Date and time format variables

This topic lists the variables that you can use to define time formats in the evaluation functions, strftime() and strptime(). You can also use these variables to describe timestamps in event data.

Additionally, you can use the `relative_time()` and `now()` time functions as arguments.

For more information about working with dates and time, see Time modifiers for search and About searching with time in the *Search Manual*.

Refer to the list of tz database time zones for all permissible time zone values. For more information about how the Splunk software determines a time zone and the tz database, see Specify time zones for timestamps in *Getting Data In*.

> ⓘ NOTE: Subsecond time variables such as %N and %Q can be used in metrics searches of metrics indexes that are enabled for millisecond timestamp resolution.

For more information about enabling metrics indexes to index metric data points with millisecond timestamp precision:

- For Splunk Cloud Platform, see Manage Splunk Cloud Platform indexes in the *Splunk Cloud Platform Admin Manual*.
- For Splunk Enterprise, see Create custom indexes in *Managing indexers and clusters of indexers*.

Date and time variables

Variable	Description
%c	The date and time in the current locale's format as defined by the server's operating system. For example, `Thu Jul 18 09:30:00 2019` for US English on Linux.
%+	The date and time with time zone in the current locale's format as defined by the server's operating system. For example, `Thu Jul 18 09:30:00 PDT 2019` for US English on Linux.

Time variables

Variable	Description
%Ez	Splunk-specific, timezone in minutes.
%f	Microseconds as a decimal number.
%H	Hour (24-hour clock) as a decimal number. Hours are represented by the values 00 to 23. Leading zeros are accepted but not required.
%I	Uppercase "i". Hour (12-hour clock) with the hours represented by the values 01 to 12. Leading zeros are accepted but not required. Use with %p to specify AM or PM for the 12-hour clock.
%k	Like %H, the hour (24-hour clock) as a decimal number. Leading zeros are replaced by a space, for example 0 to 23.
%M	Minute as a decimal number. Minutes are represented by the values 00 to 59. Leading zeros are accepted but not required.
%N	The number of subsecond digits. The default is %9N. You can specify %3N = milliseconds, %6N = microseconds, %9N = nanoseconds.
%p	AM or PM. Use with %I to specify the 12-hour clock for AM or PM. Do not use with %H.
%Q	The subsecond component of a UTC timestamp. The default is milliseconds, %3Q. Some valid values are: • %3Q = milliseconds, with values of 000-999 • %6Q = microseconds, with values of 000000-999999 • %9Q = nanoseconds, with values of 000000000-999999999
%S	Second as a decimal number, for example 00 to 59.
%s	The UNIX Epoch Time timestamp, or the number of seconds since the Epoch: 1970-01-01 00:00:00 +0000 (UTC). For example the UNIX epoch time `1484993700` is equal to `Tue Jan 21 10:15:00 2020`.
%T	The time in 24-hour notation (%H:%M:%S). For example 23:59:59.
%X	The time in the format for the current locale. For US English the format for 9:30 AM is `9:30:00`.
%Z	The timezone abbreviation. For example `EST` for US Eastern Standard Time.

%z	The timezone offset from UTC, in hour and minute: +hhmm or -hhmm. For example, for 5 hours before UTC the values is `-0500` which is US Eastern Standard Time. Examples: • Use %z to specify hour and minute, for example -0500 • Use %:z to specify hour and minute separated by a colon, for example -05:00 • Use %::z to specify hour minute and second separated with colons, for example -05:00:00 • Use %:::z to specify hour only, for example -05
%%	A literal "%" character.

ⓘ NOTE: To parse timestamps with GMT and an offset in data that you upload using Add Data, such as Fri Apr 29 2022 23:45:22 GMT-0700, you might need to use `%:z` to capture both the timestamp and the offset.

Date variables

Variable	Description
%F	Equivalent to %Y-%m-%d (the ISO 8601 date format).
%x	The date in the format of the current locale. For example, 7/13/2019 for US English.

Specifying days and weeks

Variable	Description
%A	Full weekday name. (Sunday, ..., Saturday)
%a	Abbreviated weekday name. (Sun, ... ,Sat)
%d	Day of the month as a decimal number, includes a leading zero. (01 to 31)
%e	Like %d, the day of the month as a decimal number, but a leading zero is replaced by a space. (1 to 31)
%j	Day of year as a decimal number, includes a leading zero. (001 to 366)
%V (or %U)	Week of the year. The %V variable starts the count at 1, which is the most common start number. The %U variable starts the count at 0.
%w	Weekday as a decimal number. (0 = Sunday, ..., 6 = Saturday)

Specifying months

Variable	Description
%b	Abbreviated month name. (Jan, Feb, etc.)
%B	Full month name. (January, February, etc.)
%m	Month as a decimal number. (01 to 12). Leading zeros are accepted but not required.

Specifying year

Variable	Description
%C	The century as a 2-digit decimal number.
%g	The ISO 8601 date format for year as a 2-digit decimal number, without the century. (00 to 99). For example, 25.
%G	The ISO 8601 date format for year with the century as a 4-digit decimal number that corresponds to the ISO week number (see %V). For example, 2025. %G uses the year that is associated with the ISO week number; if the ISO week number belongs to the previous or next year, %G specifies that year.
%y	Year as a decimal number, without the century. (00 to 99). Leading zeros are accepted but not required.
%Y	Year as a decimal number with the century. For example, 2025.

Examples

The date format strings in the following examples include the `T` character as a delimiter, as defined by the ISO 8601 standard.

Converting UNIX timestamps into dates

The following table shows the results of several date format variables, using the `strftime` function. These examples show the results when you use the `strftime` function with the date Tue Apr 29 2025 23:45:22 GMT-0700 (Pacific Daylight Time).

Date format string	Result
`%Y-%m-%d`	2025-04-29
`%y-%m-%d`	25-04-29
`%b %d, %Y`	Apr 29, 2025
`%B %d, %Y`	April 29, 2025
`%a %b %d, %Y`	Tue Apr 29, 2025
`%d %b '%y = %Y-%m-%d`	29 Apr '25 = 2025-04-29

Converting UNIX timestamps into dates and times

The following table shows the results of several date time format variables, using the `strftime` function. These examples show the results when you use the `strftime` function with the date Tue Apr 29 2025 23:45:22 GMT-0700 (Pacific Daylight Time).

Date and Time format string	Result	Description
`%Y-%m-%dT%H:%M:%S.%Q`	2025-04-29T23:45:22.000	Displays the date, followed by the letter T to separate the date from the time. The time includes milliseconds.
`%Y-%m-%dT %H:%M:%S.%Z`	2025-04-29T 23:45:22.PDT	Displays the date followed by the letter T and a space to separate the date from the time. The time includes the letters that represent timezone abbreviation.
`%Y-%m-%dT %H:%M:%S %Z%:z`	2025-04-29T 23:45:22 PDT -07:00	Displays the date followed by the letter T and a space to separate the date from the time. The time includes the letters that represent timezone abbreviation and the hours and minutes offset from UTC.
`%Y-%m-%dT %H:%M:%S.%QZ`	2025-04-29T 23:45:22.000Z	Displays the date, followed by the letter T and a space to separate the date from the time. The time includes milliseconds followed by the letter Z to denote Zulu.
`%Y-%m-%dT%H:%M:%S.%QZ`	2025-04-29T23:45:22.000Z	Displays the date, followed by the letter T to separate the date from the time. The time includes milliseconds followed by the letter Z to denote Zulu.
`%Y-%m-%dT%H:%M:%S`	2025-04-29T23:45:22	Displays the date, followed by the letter T to separate the date from the time.
`%Y-%m-%dT%T`	2025-04-29T23:45:22	Displays the date, followed by the letter T to separate the date from the time. The time is represented in 24-hour notation (%H:%M:%S).
`%m-%d-%Y %I:%M:%S %p`	04-29-2025 11:45:22 PM	Displays the date, followed by a space separate the date from the time. The time is shown in a 12 hour format followed by PM to indicate this time is in the evening.
`%b %d, %Y %I:%M:%S %p`	Apr 29, 2025 11:45:22 PM	Displays the date, using the abbreviated name for the month. A space separates the date from the time. The time is shown in a 12 hour format followed by PM to indicate this time is in the evening.
`%m-%d-%Y %H:`		

`%M:%S.%Q`	04-29-2025 23:45:22.000	Displays the date, followed by a space to separate the date from the time. The time includes milliseconds.
`%m-%d-%Y %H:%M:%S.%Q %z`	04-29-2025 23:45:22.000 -0700	Displays the date, followed by a space to separate the date from the time. The time includes milliseconds and the hours and minutes offset from UTC.
`%d/%b/%Y:%H:%M:%S.%f %z`	29/Apr/2025:23:45:22.000000 -0700	Displays the date, using the abbreviation for the month, and is immediately followed by the time. The time includes nanoseconds and the hours and minutes offset from UTC.

Converting timestamps into UNIX

The following table shows the results of using several date time format variables to convert timestamps into UNIX time using the `strptime` function.

For example, this search returns the UNIX time `1671126322.000000`.

```
... |  eval mytime=strptime("2022-12-15T09:45:22","%Y-%m-%dT%H:%M:%S")
```

Timestamps	Date and Time format string	UNIX time
2022-9-25T09:45:22.000	`%Y-%m-%dT%H:%M:%S.%Q`	1664124322.000000
2022-12-15 09:45:22	`%Y-%m-%d %H:%M:%S`	1671126322.000000

The following table shows the results of searches that use time variables:

Sample search	Result
`host="www1" | eval WeekNo = strftime(_time, "%V")`	Creates a field called `WeekNo` and returns the values for the week numbers that correspond to the dates in the `_time` field.
`... |  eval mytime=strftime(_time,"%Y-%m-%dT%H:%M:%S.%Q")`	Creates a field called `mytime` and returns the converted timestamp values in the `_time` field. The values are stored in UNIX format and converted using the format specified, which is the ISO 8601 format. For example: 2021-04-13T14:00:15.000.
`... | eval start=strptime(Sent, "%H:%M:%S.%N"), end=strptime(Received, "%H:%M:%S.%N") | eval difference=end-start | table end, start, difference`	Takes the values in the Sent and Received fields and converts them into a standard time using the `strptime` function. Then calculates the difference between the start and end times. The results are displayed in a table. You can use the `round` function to round the difference to a specific number of decimal places. For example `...| eval difference=round(end-start, 2)` .

Converting timestamp results to the year of the week format

The following search generates timestamp results using the `gentimes` command and converts the dates to year of the week in ISO date format using the `strftime` function and %G.

```
|sort 0 -  _time
```

The results look like this:

_time	year_week_Y_format	final_week_YV_format	year_week_G_format	final_week_G_format	ti
2025-02-03	2025	2025-06	2025	2025-06	1
2025-02-02	2025	2025-05	2025	2025-05	1
2025-02-01	2025	2025-05	2025	2025-05	1
2025-01-31	2025	2025-05	2025	2025-05	1
2025-01-30	2025	2025-05	2025	2025-05	1