary indexes without the help of the Splunk Web interface or the summary indexing transforming commands.

- **addinfo**: Summary indexing uses addinfo to add fields containing general information about the current report to the report results going into a summary index. Add | addinfo to any report to see what results will look like if they are indexed into a summary index.
- **collect**: Summary indexing uses collect to index report results into the summary index. Use | collect to index any report results into another
Manually configure a report to populate a summary index

If you want to configure summary indexing without using the report options dialog in Splunk Web and the summary indexing transforming commands, you must first configure a summary index just like you would any other index via `indexes.conf`. For more information about manual index configuration, see, see the topic "About managing indexes" in the Managing Indexers and Clusters manual.

**Important:** You must restart Splunk Enterprise for changes in `indexes.conf` to take effect.

1. Design a search string that you want to summarize results from in Splunk Web.
   - Be sure to limit the time range of your report. The number of results that the report generates needs to fit within the maximum report result limits you have set for reporting.
   - Make sure to choose a time interval that works for your data, such as 10 minutes, 2 hours, or 1 day. (For more information about using Splunk Web to schedule report intervals, see the topic "Schedule reports" in the Reporting Manual.)

2. Use the `addinfo` search command. Append `| addinfo` to the end of the report's search string.
   - This command adds information about the report to events that the collect command requires in order to place them into a summary index.
   - You can always add `| addinfo` to any search string to preview what its results will look like in a summary index.

3. Add the `collect` search command to the report's search string. Append `|collect index=<index_name> addtime=t marker="report_name="<summary_report_name>\""` to the end of the search string.
   - Replace `index_name` with the name of the summary index.
• Replace `summary_report_name` with a key to find the results of this report in the index.
• A `summary_report_name` *must* be set if you wish to use the overlap search command on the generated events.

**Note:** For the general case we recommend that you use the provided `summary_index` alert action. Configuring via `addinfo` and `collect` requires some redundant steps that are not needed when you generate summary index events from scheduled reports. Manual configuration remains necessary when you backfill a summary index for timeranges which have already transpired.

**Considerations for summary index report definition**

If for some reason you’re going to set up a summary-index-populating report that *does not use* the summary indexing transforming commands, you should take a few moments to plan out your approach. With summary indexing, the egg comes before the chicken. Use the report that you actually want to run and review the results of to help define the report you use to populate the summary index.

Many summary-searching reports involve aggregated statistics--for example, a report where you are searching for the top 10 ip addresses associated with firewall offenses over the past day--when the main index accrues millions of events per day.

If you populate the summary index with the results of the same report that you run on the summary index, you’ll likely get results that are statistically inaccurate. You should follow these rules when defining the report that populates your summary index to improve the accuracy of aggregated statistics generated from summary index reports.

**Schedule a shorter time range for the populating report**

The report that populates your summary index should be scheduled on a shorter (and therefore more frequent) interval than that of the report that you eventually run against the index. You should go for the smallest time range possible. For example, if you need to generate a daily “top” report, then the report populating the summary index should take its sample on an hourly basis.

**Set the populating report to take a larger sample**

The report populating the summary index should seek out a significantly larger sample than the report that you want to run on the summary index. **So, for example, if you plan to search the summary index for the daily top 10**
offending ip addresses, you would set up a report to populate the summary index with the *hourly top 100* offending ip addresses.

This approach has two benefits--it ensures a higher amount of statistical accuracy for the top 10 report (due to the larger and more-frequently-taken overall sample) and it gives you a bit of wiggle room if you decide you'd rather report on the top 20 or 30 offending ips.

The summary indexing transforming commands automatically take a sample that is larger than the report that you'll run to query the completed summary index, thus creating summary indexes with event data that is not incorrectly skewed. If you do not use those commands, you can use the head command to select a larger sample for the summary-index-populating report than the report that you run over the summary index. In other words, you would have `| head=100` for the hourly summary-index-populating report, and `| head=10` for the daily report over the completed summary index.

**Set up your report to get a weighted average**

If your summary-index-populating report involves averages, and you are not using the summary indexing transforming commands, you need to set that report up to get a weighted average.

For example, say you want to build hourly, daily, or weekly reports of average response times. To do this, you'd generate the "daily average" by averaging the "hourly averages" together. Unfortunately, the daily average becomes skewed if there aren't the same number of events in each "hourly average". You can get the correct "daily average" by using a weighted average function.

The following expression calculates the daily average response time correctly with a weighted average by using the `stats` and `eval` commands in conjunction with the `sum` statistical aggregator. In this example, the `eval` command creates a `daily_average` field, which is the result of dividing the average response time sum by the average response time count.

```bash
| stats sum(hourly_resp_time_sum) as resp_time_sum, 
  sum(hourly_resp_time_count) as resp_time_count | eval daily_average= 
  resp_time_sum/resp_time_count | ......
```

**Schedule the summary-index-populating report to avoid data gaps and overlaps**

Along with the above two rules, to minimize data gaps and overlaps you should
also be sure to set appropriate intervals and delays in the schedules of reports you use to populate summary indexes.

**Gaps** in a summary index are periods of time when a summary index fails to index events. Gaps can occur if:

- `splunkd` goes down.
- the scheduled saved report (the one being summary indexed) takes too long to run and runs past the next scheduled run time. For example, if you were to schedule the report that populates the summary to run every 5 minutes when that report typically takes around 7 minutes to run, you would have problems, because the report won't run again when it's still running a preceding report.

**Overlaps** are events in a summary index (from the same report) that share the same timestamp. Overlapping events skew reports and statistics created from summary indexes. Overlaps can occur if you set the time range of a saved report to be longer than the frequency of the schedule of the report, or if you manually run summary indexing using the collect command.

**Example of a summary index configuration**

This example shows a configuration for a summary index of Apache server statistics as it might appear in `savedsearches.conf`. The keys listed below enable summary indexing for the "Apache Method Summary" report.

**Note:** If you set `action_summary.index=1`, you don't need to have the `addinfo` or `collect` commands in the report's search string.

```
# name of the report = Apache Method Summary
[Apache Method Summary]
# sets the report to run at each interval
counttype = always
# enable the report schedule
enableSched = 1
# report interval in cron notation (this means "every 5 minutes")
schedule = */5****
# id of user for report
userid = jsmith
# search string for summary index
search = index=apache_raw startminutesago=30 endminutesago=25 | extract auto=false | stats count by method
# enable summary indexing
action.summary_index = 1
# name of summary index to which report results are added
```
action.summary_index._name = summary
# add these keys to each event
action.summary_index.report = "count by method"

Other configuration files affected by summary indexing

In addition to the settings you configure in savedsearches.conf, there are also settings for summary indexing in indexes.conf and alert_actions.conf.

Indexes.conf specifies index configuration for the summary index. Alert_actions.conf controls the alert actions (including summary indexing) associated with reports.

Caution: Do not edit settings in alert_actions.conf without explicit instructions from Splunk Technical Support.

Configure batch mode search

A search running in batch mode searches one bucket at a time in batches instead of searching through events over time. Transforming searches that qualify for batch mode processing can complete faster than they would otherwise.

Batch mode search also improves the reliability for long-running distributed searches, which can fail when an indexer goes down while the search is running. In this case, Splunk software attempts to complete the search by reconnecting to the missing peer or redistributing the search across the rest of the peers.

Batch mode search functionality is enabled by default. See "Configure batch mode search in limits.conf" in this topic for information about configuring or disabling batch mode search.

You can make your batch mode searches even faster by enabling batch mode search parallelization. Under batch mode search parallelization, two or more search pipelines are launched for a qualifying search, and they process the search results concurrently. See "Configure batch mode search parallelization" in this topic.

Requirements for batch mode search

Transforming searches that meet the following conditions can run in batch mode.
• The searches need to use generating commands like search, loadjob, datamodel, pivot, or dbinspect.
• The search can include transforming commands, like stats, chart, and so on. However the search cannot include commands like localize and transaction.
• If the search is not distributed, it cannot use commands that require time-ordered events, like streamstats, head, and tail.

Confirm whether or not a search is running in batch mode by using the Search Job Inspector. Batch mode search is indicated by the boolean parameter isBatchModeSearch. See View search job properties in the Search Manual.

Configure batch mode search in limits.conf

If you have a Splunk Enterprise deployment (as opposed to Splunk Cloud), you can configure batch mode search throughout the implementation by changing settings in the limits.conf configuration file, under the [search] stanza.

When you have several batch mode search threads running concurrently, they can become a memory usage burden. You can deal with this by disabling batch mode search for your entire implementation, or by limiting the number of events that a batch mode search thread can read at once from an index bucket.

[search]
allow_batch_mode = <bool>
batch_search_max_index_values = <int>

• allow_batch_mode defaults to true, meaning that batch mode search is enabled for qualifying transforming searches. Disable batch mode search by setting allow_batch_mode = false.
• When allow_batch_mode = true, use the batch_search_max_index_values to limit the number of events read from the index file (bucket). These entries are small, approximately 72 bytes; however, batch mode is more efficient when it can read more entries at once. Defaults to 10000000 (or 10M).

For example, if your batch mode searches are causing you to run low in system memory, you can lower batch_search_max_index_values to 1000000 (1M) to decrease their memory usage. Setting this parameter to a smaller number can lead to slower search performance. You want to find a balance between efficient batch mode searching and system memory conservation.
Set search peer retry period

Other limits.conf settings control the periodicity of retries to search peers in the event of failures, such as connection errors. The interval exists between failure and first retry, as well as successive retries in the event of further failures.

```
[search]
batch_retry_min_interval = <int>
batch_retry_max_interval = <int>
batch_retry_scaling = <double>
batch_wait_after_end = <int>
```

- Use the `batch_retry_min_interval` and `batch_retry_max_interval` parameters to specify the minimum or maximum interval (in seconds) to wait before batch mode attempts to retry the search on a failed peer. The minimum interval defaults to 5 seconds. The maximum interval defaults to 300 seconds.
- After a retry attempt fails increase the time to wait before another retry by a scaling factor, `batch_retry_scaling`, which takes a value greater than 1.0. Defaults to 1.5.
- Batch mode considers the search complete when all peers have indicated without failure that they have delivered the full answer. If the search finishes, but one or more of the peers has failed, batch mode retries connection with the failed peer(s) for the number of seconds specified by `batch_wait_after_end`. If batch mode cannot reconnect within this period of time, it declares the search results to be incomplete. Defaults to 900 seconds.

Search peer restart for batch mode search

Batch mode handles a search peer restart differently depending on whether the peer is clustered or not.

- If the search peer is clustered, batch mode waits for the cluster master to spawn a new generation.
- If the search peer is not clustered and connection to it is lost, batch mode attempts to reconnect to it, following the retry period parameters described above. When batch mode reestablishes connection to the search peer, it resumes the batch mode search until the search completes.
Configure batch mode search parallelization

You can optionally take advantage of batch mode search parallelization to make your batch mode searches even more efficient. When you enable batch mode search parallelization, two or more search pipelines for batch search run concurrently to read from index buckets and process events. This approach improves the speed and efficiency of your batch mode searches, but at the expense of increased system memory consumption.

You can enable and configure batch mode search parallelization with an additional set of limits.conf parameters. This is an indexer-side setting. It needs to be configured on all of your indexers, not your search head(s).

[search]
batch_search_max_pipeline = <int>
batch_search_max_results_aggregator_queue_size = <int>
batch_search_max_serialized_results_queue_size = <int>

- Use `batch_search_max_pipeline` to set the number of batch mode search pipelines launched when you run a search that qualifies for batch mode. This parameter has a default value of 1. Set it to 2 or higher to parallelize batch mode searches throughout your Splunk deployment. A higher setting improves search performance at the cost of increasing thread usage and memory consumption.
- The `batch_search_max_results_aggregator_queue_size` parameter controls the size of the results queue. The results queue is where the search pipelines leave processed search results. Its default size is 100MB. Never set it to zero.
- The `batch_search_max_serialized_results_queue_size` parameter controls the size of the serialized results queue, from which the batch search process transmits serialized search results. Its default size is 100MB. Never set it to zero.