Docs (/guide)

You are looking at documentation for an older release. Not what you want? See the current release documentation (../current/index.html).

edit (https://github.com/elastic/logstash/edit/5.3/docs/static/configuration.asciidoc)

Logstash Reference [5.3] (index.html) » Filter plugins (filter-plugins.html) » date

« csv (plugins-filters-csv.html) de_dot » (plugins-filters-de_dot.html)

date

edit (https://github.com/logstash-plugins/logstash-filter-date/edit/master/lib/logstash/filters/date.rb)

- Version: 3.1.3
- Released on: February 7, 2017
- Changelog (https://github.com/logstash-plugins/logstash-filter-date/blob/master/CHANGELOG.md#313)

Getting Help

edit (https://github.com/logstash-plugins/logstash-filter-date/edit/master/lib/logstash/filters/date.rb)

For questions about the plugin, open a topic in the Discuss (http://discuss.elastic.co) forums. For bugs or feature requests, open an issue in Github (https://github.com/elastic/logstash). For the list of Elastic supported plugins, please consult the Elastic Support Matrix (https://www.elastic.co/support/matrix#show_logstash_plugins).

Description

edit (https://github.com/logstash-plugins/logstash-filter-date/edit/master/lib/logstash/filters/date.rb)

The date filter is used for parsing dates from fields, and then using that date or timestamp as the logstash timestamp for the event.

For example, syslog events usually have timestamps like this:

```
"Apr 17 09:32:01"
```

You would use the date format `MMM dd HH:mm:ss` to parse this.

The date filter is especially important for sorting events and for backfilling old data. If you don't get the date correct in your event, then searching for them later will likely sort out of order.

In the absence of this filter, logstash will choose a timestamp based on the first time it sees the event (at input time), if the timestamp is not already set in the event. For example, with file input, the timestamp is set to the time of each read.

Synopsis

edit (https://github.com/logstash-plugins/logstash-filter-date/edit/master/lib/logstash/filters/date.rb)

This plugin supports the following configuration options:

Required configuration options:

```
date {
}
```

Available configuration options:

Setting	Input type	Required
`add_field` (plugins-filters-date.html#plugins-filters-date-add_field)	hash (configuration-file-structure.html#hash)	No
`add_tag` (plugins-filters-date.html#plugins-filters-date-add_tag)	array (configuration-file-structure.html#array)	No
`enable_metric` (plugins-filters-date.html#plugins-filters-date-enable_metric)	boolean (configuration-file-structure.html#boolean)	No
`id` (plugins-filters-date.html#plugins-filters-date-id)	string (configuration-file-structure.html#string)	No
`locale` (plugins-filters-date.html#plugins-filters-date-locale)	string (configuration-file-structure.html#string)	No
`match` (plugins-filters-date.html#plugins-filters-date-match)	array (configuration-file-structure.html#array)	No
`periodic_flush` (plugins-filters-date.html#plugins-filters-date-periodic_flush)	boolean (configuration-file-structure.html#boolean)	No

Setting	Input type	Required
`remove_field` (plugins-filters-date.html#plugins-filters-date-remove_field)	array (configuration-file-structure.html#array)	No
`remove_tag` (plugins-filters-date.html#plugins-filters-date-remove_tag)	array (configuration-file-structure.html#array)	No
`tag_on_failure` (plugins-filters-date.html#plugins-filters-date-tag_on_failure)	array (configuration-file-structure.html#array)	No
`target` (plugins-filters-date.html#plugins-filters-date-target)	string (configuration-file-structure.html#string)	No
`timezone` (plugins-filters-date.html#plugins-filters-date-timezone)	string (configuration-file-structure.html#string)	No

Details

edit (https://github.com/logstash-plugins/logstash-filter-date/edit/master/lib/logstash/filters/date.rb)

`add_field`

edit (https://github.com/logstash-plugins/logstash-filter-date/edit/master/lib/logstash/filters/date.rb)

- Value type is hash (configuration-file-structure.html#hash)
- Default value is `{}`

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
  date {
    add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
  }
}
```

```
# You can also add multiple fields at once:
filter {
  date {
    add_field => {
      "foo_%{somefield}" => "Hello world, from %{host}"
      "new_field" => "new_static_value"
    }
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

add_tag

edit (https://github.com/logstash-plugins/logstash-filter-date/edit/master/lib/logstash/filters/date.rb)

- Value type is array (configuration-file-structure.html#array)
- Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  date {
    add_tag => [ "foo_%{somefield}" ]
  }
}
```

```
# You can also add multiple tags at once:
filter {
  date {
    add_tag => [ "foo_%{somefield}", "taggedy_tag"]
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

enable_metric

edit (https://github.com/logstash-plugins/logstash-filter-date/edit/master/lib/logstash/filters/date.rb)

- Value type is boolean (configuration-file-structure.html#boolean)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

edit (https://github.com/logstash-plugins/logstash-filter-date/edit/master/lib/logstash/filters/date.rb)

- Value type is string (configuration-file-structure.html#string)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
   id => "my_plugin_id"
 }
}
```

`locale`

edit (https://github.com/logstash-plugins/logstash-filter-date/edit/master/lib/logstash/filters/date.rb)

- Value type is string (configuration-file-structure.html#string)
- There is no default value for this setting.

Specify a locale to be used for date parsing using either IETF-BCP47 or POSIX language tag. Simple examples are `en`,`en-US` for BCP47 or `en_US` for POSIX.

The locale is mostly necessary to be set for parsing month names (pattern with `MMM`) and weekday names (pattern with `EEE`).

If not specified, the platform default will be used but for non-english platform default an english parser will also be used as a fallback mechanism.

`match`

edit (https://github.com/logstash-plugins/logstash-filter-date/edit/master/lib/logstash/filters/date.rb)

- Value type is array (configuration-file-structure.html#array)
- Default value is `[]`

An array with field name first, and format patterns following, `[ field, formats... ]`

If your time field has multiple possible formats, you can do this:

```
match => [ "logdate", "MMM dd yyyy HH:mm:ss",
          "MMM  d yyyy HH:mm:ss", "ISO8601" ]
```

The above will match a syslog (rfc3164) or `iso8601` timestamp.

There are a few special exceptions. The following format literals exist to help you save time and ensure correctness of date parsing.

- `ISO8601` - should parse any valid ISO8601 timestamp, such as `2011-04-19T03:44:01.103Z`
- `UNIX` - will parse **float or int** value expressing unix time in seconds since epoch like 1326149001.132 as well as 1326149001
- `UNIX_MS` - will parse **int** value expressing unix time in milliseconds since epoch like 1366125117000
- `TAI64N` - will parse tai64n time values

For example, if you have a field `logdate`, with a value that looks like `Aug 13 2010 00:03:44`, you would use this configuration:

```
filter {
  date {
    match => [ "logdate", "MMM dd yyyy HH:mm:ss" ]
  }
}
```

If your field is nested in your structure, you can use the nested syntax `[foo][bar]` to match its value. For more information, please refer to the section called “Field References (event-dependent-configuration.html#logstash-config-field-references)”

More details on the syntax

The syntax used for parsing date and time text uses letters to indicate the kind of time value (month, minute, etc), and a repetition of letters to indicate the form of that value (2-digit month, full month name, etc).

Here’s what you can use to parse dates and times:

y year

yyyy

full year number. Example: `2015` .

yy

two-digit year. Example: `15` for the year 2015.

M month of the year

M

minimal-digit month. Example: `1` for January and `12` for December.

MM

two-digit month. zero-padded if needed. Example: `01` for January and `12` for December

MMM

abbreviated month text. Example: `Jan` for January. Note: The language used depends on your locale. See the `locale` setting for how to change the language.

MMMM

full month text, Example: `January`. Note: The language used depends on your locale.

d day of the month

d

minimal-digit day. Example: `1` for the 1st of the month.

dd

two-digit day, zero-padded if needed. Example: `01` for the 1st of the month.

H hour of the day (24-hour clock)

H

minimal-digit hour. Example: `0` for midnight.

HH

two-digit hour, zero-padded if needed. Example: `00` for midnight.

m minutes of the hour (60 minutes per hour)

m

minimal-digit minutes. Example: `0`.

mm

two-digit minutes, zero-padded if needed. Example: `00`.

s seconds of the minute (60 seconds per minute)

s

minimal-digit seconds. Example: `0`.

ss

two-digit seconds, zero-padded if needed. Example: `00`.

S fraction of a second **Maximum precision is milliseconds (`SSS`). Beyond that, zeroes are appended.**

S

tenths of a second. Example: `0` for a subsecond value `012`

SS

hundredths of a second. Example: `01` for a subsecond value `01`

SSS

thousandths of a second. Example: `012` for a subsecond value `012`

Z time zone offset or identity

Z

Timezone offset structured as HHmm (hour and minutes offset from Zulu/UTC). Example: `-0700`.

ZZ

Timezone offset structured as HH:mm (colon in between hour and minute offsets). Example: `-07:00`.

ZZZ

Timezone identity. Example: `America/Los_Angeles`. Note: Valid IDs are listed on the Joda.org available time zones page (http://joda-time.sourceforge.net/timezones.html).

z time zone names. **Time zone names (*z*) cannot be parsed.**

w week of the year

w

minimal-digit week. Example: `1`.

ww

two-digit week, zero-padded if needed. Example: `01`.

D	day of the year
e	day of the week (number)
E	day of the week (text)

E, EE, EEE
: Abbreviated day of the week. Example: `Mon` , `Tue` , `Wed` , `Thu` , `Fri` , `Sat` , `Sun` . Note: The actual language of this will depend on your locale.

EEEE
: The full text day of the week. Example: `Monday` , `Tuesday` , … Note: The actual language of this will depend on your locale.

For non-formatting syntax, you'll need to put single-quote characters around the value. For example, if you were parsing ISO8601 time, "2015-01-01T01:12:23" that little "T" isn't a valid time format, and you want to say "literally, a T", your format would be this: "yyyy-MM-dd'T'HH:mm:ss"

Other less common date units, such as era (G), century (C), am/pm (a), and # more, can be learned about on the joda-time documentation (http://www.joda.org/joda-time/key_format.html).

`periodic_flush`

edit (https://github.com/logstash-plugins/logstash-filter-date/edit/master/lib/logstash/filters/date.rb)

- Value type is boolean (configuration-file-structure.html#boolean)
- Default value is `false`

Call the filter flush method at regular interval. Optional.

`remove_field`

edit (https://github.com/logstash-plugins/logstash-filter-date/edit/master/lib/logstash/filters/date.rb)

- Value type is array (configuration-file-structure.html#array)
- Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
  date {
    remove_field => [ "foo_%{somefield}" ]
  }
}
```

```
# You can also remove multiple fields at once:
filter {
  date {
    remove_field => [ "foo_%{somefield}", "my_extraneous_field" ]
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

remove_tag

edit (https://github.com/logstash-plugins/logstash-filter-date/edit/master/lib/logstash/filters/date.rb)

- Value type is array (configuration-file-structure.html#array)
- Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  date {
    remove_tag => [ "foo_%{somefield}" ]
  }
}
```

```
# You can also remove multiple tags at once:
filter {
  date {
    remove_tag => [ "foo_%{somefield}", "sad_unwanted_tag"]
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

tag_on_failure

edit (https://github.com/logstash-plugins/logstash-filter-date/edit/master/lib/logstash/filters/date.rb)

- Value type is array (configuration-file-structure.html#array)
- Default value is `["_dateparsefailure"]`

Append values to the `tags` field when there has been no successful match

`target`

edit (https://github.com/logstash-plugins/logstash-filter-date/edit/master/lib/logstash/filters/date.rb)

- Value type is string (configuration-file-structure.html#string)
- Default value is `"@timestamp"`

Store the matching timestamp into the given target field. If not provided, default to updating the `@timestamp` field of the event.

`timezone`

edit (https://github.com/logstash-plugins/logstash-filter-date/edit/master/lib/logstash/filters/date.rb)

- Value type is string (configuration-file-structure.html#string)
- There is no default value for this setting.

Specify a time zone canonical ID to be used for date parsing. The valid IDs are listed on the Joda.org available time zones page (http://joda-time.sourceforge.net/timezones.html). This is useful in case the time zone cannot be extracted from the value, and is not the platform default. If this is not specified the platform default will be used. Canonical ID is good as it takes care of daylight saving time for you For example, `America/Los_Angeles` or `Europe/Paris` are valid IDs. This field can be dynamic and include parts of the event using the `%{field}` syntax

« csv (plugins-filters-csv.html)

de_dot » (plugins-filters-de_dot.html)

Top Videos

- Elasticsearch Demo (https://www.elastic.co/webinars/getting-started-elasticsearch?baymax=default&elektra=docs&storm=top-video)
- Kibana 101 (https://www.elastic.co/webinars/getting-started-kibana?baymax=default&elektra=docs&storm=top-video)
- Logstash Primer (https://www.elastic.co/webinars/getting-started-logstash?baymax=default&elektra=docs&storm=top-video)

On this page

Logstash Reference: 5.3 ▼