Date and time input and output formats

Date and time formats provide a method for representing dates, times, and timestamps.

How Snowflake determines the input and output formats to use

To determine the input and output formats to use for dates, times, and timestamps, Snowflake uses:

- Session parameters for dates, times, and timestamps
- File format options for loading/unloading dates, times, and timestamps

Session parameters for dates, times, and timestamps

A set of session parameters determines how date, time, and timestamp data is passed into and out of Snowflake, as well as the time zone used in the time and timestamp formats that support time zones.

You can set the parameters at the account, user, and session levels. Execute the SHOW PARAMETERS command to view the current parameter settings that apply to all operations in the current session.

The following parameters define which date, time, and timestamp formats are recognized for DML, including COPY, INSERT, and MERGE operations:

- DATE_INPUT_FORMAT
- TIME_INPUT_FORMAT
- TIMESTAMP_INPUT_FORMAT

The default for all three parameters is AUTO. When the parameter value is set to AUTO, Snowflake attempts to match date, time, or timestamp strings in any input expression with one of the formats listed in Supported formats for AUTO detection:

- If a matching format is found, Snowflake accepts the string.
- If no matching format is found, Snowflake returns an error.

Output formats

The following parameters define the formats for date and time output from Snowflake:

- DATE_OUTPUT_FORMAT
- TIME_OUTPUT_FORMAT
- CSV_TIMESTAMP_FORMAT
- TIMESTAMP_OUTPUT_FORMAT
- TIMESTAMP_LTZ_OUTPUT_FORMAT
- TIMESTAMP_NTZ_OUTPUT_FORMAT

In addition, the following parameter maps the TIMESTAMP data type alias to one of the three TIMESTAMP_* variations:

- TIMESTAMP_TYPE_MAPPING

Time zone

The following parameter determines the time zone:

- TIMEZONE

File format options for loading/unloading dates, times, and timestamps

Separate from the input and output format parameters, Snowflake provides three file format options to use when loading data into or unloading data from Snowflake tables:

- DATE_FORMAT
- TIME_FORMAT
- TIMESTAMP_FORMAT

The options can be specified directly in the COPY command or in a named stage or file format object referenced in the COPY command. When specified, these options override the corresponding input formats (when loading data) or output formats (when unloading data).

When used in data loading, the options specify the format of the date, time, and timestamp strings in your staged data files. The options override the DATE_INPUT_FORMAT, TIME_INPUT_FORMAT, or TIMESTAMP_INPUT_FORMAT parameter settings.

The default for all these options is AUTO, meaning the COPY INTO <table> command attempts to match all date and timestamp strings in the staged data files with one of the formats listed in Supported formats for AUTO detection:

- If a matching format is found, Snowflake accepts the string.
- If no matching format is found, Snowflake returns an error and then performs the action specified for the ON_ERROR copy option.

Warning
Snowflake supports automatic detection of most common date, time, and timestamp formats (see tables below). However, some formats might produce ambiguous results, which can cause Snowflake to apply an incorrect format when using AUTO for data loading.

To guarantee correct loading of data, Snowflake ***strongly*** recommends explicitly setting the file format options for data loading.

Data unloading

When used in data unloading, the options specify the format applied to the dates, times, and timestamps unloaded to the files in specified stage.

- TIME_OUTPUT_FORMAT
- TIMESTAMP_*_OUTPUT_FORMAT (depending on the TIMESTAMP_TYPE_MAPPING setting)

About the elements used in input and output formats

In input and output formats that you specify in parameters, file format options, and conversion functions, you can use the elements listed in the table below.

The next sections also use these elements to describe the formats recognized by Snowflake automatically.

Format element	Description
`YYYY`	Four-digit year.
`YY`	Two-digit year, controlled by the TWO_DIGIT_CENTURY_START session parameter. For example, when set to `1980`, values of `79` and `80` are parsed as `2079` and `1980` respectively.
`MM`	Two-digit month (`01` = January, and so on).
`MON`	Full or abbreviated month name.
`MMMM`	Full month name.
`DD`	Two-digit day of month (`01` through `31`).
`DY`	Abbreviated day of week.

Format element	Description
`HH12`	Two digits for hour (`01` through `12`). You can specify `AM` / `PM`.
`AM` , `PM`	Ante meridiem (`AM`) / post meridiem (`PM`). Use this only with `HH12` (***not*** with `HH24`).
`MI`	Two digits for minute (`00` through `59`).
`SS`	Two digits for second (`00` through `59`).
`FF[0-9]`	Fractional seconds with precision `0` (seconds) to `9` (nanoseconds), e.g. `FF`, `FF0`, `FF3`, `FF9`. Specifying `FF` is equivalent to `FF9` (nanoseconds).
`TZH:TZM` , `TZHTZM` , `TZH`	Time zone hour and minute, offset from UTC. Can be prefixed by `+`/`-` for sign.
`UUUU`	Four-digit year in ISO format, which are negative for BCE years.

Note

- When a date-only format is used, the associated time is assumed to be midnight on that day.
- Anything in the format between double quotes or other than the above elements is parsed/formatted without being interpreted.
- For more details about valid ranges, number of digits, and best practices, see Additional information about using date, time, and timestamp formats.

If instructed to do so, Snowflake automatically detects and processes specific formats for date, time, and timestamp input strings. The following sections describe the supported formats:

- Date formats
- Time formats
- Timestamp formats

Attention
Some strings can match multiple formats. For example, `'07-04-2016'` is compatible with both `MM-DD-YYYY` and `DD-MM-YYYY`, but has different meanings in each format (July 4 vs. April 7). The fact that a matching format is found does ***not*** guarantee that the string is parsed as the user intended.

Although automatic date format detection is convenient, it increases the possibility of misinterpretation. Snowflake strongly recommends specifying the format explicitly rather than relying on automatic date detection.

Date formats

For descriptions of the elements used in the formats below, see About the elements used in input and output formats.

Format	Example	Notes
ISO Date Formats		

Format	Example	Notes
Other Date Formats		
`DD-MON-YYYY`	`17-DEC-1980`	
`MM/DD/YYYY`	`12/17/1980`	Could produce incorrect dates when loading or operating on dates in common European formats (that is, `DD/MM/YYYY`). For example, 05/02/2013 could be interpreted as May 2, 2013 instead of February 5, 2013.

When using AUTO date formatting, dashes and slashes aren't interchangeable. Slashes imply `MM/DD/YYYY` format, and dashes imply `YYYY-MM-DD` format. Strings such as `'2019/01/02'` or `'01-02-2019'` aren't interpreted as you might expect.

Time formats

For descriptions of the elements used in the formats below, see About the elements used in input and output formats.

Format	Example	Notes
ISO Time Formats		
`HH24:MI:SS.FFTZH:TZM`	`20:57:01.123456789+07:00`	
`HH24:MI:SS.FF`	`20:57:01.123456789`	

Format	Example	Notes
`HH24:MI`	`20:57`	
Internet (RFC) Time Formats		
`HH12:MI:SS.FF AM`	`07:57:01.123456789 AM`	
`HH12:MI:SS AM`	`04:01:07 AM`	
`HH12:MI AM`	`04:01 AM`	

The `AM` format element allows values with either `AM` or `PM`.

Note

Use the `AM` format element only with `HH12` (not with `HH24`).

When a timezone offset (for example, `0800`) occurs immediately after a digit in a time or timestamp string, the timezone offset must start with `+` or `-`. The sign prevents ambiguity when the fractional seconds or the time zone offset does not contain the maximum number of allowable digits. For example, without a separator between the last digit of the fractional seconds and the first digit of the timezone, the `1` in the time `04:04:04.321200` could be either the last digit of the fractional seconds (that is, 321 milliseconds) or the first digit of the timezone offset (that is, 12 hours ahead of UTC).

Format	Example	Notes
ISO Timestamp Formats		
`YYYY-MM-DD"T"HH24:MI:SS.FFTZH:TZM`	`2013-04-28T20:57:01.123456789+07:00`	The double quotes around the `T` are optional (see the tip following this table for details).
`YYYY-MM-DD HH24:MI:SS.FFTZH:TZM`	`2013-04-28 20:57:01.123456789+07:00`	
`YYYY-MM-DD HH24:MI:SS.FFTZH`	`2013-04-28 20:57:01.123456789+07`	
`YYYY-MM-DD HH24:MI:SS.FF TZH:TZM`	`2013-04-28 20:57:01.123456789 +07:00`	
`YYYY-MM-DD HH24:MI:SS.FF TZHTZM`	`2013-04-28 20:57:01.123456789 +0700`	
`YYYY-MM-DD HH24:MI:SS TZH:TZM`	`2013-04-28 20:57:01 +07:00`	
`YYYY-MM-DD HH24:MI:SS TZHTZM`	`2013-04-28 20:57:01 +0700`	
`YYYY-MM-DD"T"HH24:MI:SS.FF`	`2013-04-28T20:57:01.123456`	The double quotes around the `T` are optional (see the tip following this table for details).
`YYYY-MM-DD HH24:MI:SS.FF`	`2013-04-28 20:57:01.123456`	

Format	Example	Notes
`YYYY-MM-DD HH24:MI:SS`	`2013-04-28 20:57:01`	
`YYYY-MM-DD"T"HH24:MI`	`2013-04-28T20:57`	The double quotes around the `T` are optional (see the tip following this table for details).
`YYYY-MM-DD HH24:MI`	`2013-04-28 20:57`	
`YYYY-MM-DD"T"HH24`	`2013-04-28T20`	The double quotes around the `T` are optional (see the tip following this table for details).
`YYYY-MM-DD HH24`	`2013-04-28 20`	
`YYYY-MM-DD"T"HH24:MI:SSTZH:TZM`	`2013-04-28T20:57:01-07:00`	The double quotes around the `T` are optional (see the tip following this table for details).
`YYYY-MM-DD HH24:MI:SSTZH:TZM`	`2013-04-28 20:57:01-07:00`	
`YYYY-MM-DD HH24:MI:SSTZH`	`2013-04-28 20:57:01-07`	
`YYYY-MM-DD"T"HH24:MITZH:TZM`	`2013-04-28T20:57+07:00`	The double quotes around the `T` are optional (see the tip following this table for details).
`YYYY-MM-DD HH24:MITZH:TZM`	`2013-04-28 20:57+07:00`	

Internet (RFC) Timestamp

Format	Example	Notes
`DY, DD MON YYYY HH24:MI:SS TZHTZM`	`Thu, 21 Dec 2000 16:01:07 +0200`	
`DY, DD MON YYYY HH24:MI:SS.FF TZHTZM`	`Thu, 21 Dec 2000 16:01:07.123456789 +0200`	
`DY, DD MON YYYY HH12:MI:SS AM TZHTZM`	`Thu, 21 Dec 2000 04:01:07 PM +0200`	
`DY, DD MON YYYY HH12:MI:SS.FF AM TZHTZM`	`Thu, 21 Dec 2000 04:01:07.123456789 PM +0200`	
`DY, DD MON YYYY HH24:MI:SS`	`Thu, 21 Dec 2000 16:01:07`	
`DY, DD MON YYYY HH24:MI:SS.FF`	`Thu, 21 Dec 2000 16:01:07.123456789`	
`DY, DD MON YYYY HH12:MI:SS AM`	`Thu, 21 Dec 2000 04:01:07 PM`	
`DY, DD MON YYYY HH12:MI:SS.FF AM`	`Thu, 21 Dec 2000 04:01:07.123456789 PM`	
Other Timestamp Formats		

Format	Example	Notes
`MM/DD/YYYY HH24:MI:SS`	`2/18/2008 02:36:48`	Could produce incorrect dates when loading or operating on dates in common European formats (i.e. `DD/MM/YYYY`). For example, 05/02/2013 could be interpreted as May 2, 2013 instead of February 5, 2013.
`DY MON DD HH24:MI:SS TZHTZM YYYY`	`Mon Jul 08 18:09:51 +0000 2013`	

When a timezone offset (for example, `0800`) occurs immediately after a digit in a time or timestamp string, the timezone offset must start with `+` or `-`. The sign prevents ambiguity when the fractional seconds or the time zone offset does not contain the maximum number of allowable digits. For example, without a separator between the last digit of the fractional seconds and the first digit of the timezone, the `1` in the time `04:04:04.321200` could be either the last digit of the fractional seconds (that is, 321 milliseconds) or the first digit of the timezone offset (that is, 12 hours ahead of UTC).

Tip

In some of the timestamp formats, the letter `T` is used as a separator between the date and time (for example, `'YYYY-MM-DD"T"HH24:MI:SS'`).

The double quotes around the `T` are optional. However, Snowflake recommends using double quotes around the `T` (and other literals) to avoid ambiguity.

Use the double quotes only in the format specifier, ***not*** the actual values. For example:

```
SELECT TO_TIMESTAMP('2019-02-28T23:59:59', 'YYYY-MM-DD"T"HH24:MI:SS');
```

In addition, the quotes around the `T` must be double quotes.

Additional information about using date, time, and timestamp formats

The following sections describe requirements and best practices for individual fields in dates, times, and timestamps.

Valid ranges of values for fields

The recommended ranges of values for each field are shown below:

Field	Values	Notes
Years	`0001` to `9999`	Some values outside this range might be accepted in some contexts, but Snowflake recommends using only values in this range. For example, the year 0000 is accepted, but is incorrect because in the Gregorian calendar the year 1

Field	Values	Notes
Months	`01` to `12`	
Days	`01` to `31`	In months that have fewer than 31 days, the actual maximum is the number of days in the month.
Hours	`00` to `23`	Or `01`-`12` if you are using `HH12` format.
Minutes	`00` to `59`	
Seconds	`00` to `59`	Snowflake doesn't support leap seconds or leap-leap seconds; values `60` and `61` are rejected.
Fraction	`0` to `999999999`	The number of digits after the decimal point depends in part upon the exact format specifier (for example, `FF3` supports up to 3 digits after the decimal point and `FF9` supports up to 9 digits after the decimal point). You can enter fewer digits than you specified (for example, 1 digit is allowed even if you use `FF9`); trailing zeros aren't required to fill out the field to the specified width.

Using the correct number of digits with format elements

For most fields (year, month, day, hour, minute, and second), the elements (`YYYY`, `MM`, `DD`, and so on) of the format specifier are two or four characters.

The following rules tell you how many digits you should actually specify in the literal values:

- `YY`: Specify 1 or 2 digits of the year. However, Snowflake recommends specifying 2 digits. If necessary, prepend a leading zero.
- `MM`: Specify one or two digits. For example, January can be represented as `01` or `1`. Snowflake recommends using two digits.
- `DD`: Specify one or two digits. Snowflake recommends using two digits.
- `HH12` and `HH24`: Specify one or two digits. Snowflake recommends using two digits.
- `MI`: Specify one or two digits. Snowflake recommends using two digits.
- `SS`: Specify one or two digits. Snowflake recommends using two digits.
- `FF9`: Specify between 1 and 9 digits (inclusive). Snowflake recommends specifying the number of actual significant digits. Trailing zeros aren't required.
- `TZH`: Specify one or two digits. Snowflake recommends using two digits.
- `TZM`: Specify one or two digits. Snowflake recommends using two digits.

For all fields (other than fractional seconds), Snowflake recommends specifying the maximum number of digits. Use leading zeros if necessary. For example, `0001-02-03 04:05:06 -07:00` follows the recommended format.

For fractional seconds, trailing zeros are optional. In general, it is considered good practice to specify only the number of digits that are reliable and meaningful. For example, if a time measurement is accurate to three decimal places (milliseconds), then specifying it as nine digits (for example, `.123000000`) might be misleading.

Whitespace in values and format specifiers

Snowflake enforces matching whitespace in some, but not all, situations. For example, the following statement generates

```
SELECT TO_TIMESTAMP('2019-02-2823:59:59 -07:00', 'YYYY-MM-DD HH24:MI:SS TZH:TZM');
```

However, the following statement doesn't generate an error, even though the value contains a whitespace where the specifier doesn't:

```
SELECT TO_TIMESTAMP('2019-02-28 23:59:59.000000000 -07:00', 'YYYY-MM-DDHH24:MI:SS.FF TZH:TZM');
```

The reason for the difference is that in the former case, the values would be ambiguous if the fields aren't all at their maximum width. For example, `213` could be interpreted as 2 days and 13 hours, or as 21 days and 3 hours. However, `DDHH` is unambiguously the same as `DD HH` (other than the whitespace).

Tip
Although some whitespace differences are allowed in order to handle variably-formatted data, Snowflake recommends that values and specifiers exactly match, including spaces.

Context dependency

Not all restrictions are enforced equally in all contexts. For example, some expressions might roll over February 31, while others might not.

These best practices minimize ambiguities and other potential issues in past, current, and projected future versions of Snowflake:

- Be aware of the dangers of mixing data from sources that use different formats (for example, of mixing data that follows the common U.S. format `MM-DD-YYYY` and the common European format `DD-MM-YYYY`).
- Specify the maximum number of digits for each field (except fractional seconds). For example, use four-digit years, specifying leading zeros if necessary.
- Specify a blank or the letter `T` between the date and time in a timestamp.
- Make sure whitespace (and the optional `T` separator between the date and time) are the same in values and in the format specifier.
- Use interval arithmetic if you need the equivalent of rollover.
- Be careful when using AUTO formatting. When possible, specify the format, and ensure that values always match the specified format.
- Specify the format in the command, because it is safer than specifying the format outside the command, for example in a parameter such as DATE_INPUT_FORMAT. (See below.)
- When moving scripts from one environment to another, ensure that date-related parameters, such as DATE_INPUT_FORMAT, are the same in the new environment as they were in the old environment (assuming that the values are also in the same format).

Date & time functions

Snowflake provides a set of functions to construct, convert, extract, or modify DATE, TIME, and TIMESTAMP data. For more information, see Date & Time Functions.

AUTO detection of integer-stored date, time, and timestamp values

For integers of seconds or milliseconds stored in a string, Snowflake attempts to determine the correct unit of measurement based on the length of the value.

Note
The use of quoted integers as inputs is deprecated.

This example calculates the timestamp equivalent to 1487654321 seconds since the start of the Unix epoch:

```
SELECT TO_TIMESTAMP('1487654321');
```

```
+-------------------------------+
| TO_TIMESTAMP('1487654321')    |
|-------------------------------|
| 2017-02-21 05:18:41.000000000 |
+-------------------------------+
```

Here is a similar calculation using milliseconds since the start of the epoch:

```
SELECT TO_TIMESTAMP('1487654321321');
```

```
+-------------------------------+
| TO_TIMESTAMP('1487654321321') |
|-------------------------------|
| 2017-02-21 05:18:41.321000000 |
+-------------------------------+
```

Depending on the magnitude of the value, Snowflake uses a different unit of measure:

- After the string is converted to an integer, the integer is treated as a number of seconds, milliseconds, microseconds, or nanoseconds after the start of the Unix epoch (1970-01-01 00:00:00.000000000 UTC).
  - If the integer is less than 31536000000 (the number of milliseconds in a year), then the value is treated as a number of seconds.
  - If the value is greater than or equal to 31536000000 and less than 31536000000000, then the value is treated as milliseconds.
  - If the value is greater than or equal to 31536000000000 and less than 31536000000000000, then the value is treated as microseconds.
  - If the value is greater than or equal to 31536000000000000, then the value is treated as nanoseconds.
- If more than one row is evaluated (for example, if the input is the column name of a table that contains more than one row), each value is examined independently to determine if the value represents seconds, milliseconds, microseconds, or nanoseconds.

In cases where formatted strings and integers in strings are passed to the function, each value is cast according to the contents of the string. For example, if you pass a date-formatted string and a string containing an integer to TO_TIMESTAMP, the function interprets each value correctly according to what each string contains:

```
+-------------------------+
| TO_TIMESTAMP(COLUMN1)   |
|-------------------------|
| 2013-04-05 00:00:00.000 |
| 2017-02-21 05:18:41.000 |
+-------------------------+
```

Date & time function format best practices

AUTO detection usually determines the correct input format. However, there are situations where it might not be able to make the correct determination.

To avoid this, Snowflake strongly recommends the following best practices (substituting TO_DATE , DATE or TO_TIME , TIME for TO_TIMESTAMP, as appropriate):

- Avoid using AUTO format if there is any chance for ambiguous results. Instead, specify an explicit format string by:
  - Setting TIMESTAMP_INPUT_FORMAT and other session parameters for dates, timestamps, and times. See Session Parameters for Dates, Times, and Timestamps (in this topic).
  - Specifying the format using the following syntax:

    ```
    TO_TIMESTAMP(<value>, '<format>')
    ```

- For strings containing integer values, specify the scale using the following syntax: