Datetime Data Type

Trifacta® Wrangler supports a variety of date-time formats, each of which has additional variations to it.

Supported Date Ranges:

- **Earliest:** January 1, 1400

  > **NOTE:** Two-digit values for the year that are older than 80 years from the current year are forward-ported into the future. For example, in a job run on Dec 31, 2021, the date `01/01/41` is interpreted as 01/01/1941. However, if the job is run the next day (January 01, 2022), then the same data is interpreted as 01/01/2041. See "Two-digit year values" below.

- **Latest:** December 31, 2599

You can use dates in the Gregorian calendar system only. Dates in the Julian calendar are not supported.

Data Validation

When values are validated against the Datetime data type, the Trifacta application does not compare them to an underlying calendar system. Instead, the application validates the values using regular expressions. This regular expression method checks for general Datetime validation and is fast to evaluate.

However, some values may follow the regular expression validation pattern but are not accurate dates. For example, every four years, February 29 is a valid date. When this date is validated against the Datetime data type, it may be detected as a valid value, while the date is changed in the application to be incremented to a close accurate date, such as March 1 in this example.

Formatting Tokens

You can use the following format strings to change the format of a column of dates:

Letter	Date or Time Component	Presentation	Examples
M	Month in year	Number	1
MM	Month in year	Number	01
MMMM	Month in year	Month	January
MMM	Month in year	Month	Jan
yy	Year	Number	16 **NOTE:** Two-digit values for the year that are older than 80 years from the current year are forward-ported into the future. For example, in a job run on Dec 31, 2021, the date `01/01/41` is interpreted as 01/01/1941. However, if the job is run the next day (January 01, 2022), then the same data is interpreted as 01/01/2041. See "Two-digit year values" below.
yyyy	Year	Number	2016
D	Day in year	Number	352
d	Day in month	Number	9
dd	Day in a month	Number	09

EEE	Day in week (three-letter abbreviation)	Text	Wed
EEEE	Day in week	Text	Wednesday
h	Hour in day (1-12) **NOTE:** Requires an AM/PM indicator (a).	Number	2
hh	Hour in am/pm (01-12) **NOTE:** Requires an AM/PM indicator (a).	Number	02
H	Hour in day (1-12)	Number	2
HH	Hour in day (0-23)	Number	20
m	Minute in an hour	Number	9
mm	Minute in an hour	Number	09
s	Second in a minute	Number	3
ss	Second in a minute	Number	03
SSS	Millisecond	Number	218
X	Time zone	ISO 8601 time zone	`-08:00`
a	AM/PM indicator	String	AM

> **Tip:** If your DateTime column contains data in multiple formats, you must change the format of the DateTime column to one format and then add a transform to convert that data to the other format. When all formats of your source date values are converted to a single format, the application should infer the appropriate date and time format.

Supported Separators:

- Date separators: blank space, comma, single hyphen, or forward slash
- Time separators: blank space, comma, single hyphen, colon, t or T
- Non-delimited Datetime values are supported. For example, yyyymmdd, yyyymmddThhmmssX.

ISO 8601 Time Zone Notes:

- Support for timezone offset from UTC indicated by +hh:mm, +hhmm, or +hh. For example, the date '2013-11-18 11:55-04:00' is recognized as a DateTime value.
- Datetime part functions (for example, Hour) truncate time zones and return local time.
- If you have a column with multiple time zones, you can convert the column to Unixtime so you can perform Date/Time operations with a standardized time zone. If you want to work with local times, you can truncate the time zone or use other Datetime functions. See *UNIXTIME Function*.

For more information on supported date formatting strings, see *DATEFORMAT Function*.

Two-digit year values

Depending on the system, a two-digit value for year in a Datetime value is subject to different interpretations. In Trifacta Wrangler, two-digit values for the year that are older than 80 years from the current year are forward-ported into the future. For example, in a job run on Dec 31, 2021, the date `01/01/41` is interpreted as 01/01/1941. However, if the job is run the next day (January 01, 2022), then the same data is interpreted as 01/01/2041.

Other systems use different limits for backward versus forward porting of year values:

- In BigQuery, if no century value is provided, then the year value has a century value applied to it based on a fixed range. See *https://cloud.google.com/bigquery/docs/reference/standard-sql/format-elements#:~:text=The%20year%20without%20century%20as%20a%20decimal%20number%20(00%2D99)* .
- Snowflake permits customization of two-digit year values at the Account, Session, or Object level. See *https://docs.snowflake.com/en/sql-reference/parameters.html#two-digit-century-start*.

As a result, it can be a challenge to manage these system-dependent two-digit years in a consistent manner.

> **Tip:** For best results, you should format year values as four-digit values before the data is ingested into Trifacta Wrangler. Four-digit years are consistently represented across all systems.

If the above is not possible, you can create replacement steps in your recipe to convert two-digit years to four-digit values. In the following example, `00-39` is interpreted as a `19XX` year, while `40-99` is interpreted as a `20XX` year:

Transformation Name	`Replace text or pattern`
Parameter: Column	`myDateColumn`
Parameter: Find	`/\b([456789][0-9])\b$/`
Parameter: Replace with	`19$1`

and

Transformation Name	Replace text or pattern
Parameter: Column	myDateColumn
Parameter: Find	/\b([0123][0-9])\b$/
Parameter: Replace with	20$1

Job Execution

Datetime data typing involves the basic type definition, plus any supported formatting options. Depending on where the job is executed, there may be variation in how the Datetime data type is interpreted.

- Some running environments may perform additional inference on the typing.

  NOTE: During job execution on Spark, inputs of Datetime data type may result in row values being inferred for data type individually. For example, the String value `01/10/2020` may be inferred by date transformations as `1st Oct, 2020` or `10th Jan, 2020`. Resulting outputs of Datetime values may not be deterministic in this scenario.

- Some formatting options may not be supported.

Differences between Trifacta Photon and Spark running environments

If your Datetime data does not contain time zone information, by default:

- Spark uses the time zone of the Trifacta node for Datetime values.
- Trifacta Photon uses the UTC time zone for Datetime values.

This difference in how the values are treated can result in differences in Datetime-based calculations, such as the DATEDIF function.

Workarounds:

You can do one of the following:

- Set the time zone for the Trifacta node to be UTC. You must also set the time zone for your Spark running environment to UTC.
- Apply the following Spark property overrides:

```
"spark":
  "props": {
    ...
    "spark.driver.extraJavaOptions" : "-Duser.timezone=\"UTC\"",
    "spark.executor.extraJavaOptions" : "-Duser.timezone=\"UTC\""
  }
  ...
}
```

For more information, see *Spark Execution Properties Settings*.