CREATE STAGE

Creates a new named *internal* or *external* stage to use for loading data from files into Snowflake tables and unloading data from tables into files:

Internal stage Stores data files internally within Snowflake. For more details, see Choosing an internal stage for local files.

External stage References data files stored in a location outside of Snowflake. Currently, the following cloud storage services are supported:

- Amazon S3 buckets
- Google Cloud Storage buckets
- Microsoft Azure containers

The storage location can be either private/protected or public.

You cannot access data held in archival cloud storage classes that requires restoration before it can be retrieved. These archival storage classes include, for example, the Amazon S3 Glacier Flexible Retrieval or Glacier Deep Archive storage class, or Microsoft Azure Archive Storage.

An internal or external stage can include a *directory table*. Directory tables store a catalog of staged files in cloud storage.

See also:

DROP STAGE , ALTER STAGE , SHOW STAGES , DESCRIBE STAGE

PUT , COPY INTO <table>

COPY INTO <location> , GET

Syntax

```
-- Internal stage
CREATE [ OR REPLACE ] [ { TEMP | TEMPORARY } ] STAGE [ IF NOT EXISTS ]
<internal_stage_name>
    internalStageParams
    directoryTableParams
  [ FILE_FORMAT = ( { FORMAT_NAME = '<file_format_name>' | TYPE = { CSV |
```

```
JSON | AVRO | ORC | PARQUET | XML | CUSTOM } [ formatTypeOptions ] } ) ]
  [ COMMENT = '<string_literal>' ]
  [ [ WITH ] TAG ( <tag_name> = '<tag_value>' [ , <tag_name> = '<tag_value>'
, ... ] ) ]

-- External stage
CREATE [ OR REPLACE ] [ { TEMP | TEMPORARY } ] STAGE [ IF NOT EXISTS ]
<external_stage_name>
    externalStageParams
    directoryTableParams
  [ FILE_FORMAT = ( { FORMAT_NAME = '<file_format_name>' | TYPE = { CSV |
JSON | AVRO | ORC | PARQUET | XML | CUSTOM } [ formatTypeOptions ] } ) ]
  [ COMMENT = '<string_literal>' ]
  [ [ WITH ] TAG ( <tag_name> = '<tag_value>' [ , <tag_name> = '<tag_value>'
, ... ] ) ]
```

Where:

```
internalStageParams ::=
  [ ENCRYPTION = (TYPE = 'SNOWFLAKE_FULL' | TYPE = 'SNOWFLAKE_SSE') ]
```

```
externalStageParams (for Amazon S3) ::=
  URL = '<protocol>://<bucket>[/<path>/]'

  [ { STORAGE_INTEGRATION = <integration_name> } | { CREDENTIALS = ( {  {
AWS_KEY_ID = '<string>' AWS_SECRET_KEY = '<string>' [ AWS_TOKEN =
'<string>' ] } | AWS_ROLE = '<string>'  } ) } ]
  [ ENCRYPTION = ( [ TYPE = 'AWS_CSE' ] [ MASTER_KEY = '<string>' ] |
                   [ TYPE = 'AWS_SSE_S3' ] |
                   [ TYPE = 'AWS_SSE_KMS' [ KMS_KEY_ID = '<string>' ] ] |
                   [ TYPE = 'NONE' ] ) ]
```

```
externalStageParams (for Google Cloud Storage) ::=
  URL = 'gcs://<bucket>[/<path>/]'
  [ STORAGE_INTEGRATION = <integration_name> ]
  [ ENCRYPTION = ( [ TYPE = 'GCS_SSE_KMS' ] [ KMS_KEY_ID = '<string>' ] |
[ TYPE = 'NONE' ] ) ]
```

```
externalStageParams (for Microsoft Azure) ::=
  URL = 'azure://<account>.blob.core.windows.net/<container>[/<path>/]'
  [ { STORAGE_INTEGRATION = <integration_name> } | { CREDENTIALS = ( [
AZURE_SAS_TOKEN = '<string>' ] ) } ]
   [ ENCRYPTION = ( [ TYPE = 'AZURE_CSE' ] [ MASTER_KEY = '<string>' ] | [
TYPE = 'NONE' ] ) ]
```

```
externalStageParams (for Amazon S3-compatible Storage) ::=
  URL = 's3compat://{bucket}[/{path}/]'
  ENDPOINT = '<s3_api_compatible_endpoint>'
  [ { CREDENTIALS = ( AWS_KEY_ID = '<string>' AWS_SECRET_KEY = '<string>'
) } ]
```

```
directoryTableParams (for internal stages) ::=
  [ DIRECTORY = ( ENABLE = { TRUE | FALSE }
                  [ REFRESH_ON_CREATE =  { TRUE | FALSE } ] ) ]
```

```
directoryTableParams (for Amazon S3) ::=
  [ DIRECTORY = ( ENABLE = { TRUE | FALSE }
                  [ REFRESH_ON_CREATE =  { TRUE | FALSE } ]
                  [ AUTO_REFRESH = { TRUE | FALSE } ] ) ]
```

```
directoryTableParams (for Google Cloud Storage) ::=
  [ DIRECTORY = ( ENABLE = { TRUE | FALSE }
                  [ AUTO_REFRESH = { TRUE | FALSE } ]
                  [ REFRESH_ON_CREATE =  { TRUE | FALSE } ]
                  [ NOTIFICATION_INTEGRATION =
'<notification_integration_name>' ] ) ]
```

```
directoryTableParams (for Microsoft Azure) ::=
  [ DIRECTORY = ( ENABLE = { TRUE | FALSE }
                  [ REFRESH_ON_CREATE =  { TRUE | FALSE } ]
                  [ AUTO_REFRESH = { TRUE | FALSE } ]
                  [ NOTIFICATION_INTEGRATION =
'<notification_integration_name>' ] ) ]
```

```
formatTypeOptions ::=
-- If TYPE = CSV
     COMPRESSION = AUTO | GZIP | BZ2 | BROTLI | ZSTD | DEFLATE |
RAW_DEFLATE | NONE
     RECORD_DELIMITER = '<string>' | NONE
     FIELD_DELIMITER = '<string>' | NONE
     FILE_EXTENSION = '<string>'
     PARSE_HEADER = TRUE | FALSE
     SKIP_HEADER = <integer>
     SKIP_BLANK_LINES = TRUE | FALSE
     DATE_FORMAT = '<string>' | AUTO
     TIME_FORMAT = '<string>' | AUTO
     TIMESTAMP_FORMAT = '<string>' | AUTO
     BINARY_FORMAT = HEX | BASE64 | UTF8
     ESCAPE = '<character>' | NONE
```

```
     ESCAPE_UNENCLOSED_FIELD = '<character>' | NONE
     TRIM_SPACE = TRUE | FALSE
     FIELD_OPTIONALLY_ENCLOSED_BY = '<character>' | NONE
     NULL_IF = ( '<string>' [ , '<string>' ... ] )
     ERROR_ON_COLUMN_COUNT_MISMATCH = TRUE | FALSE
     REPLACE_INVALID_CHARACTERS = TRUE | FALSE
     EMPTY_FIELD_AS_NULL = TRUE | FALSE
     SKIP_BYTE_ORDER_MARK = TRUE | FALSE
     ENCODING = '<string>' | UTF8
-- If TYPE = JSON
     COMPRESSION = AUTO | GZIP | BZ2 | BROTLI | ZSTD | DEFLATE |
RAW_DEFLATE | NONE
     DATE_FORMAT = '<string>' | AUTO
     TIME_FORMAT = '<string>' | AUTO
     TIMESTAMP_FORMAT = '<string>' | AUTO
     BINARY_FORMAT = HEX | BASE64 | UTF8
     TRIM_SPACE = TRUE | FALSE
     NULL_IF = ( '<string>' [ , '<string>' ... ] )
     FILE_EXTENSION = '<string>'
     ENABLE_OCTAL = TRUE | FALSE
     ALLOW_DUPLICATE = TRUE | FALSE
     STRIP_OUTER_ARRAY = TRUE | FALSE
     STRIP_NULL_VALUES = TRUE | FALSE
     REPLACE_INVALID_CHARACTERS = TRUE | FALSE
     IGNORE_UTF8_ERRORS = TRUE | FALSE
     SKIP_BYTE_ORDER_MARK = TRUE | FALSE
-- If TYPE = AVRO
     COMPRESSION = AUTO | GZIP | BROTLI | ZSTD | DEFLATE | RAW_DEFLATE |
NONE
     TRIM_SPACE = TRUE | FALSE
     REPLACE_INVALID_CHARACTERS = TRUE | FALSE
     NULL_IF = ( '<string>' [ , '<string>' ... ] )
-- If TYPE = ORC
     TRIM_SPACE = TRUE | FALSE
     REPLACE_INVALID_CHARACTERS = TRUE | FALSE
     NULL_IF = ( '<string>' [ , '<string>' ... ] )
-- If TYPE = PARQUET
     COMPRESSION = AUTO | LZO | SNAPPY | NONE
     SNAPPY_COMPRESSION = TRUE | FALSE
     BINARY_AS_TEXT = TRUE | FALSE
     USE_LOGICAL_TYPE = TRUE | FALSE
     TRIM_SPACE = TRUE | FALSE
     REPLACE_INVALID_CHARACTERS = TRUE | FALSE
     NULL_IF = ( '<string>' [ , '<string>' ... ] )
-- If TYPE = XML
     COMPRESSION = AUTO | GZIP | BZ2 | BROTLI | ZSTD | DEFLATE |
RAW_DEFLATE | NONE
     IGNORE_UTF8_ERRORS = TRUE | FALSE
     PRESERVE_SPACE = TRUE | FALSE
     STRIP_OUTER_ELEMENT = TRUE | FALSE
     DISABLE_SNOWFLAKE_DATA = TRUE | FALSE
     DISABLE_AUTO_CONVERT = TRUE | FALSE
     REPLACE_INVALID_CHARACTERS = TRUE | FALSE
     SKIP_BYTE_ORDER_MARK = TRUE | FALSE
```

Note
Do not specify copy options using the CREATE STAGE, ALTER STAGE, CREATE TABLE, or ALTER TABLE commands. We recommend that you use the COPY INTO <table> command to specify copy options.

Format type options (`formatTypeOptions`)

Depending on the file format type specified (`FILE_FORMAT = ( TYPE = ... )`), you can include one or more of the following format-specific options (separated by blank spaces, commas, or new lines):

TYPE = CSV

`COMPRESSION = AUTO | GZIP | BZ2 | BROTLI | ZSTD | DEFLATE | RAW_DEFLATE | NONE`

Use Data loading, data unloading, and external tables

Definition

- When loading data, specifies the current compression algorithm for the data file. Snowflake uses this option to detect how an ***already-compressed*** data file was compressed so that the compressed data in the file can be extracted for loading.
- When unloading data, compresses the data file using the specified compression algorithm.

Values

Supported Values	Notes
`AUTO`	When loading data, compression algorithm detected automatically, except for Brotli-compressed files, which cannot currently be detected automatically. When unloading data, files are automatically compressed using the default, which is gzip.
`GZIP`	
`BZ2`	
`BROTLI`	Must be specified when loading/unloading Brotli-compressed files.
`ZSTD`	Zstandard v0.8 (and higher) is supported.
`DEFLATE`	Deflate-compressed files (with zlib header, RFC1950).

Supported Values	Notes
`RAW_DEFLATE`	Raw Deflate-compressed files (without header, RFC1951).
`NONE`	When loading data, indicates that the files have not been compressed. When unloading data, specifies that the unloaded files are not compressed.

Default `AUTO`

`RECORD_DELIMITER = '<string>' | NONE`

Use Data loading, data unloading, and external tables

Definition One or more singlebyte or multibyte characters that separate records in an input file (data loading) or unloaded file (data unloading). Accepts common escape sequences or the following singlebyte or multibyte characters:

Singlebyte characters Octal values (prefixed by `\\`) or hex values (prefixed by `0x` or `\x`). For example, for records delimited by the circumflex accent (`^`) character, specify the octal (`\\136`) or hex (`0x5e`) value.

Multibyte characters Hex values (prefixed by `\x`). For example, for records delimited by the cent (`¢`) character, specify the hex (`\xC2\xA2`) value.

The delimiter for RECORD_DELIMITER or FIELD_DELIMITER cannot be a substring of the delimiter for the other file format option (e.g. `FIELD_DELIMITER = 'aa' RECORD_DELIMITER = 'aabb'`).

The specified delimiter must be a valid UTF-8 character and not a random sequence of bytes. Also note that the delimiter is limited to a maximum of 20 characters.

Also accepts a value of `NONE`.

Default **Data loading** New line character. Note that “new line” is logical such that `\r\n` will be understood as a new line for files on a Windows platform.

Data unloading New line character (`\n`).

`FIELD_DELIMITER = '<string>' | NONE`

Use Data loading, data unloading, and external tables

Definition One or more singlebyte or multibyte characters that separate fields in an input file (data loading) or unloaded file (data unloading). Accepts common escape sequences or the following singlebyte or multibyte characters:

Singlebyte characters Octal values (prefixed by `\\`) or hex values (prefixed by `0x` or `\x`). For example, for records delimited by the circumflex accent (`^`) character, specify the octal (`\\136`) or hex (`0x5e`) value.

Multibyte characters Hex values (prefixed by `\x`). For example, for records delimited by the cent (`¢`) character, specify the hex (`\xC2\xA2`) value.

The delimiter for RECORD_DELIMITER or FIELD_DELIMITER cannot be a substring of the delimiter for the other file format option (e.g. `FIELD_DELIMITER = 'aa' RECORD_DELIMITER = 'aabb'`).

> **Note**
> For non-ASCII characters, you must use the hex byte sequence value to get a deterministic behavior.

The specified delimiter must be a valid UTF-8 character and not a random sequence of bytes. Also note that the delimiter is limited to a maximum of 20 characters.

Also accepts a value of `NONE`.

Default comma (`,`)

`FILE_EXTENSION = '<string>' | NONE`

Use Data unloading only

Definition Specifies the extension for files unloaded to a stage. Accepts any extension. The user is responsible for specifying a file extension that can be read by any

desired software or services.

Default null, meaning the file extension is determined by the format type: `.csv[<compression>]`, where `<compression>` is the extension added by the compression method, if `COMPRESSION` is set.

> **Note**
> If the `SINGLE` copy option is `TRUE`, then the COPY command unloads a file without a file extension by default. To specify a file extension, provide a file name and extension in the `<internal_location>` or `<external_location>` path (e.g. `copy into @stage/data.csv`).

`PARSE_HEADER = TRUE | FALSE`

Use Data loading only

Definition Boolean that specifies whether to use the first row headers in the data files to determine column names.

This file format option is applied to the following actions only:

- Automatically detecting column definitions by using the INFER_SCHEMA function.
- Loading CSV data into separate columns by using the INFER_SCHEMA function and MATCH_BY_COLUMN_NAME copy option.

If the option is set to TRUE, the first row headers will be used to determine column names. The default value FALSE will return column names as c*, where * is the position of the column.

> **Note**
> - This option isn't supported for external tables.
> - The SKIP_HEADER option isn't supported if you set `PARSE_HEADER = TRUE`.

Default: `FALSE`

`SKIP_HEADER = <integer>`

Use Data loading and external tables

Definition Number of lines at the start of the file to skip.

Note that SKIP_HEADER does not use the RECORD_DELIMITER or FIELD_DELIMITER values to determine what a header line is; rather, it simply skips the specified number of CRLF (Carriage Return, Line Feed)-delimited lines in the file. RECORD_DELIMITER and FIELD_DELIMITER are then used to determine the rows of data to load.

Default `0`

`SKIP_BLANK_LINES = TRUE | FALSE`

Use Data loading and external tables

Definition Boolean that specifies to skip any blank lines encountered in the data files; otherwise, blank lines produce an end-of-record error (default behavior).

Default: `FALSE`

`DATE_FORMAT = '<string>' | AUTO`

Use Data loading and unloading

Definition Defines the format of date values in the data files (data loading) or table (data unloading). If a value is not specified or is `AUTO`, the value for the DATE_INPUT_FORMAT (data loading) or DATE_OUTPUT_FORMAT (data unloading) parameter is used.

Default `AUTO`

`TIME_FORMAT = '<string>' | AUTO`

Use Data loading and unloading

Definition Defines the format of time values in the data files (data loading) or table (data unloading). If a value is not specified or is `AUTO`, the value for the TIME_INPUT_FORMAT (data loading) or TIME_OUTPUT_FORMAT (data unloading) parameter is used.

Default `AUTO`

`TIMESTAMP_FORMAT = <string>' | AUTO`

Use Data loading and unloading

Definition Defines the format of timestamp values in the data files (data loading) or table (data unloading). If a value is not specified or is `AUTO`, the value for the TIMESTAMP_INPUT_FORMAT (data loading) or TIMESTAMP_OUTPUT_FORMAT (data unloading) parameter is used.

Default `AUTO`

`BINARY_FORMAT = HEX | BASE64 | UTF8`

Use Data loading and unloading

Definition Defines the encoding format for binary input or output. The option can be used when loading data into or unloading data from binary columns in a table.

Default `HEX`

`ESCAPE = '<character>' | NONE`

Use Data loading and unloading

Definition A singlebyte character string used as the escape character for enclosed or unenclosed field values. An escape character invokes an alternative interpretation on subsequent characters in a character sequence. You can use the ESCAPE character to interpret instances of the `FIELD_OPTIONALLY_ENCLOSED_BY` character in the data as literals.

Accepts common escape sequences, octal values, or hex values.

Loading data Specifies the escape character for enclosed fields only. Specify the character used to enclose fields by setting `FIELD_OPTIONALLY_ENCLOSED_BY`.

> **Note**
> This file format option supports singlebyte characters only. Note that UTF-8 character encoding represents high-order ASCII characters as multibyte characters. If your data file is encoded with the UTF-8 character set, you cannot specify a high-order ASCII character as the option value.
>
> In addition, if you specify a high-order ASCII character, we recommend that you set the `ENCODING = '<string>'` file format option as the character encoding for your data files to ensure the character is interpreted correctly.

Unloading data If this option is set, it overrides the escape character set for `ESCAPE_UNENCLOSED_FIELD`.

Default `NONE`

`ESCAPE_UNENCLOSED_FIELD = '<character>' | NONE`

Use Data loading, data unloading, and external tables

Definition A singlebyte character string used as the escape character for unenclosed field values only. An escape character invokes an alternative interpretation on subsequent characters in a character sequence. You can use the ESCAPE character to interpret instances of the `FIELD_DELIMITER` or `RECORD_DELIMITER` characters in the data as literals. The escape character can also be used to escape instances of itself in the data.

Accepts common escape sequences, octal values, or hex values.

Loading data Specifies the escape character for unenclosed fields only.

> **Note**
>
> - The default value is `\\`. If a row in a data file ends in the backslash (`\`) character, this character escapes the newline or carriage return character specified for the `RECORD_DELIMITER` file format option. As a result, the load operation treats this row and the next row as a single row of data. To avoid this issue, set the value to `NONE`.
> - This file format option supports singlebyte characters only. Note that UTF-8 character encoding represents high-order ASCII characters as multibyte characters. If your data file is encoded with the UTF-8 character set, you cannot specify a high-order ASCII character as the option value.
>
>   In addition, if you specify a high-order ASCII character, we recommend that you set the `ENCODING = '<string>'` file format option as the character encoding for your data files to ensure the character is interpreted correctly.

Unloading data If `ESCAPE` is set, the escape character set for that file format option overrides this option.

Default backslash (`\\`)

`TRIM_SPACE = TRUE | FALSE`

Use Data loading and external tables

Definition Boolean that specifies whether to remove white space from fields.

For example, if your external database software encloses fields in quotes, but inserts a leading space, Snowflake reads the leading space rather than the opening quotation character as the beginning of the field (i.e. the quotation marks are interpreted as part of the string of field data). Set this option to `TRUE` to remove undesirable spaces during the data load.

As another example, if leading or trailing spaces surround quotes that enclose strings, you can remove the surrounding spaces using this option and the quote character using the `FIELD_OPTIONALLY_ENCLOSED_BY` option. Note that any spaces ***within*** the quotes are preserved. For example, assuming `FIELD_DELIMITER = '|'` and `FIELD_OPTIONALLY_ENCLOSED_BY = '"'`:

```
|"Hello world"|    /* loads as */  >Hello world<
|" Hello world "|  /* loads as */  > Hello world <
| "Hello world" |  /* loads as */  >Hello world<
```

(the brackets in this example are not loaded; they are used to demarcate the beginning and end of the loaded strings)

Default `FALSE`

`FIELD_OPTIONALLY_ENCLOSED_BY = '<character>' | NONE`

Use Data loading, data unloading, and external tables

Definition Character used to enclose strings. Value can be `NONE`, single quote character (`'`), or double quote character (`"`). To use the single quote character, use the octal or hex representation (`0x27`) or the double single-quoted escape (`''`).

Data unloading only When a field in the source table contains this character, Snowflake escapes it using the same character for unloading. For example, if the value is the double quote character and a field contains the string `A "B" C`, Snowflake escapes the double quotes for unloading as follows:

```
A ""B"" C
```

Default `NONE`

`NULL_IF = ( '<string1>' [ , '<string2>' , ... ] )`

Use Data loading, data unloading, and external tables

Definition String used to convert to and from SQL NULL:

- When loading data, Snowflake replaces these values in the data load source with SQL NULL. To specify more than one string, enclose the list of strings in parentheses and use commas to separate each value.

  Note that Snowflake converts all instances of the value to NULL, regardless of the data type. For example, if `2` is specified as a value, all instances of `2` as either a string or number are converted.

  For example:

  ```
  NULL_IF = ('\N', 'NULL', 'NUL', '')
  ```

  Note that this option can include empty strings.
- When unloading data, Snowflake converts SQL NULL values to the first value in the list.

Default `\\N` (i.e. NULL, which assumes the `ESCAPE_UNENCLOSED_FIELD` value is `\\`)

`ERROR_ON_COLUMN_COUNT_MISMATCH = TRUE | FALSE`

Use Data loading only

Definition Boolean that specifies whether to generate a parsing error if the number of delimited columns (i.e. fields) in an input file does not match the number of columns in the corresponding table.

If set to `FALSE`, an error is not generated and the load continues. If the file is successfully loaded:

- If the input file contains records with more fields than columns in the table, the matching fields are loaded in order of occurrence in the file and the remaining fields are not loaded.
- If the input file contains records with fewer fields than columns in the table, the non-matching columns in the table are loaded with NULL values.

This option assumes all the records within the input file are the same length (i.e. a file containing records of varying length return an error regardless of the value specified for this parameter).

Default `TRUE`

Note
When transforming data during loading (i.e. using a query as the source for the COPY command), this option is ignored. There is no requirement for your data files to have the same number and ordering of columns as your target table.

`REPLACE_INVALID_CHARACTERS = TRUE | FALSE`

Use Data loading only

Definition Boolean that specifies whether to replace invalid UTF-8 characters with the Unicode replacement character (`�`).

If set to `TRUE`, Snowflake replaces invalid UTF-8 characters with the Unicode replacement character.

If set to `FALSE`, the load operation produces an error when invalid UTF-8 character encoding is detected.

Default `FALSE`

`EMPTY_FIELD_AS_NULL = TRUE | FALSE`

Use Data loading, data unloading, and external tables

Definition

- When loading data, specifies whether to insert SQL NULL for empty fields in an input file, which are represented by two successive delimiters (e.g. `,,`).

  If set to `FALSE`, Snowflake attempts to cast an empty field to the corresponding column type. An empty string is inserted into columns of type STRING. For other column types, the COPY command produces an error.

- When unloading data, this option is used in combination with `FIELD_OPTIONALLY_ENCLOSED_BY`. When `FIELD_OPTIONALLY_ENCLOSED_BY = NONE`, setting `EMPTY_FIELD_AS_NULL = FALSE` specifies to unload empty strings in tables to empty string values without quotes enclosing the field values.

  If set to `TRUE`, `FIELD_OPTIONALLY_ENCLOSED_BY` must specify a character to enclose strings.

Default `TRUE`

`SKIP_BYTE_ORDER_MARK = TRUE | FALSE`

Use Data loading only

Definition Boolean that specifies whether to skip the BOM (byte order mark), if present in a data file. A BOM is a character code at the beginning of a data file that defines the byte order and encoding form.

If set to `FALSE`, Snowflake recognizes any BOM in data files, which could result in the BOM either causing an error or being merged into the first column in the table.

Default `TRUE`

`ENCODING = '<string>'`

Use Data loading and external tables

Definition String (constant) that specifies the character set of the source data when loading data into a table.

Character Set	`ENCODING` Value	Supported Languages	Notes
Big5	`BIG5`	Traditional Chinese	
EUC-JP	`EUCJP`	Japanese	
EUC-KR	`EUCKR`	Korean	
GB18030	`GB18030`	Chinese	
IBM420	`IBM420`	Arabic	
IBM424	`IBM424`	Hebrew	
IBM949	`IBM949`	Korean	
ISO-2022-CN	`ISO2022CN`	Simplified Chinese	
ISO-2022-JP	`ISO2022JP`	Japanese	

Character Set	`ENCODING` Value	Supported Languages	Notes
ISO-2022-KR	`ISO2022KR`	Korean	
ISO-8859-1	`ISO88591`	Danish, Dutch, English, French, German, Italian, Norwegian, Portuguese, Swedish	
ISO-8859-2	`ISO88592`	Czech, Hungarian, Polish, Romanian	
ISO-8859-5	`ISO88595`	Russian	
ISO-8859-6	`ISO88596`	Arabic	
ISO-8859-7	`ISO88597`	Greek	
ISO-8859-8	`ISO88598`	Hebrew	
ISO-8859-9	`ISO88599`	Turkish	
ISO-8859-15	`ISO885915`	Danish, Dutch, English, French, German, Italian, Norwegian, Portuguese, Swedish	Identical to ISO-8859-1 except for 8 characters, including the Euro currency symbol.
KOI8-R	`KOI8R`	Russian	
Shift_JIS	`SHIFTJIS`	Japanese	

Character Set	`ENCODING` Value	Supported Languages	Notes
UTF-8	`UTF8`	All languages	For loading data from delimited files (CSV, TSV, etc.), UTF-8 is the default. For loading data from all other supported file formats (JSON, Avro, etc.), as well as unloading data, UTF-8 is the only supported character set.
UTF-16	`UTF16`	All languages	
UTF-16BE	`UTF16BE`	All languages	
UTF-16LE	`UTF16LE`	All languages	
UTF-32	`UTF32`	All languages	
UTF-32BE	`UTF32BE`	All languages	
UTF-32LE	`UTF32LE`	All languages	
windows-874	`WINDOWS874`	Thai	
windows-949	`WINDOWS949`	Korean	
windows-1250	`WINDOWS1250`	Czech, Hungarian, Polish, Romanian	
windows-1251	`WINDOWS1251`	Russian	
windows-1252	`WINDOWS1252`	Danish, Dutch, English, French, German, Italian, Norwegian, Portuguese, Swedish	

Character Set	`ENCODING` Value	Supported Languages	Notes
windows-1253	`WINDOWS1253`	Greek	
windows-1254	`WINDOWS1254`	Turkish	
windows-1255	`WINDOWS1255`	Hebrew	
windows-1256	`WINDOWS1256`	Arabic	

Default `UTF8`

> **Note**
> Snowflake stores all data internally in the UTF-8 character set. The data is converted into UTF-8 before it is loaded into Snowflake.

TYPE = JSON

`COMPRESSION = AUTO | GZIP | BZ2 | BROTLI | ZSTD | DEFLATE | RAW_DEFLATE | NONE`

Use Data loading and external tables

Definition

- When loading data, specifies the current compression algorithm for the data file. Snowflake uses this option to detect how an ***already-compressed*** data file was compressed so that the compressed data in the file can be extracted for loading.
- When unloading data, compresses the data file using the specified compression algorithm.

Values

Supported Values	Notes
`AUTO`	When loading data, compression algorithm detected automatically, except for Brotli-compressed files, which cannot currently be detected automatically. When unloading data, files are automatically compressed using the default, which is gzip.

Supported Values	Notes
`GZIP`	
`BZ2`	
`BROTLI`	Must be specified if loading/unloading Brotli-compressed files.
`ZSTD`	Zstandard v0.8 (and higher) is supported.
`DEFLATE`	Deflate-compressed files (with zlib header, RFC1950).
`RAW_DEFLATE`	Raw Deflate-compressed files (without header, RFC1951).
`NONE`	When loading data, indicates that the files have not been compressed. When unloading data, specifies that the unloaded files are not compressed.

Default `AUTO`

`DATE_FORMAT = '<string>' | AUTO`

Use Data loading only

Definition Defines the format of date string values in the data files. If a value is not specified or is `AUTO`, the value for the DATE_INPUT_FORMAT parameter is used.

This file format option is applied to the following actions only:

- Loading JSON data into separate columns using the MATCH_BY_COLUMN_NAME copy option.
- Loading JSON data into separate columns by specifying a query in the COPY statement (i.e. COPY transformation).

Default `AUTO`

`TIME_FORMAT = '<string>' | AUTO`

Use Data loading only

Definition Defines the format of time string values in the data files. If a value is not specified or is `AUTO`, the value for the TIME_INPUT_FORMAT parameter is used.

This file format option is applied to the following actions only:

- Loading JSON data into separate columns using the MATCH_BY_COLUMN_NAME copy option.
- Loading JSON data into separate columns by specifying a query in the COPY statement (i.e. COPY transformation).

Default `AUTO`

`TIMESTAMP_FORMAT = <string>' | AUTO`

Use Data loading only

Definition Defines the format of timestamp string values in the data files. If a value is not specified or is `AUTO`, the value for the TIMESTAMP_INPUT_FORMAT parameter is used.

This file format option is applied to the following actions only:

- Loading JSON data into separate columns using the MATCH_BY_COLUMN_NAME copy option.
- Loading JSON data into separate columns by specifying a query in the COPY statement (i.e. COPY transformation).

Default `AUTO`

`BINARY_FORMAT = HEX | BASE64 | UTF8`

Use Data loading only

Definition Defines the encoding format for binary string values in the data files. The option can be used when loading data into binary columns in a table.

This file format option is applied to the following actions only:

- Loading JSON data into separate columns using the MATCH_BY_COLUMN_NAME copy option.
- Loading JSON data into separate columns by specifying a query in the COPY statement (i.e. COPY transformation).

Default `HEX`

`TRIM_SPACE = TRUE | FALSE`

Use Data loading only

Definition Boolean that specifies whether to remove leading and trailing white space from strings.

For example, if your external database software encloses fields in quotes, but inserts a leading space, Snowflake reads the leading space rather than the opening quotation character as the beginning of the field (i.e. the quotation marks are interpreted as part of the string of field data). Set this option to `TRUE` to remove undesirable spaces during the data load.

This file format option is applied to the following actions only when loading JSON data into separate columns using the MATCH_BY_COLUMN_NAME copy option.

Default `FALSE`

`NULL_IF = ( '<string1>' [ , '<string2>' , ... ] )`

Use Data loading only

Definition String used to convert to and from SQL NULL. Snowflake replaces these strings in the data load source with SQL NULL. To specify more than one string, enclose the list of strings in parentheses and use commas to separate each value.

This file format option is applied to the following actions only when loading JSON data into separate columns using the MATCH_BY_COLUMN_NAME copy option.

Note that Snowflake converts all instances of the value to NULL, regardless of the data type. For example, if `2` is specified as a value, all instances of `2` as either a string or number are converted.

For example:

```
NULL_IF = ('\N', 'NULL', 'NUL', '')
```

Note that this option can include empty strings.

Default `\\N` (i.e. NULL, which assumes the `ESCAPE_UNENCLOSED_FIELD` value is `\\`)

`FILE_EXTENSION = '<string>' | NONE`

Use Data unloading only

Definition Specifies the extension for files unloaded to a stage. Accepts any extension. The user is responsible for specifying a file extension that can be read by any desired software or services.

Default null, meaning the file extension is determined by the format type: `.json[<compression>]`, where `<compression>` is the extension added by the compression method, if `COMPRESSION` is set.

`ENABLE_OCTAL = TRUE | FALSE`

Use Data loading only

Definition Boolean that enables parsing of octal numbers.

Default `FALSE`

`ALLOW_DUPLICATE = TRUE | FALSE`

Use Data loading and external tables

Definition Boolean that specifies to allow duplicate object field names (only the last one will be preserved).

Default `FALSE`

`STRIP_OUTER_ARRAY = TRUE | FALSE`

Use Data loading and external tables

Definition Boolean that instructs the JSON parser to remove outer brackets (i.e. `[ ]`).

Default `FALSE`

`STRIP_NULL_VALUES = TRUE | FALSE`

Use Data loading and external tables

Definition Boolean that instructs the JSON parser to remove object fields or array elements containing `null` values. For example, when set to `TRUE`:

Before	After
`[null]`	`[]`
`[null,null,3]`	`[,,3]`
`{"a":null,"b":null,"c":123}`	`{"c":123}`
`{"a":[1,null,2],"b":{"x":null,"y":88}}`	`{"a":[1,,2],"b":{"y":88}}`

Default `FALSE`

`REPLACE_INVALID_CHARACTERS = TRUE | FALSE`

Use Data loading and external table

Definition Boolean that specifies whether to replace invalid UTF-8 characters with the Unicode replacement character (`�`). This option performs a one-to-one character replacement.

Values If set to `TRUE`, Snowflake replaces invalid UTF-8 characters with the Unicode replacement character.

If set to `FALSE`, the load operation produces an error when invalid UTF-8 character encoding is detected.

Default `FALSE`

`IGNORE_UTF8_ERRORS = TRUE | FALSE`

Use Data loading and external table

Definition Boolean that specifies whether UTF-8 encoding errors produce error conditions. It is an alternative syntax for `REPLACE_INVALID_CHARACTERS`.

Values If set to `TRUE`, any invalid UTF-8 sequences are silently replaced with the Unicode character `U+FFFD` (i.e. “replacement character”).

If set to `FALSE`, the load operation produces an error when invalid UTF-8 character encoding is detected.

Default `FALSE`

`SKIP_BYTE_ORDER_MARK = TRUE | FALSE`

Use Data loading only

Definition Boolean that specifies whether to skip the BOM (byte order mark), if present in a data file. A BOM is a character code at the beginning of a data file that defines the byte order and encoding form.

If set to `FALSE`, Snowflake recognizes any BOM in data files, which could result in the BOM either causing an error or being merged into the first column in the table.

Default `TRUE`

TYPE = AVRO

`COMPRESSION = AUTO | GZIP | BROTLI | ZSTD | DEFLATE | RAW_DEFLATE | NONE`

Use Data loading only

Definition

- When loading data, specifies the current compression algorithm for the data file. Snowflake uses this option to detect how an ***already-compressed*** data file was compressed so that the compressed data in the file can be extracted for loading.
- When unloading data, compresses the data file using the specified compression algorithm.

Values

Supported Values	Notes
`AUTO`	When loading data, compression algorithm detected automatically, except for Brotli-compressed files, which cannot currently be detected automatically. When unloading data, files are automatically compressed using the default, which is gzip.
`GZIP`	
`BROTLI`	Must be specified if loading/unloading Brotli-compressed files.
`ZSTD`	Zstandard v0.8 (and higher) is supported.
`DEFLATE`	Deflate-compressed files (with zlib header, RFC1950).
`RAW_DEFLATE`	Raw Deflate-compressed files (without header, RFC1951).

Supported Values	Notes
`NONE`	When loading data, indicates that the files have not been compressed. When unloading data, specifies that the unloaded files are not compressed.

Default `AUTO`.

Note
We recommend that you use the default `AUTO` option because it will determine both the file and codec compression. Specifying a compression option refers to the compression of files, not the compression of blocks (codecs).

`TRIM_SPACE = TRUE | FALSE`

Use Data loading only

Definition Boolean that specifies whether to remove leading and trailing white space from strings.

For example, if your external database software encloses fields in quotes, but inserts a leading space, Snowflake reads the leading space rather than the opening quotation character as the beginning of the field (i.e. the quotation marks are interpreted as part of the string of field data). Set this option to `TRUE` to remove undesirable spaces during the data load.

This file format option is applied to the following actions only when loading Avro data into separate columns using the MATCH_BY_COLUMN_NAME copy option.

Default `FALSE`

`REPLACE_INVALID_CHARACTERS = TRUE | FALSE`

Use Data loading and external table

Definition Boolean that specifies whether to replace invalid UTF-8 characters with the Unicode replacement character (`�`). This option performs a one-to-one character replacement.

Values If set to `TRUE`, Snowflake replaces invalid UTF-8 characters with the Unicode replacement character.

If set to `FALSE`, the load operation produces an error when invalid UTF-8 character encoding is detected.

Default `FALSE`

`NULL_IF = ( '<string1>' [ , '<string2>' , ... ] )`

Use Data loading only

Definition String used to convert to and from SQL NULL. Snowflake replaces these strings in the data load source with SQL NULL. To specify more than one string, enclose the list of strings in parentheses and use commas to separate each value.

This file format option is applied to the following actions only when loading Avro data into separate columns using the MATCH_BY_COLUMN_NAME copy option.

Note that Snowflake converts all instances of the value to NULL, regardless of the data type. For example, if `2` is specified as a value, all instances of `2` as either a string or number are converted.

For example:

```
NULL_IF = ('\N', 'NULL', 'NUL', '')
```

Note that this option can include empty strings.

Default `\\N` (i.e. NULL, which assumes the `ESCAPE_UNENCLOSED_FIELD` value is `\\`)

TYPE = ORC

`TRIM_SPACE = TRUE | FALSE`

Use Data loading and external tables

Definition Boolean that specifies whether to remove leading and trailing white space from strings.

For example, if your external database software encloses fields in quotes, but inserts a leading space, Snowflake reads the leading space rather than the opening quotation character as the beginning of the field (i.e. the quotation marks are interpreted as part of the string of field data). Set this option to `TRUE` to remove undesirable spaces during the data load.

This file format option is applied to the following actions only when loading Orc data into separate columns using the MATCH_BY_COLUMN_NAME copy option.

Default `FALSE`

`REPLACE_INVALID_CHARACTERS = TRUE | FALSE`

Use Data loading and external table

Definition Boolean that specifies whether to replace invalid UTF-8 characters with the Unicode replacement character (`�`). This option performs a one-to-one character replacement.

Values If set to `TRUE`, Snowflake replaces invalid UTF-8 characters with the Unicode replacement character.

If set to `FALSE`, the load operation produces an error when invalid UTF-8 character encoding is detected.

Default `FALSE`

`NULL_IF = ( '<string1>' [ , '<string2>' , ... ] )`

Use Data loading and external tables

Definition String used to convert to and from SQL NULL. Snowflake replaces these strings in the data load source with SQL NULL. To specify more than one string, enclose the list of strings in parentheses and use commas to separate each value.

This file format option is applied to the following actions only when loading Orc data into separate columns using the MATCH_BY_COLUMN_NAME copy option.

Note that Snowflake converts all instances of the value to NULL, regardless of the data type. For example, if `2` is specified as a value, all instances of `2` as either a string or number are converted.

For example:

```
NULL_IF = ('\N', 'NULL', 'NUL', '')
```

Note that this option can include empty strings.

Default `\\N` (i.e. NULL, which assumes the `ESCAPE_UNENCLOSED_FIELD` value is `\\`)

TYPE = PARQUET

`COMPRESSION = AUTO | LZO | SNAPPY | NONE`

Use Data loading, data unloading, and external tables

Definition

- When loading data, specifies the current compression algorithm for columns in the Parquet files.
- When unloading data, compresses the data file using the specified compression algorithm.

Values

Supported Values	Notes
`AUTO`	When loading data, compression algorithm detected automatically. Supports the following compression algorithms: Brotli, gzip, Lempel-Ziv-Oberhumer (LZO), LZ4, Snappy, or Zstandard v0.8 (and higher). When unloading data, unloaded files are compressed using the Snappy compression algorithm by default.
`LZO`	When unloading data, files are compressed using the Snappy algorithm by default. If unloading data to LZO-compressed files, specify this value.
`SNAPPY`	When unloading data, files are compressed using the Snappy algorithm by default. You can optionally specify this value.
`NONE`	When loading data, indicates that the files have not been compressed. When unloading data, specifies that the unloaded files are not compressed.

Default `AUTO`

`SNAPPY_COMPRESSION = TRUE | FALSE`

Use Data unloading only

Supported Values	Notes
`AUTO`	Unloaded files are compressed using the Snappy compression algorithm by default.
`SNAPPY`	May be specified if unloading Snappy-compressed files.
`NONE`	When loading data, indicates that the files have not been compressed. When unloading data, specifies that the unloaded files are not

Supported Values	Notes
	compressed.

Definition Boolean that specifies whether unloaded file(s) are compressed using the SNAPPY algorithm.

> **Note**
> ***Deprecated.*** Use `COMPRESSION = SNAPPY` instead.

Limitations Only supported for data unloading operations.

Default `TRUE`

`BINARY_AS_TEXT = TRUE | FALSE`

Use Data loading and external tables

Definition Boolean that specifies whether to interpret columns with no defined logical data type as UTF-8 text. When set to `FALSE`, Snowflake interprets these columns as binary data.

Default `TRUE`

> **Note**
> Snowflake recommends that you set BINARY_AS_TEXT to FALSE to avoid any potential conversion issues.

`TRIM_SPACE = TRUE | FALSE`

Use Data loading only

Definition Boolean that specifies whether to remove leading and trailing white space from strings.

For example, if your external database software encloses fields in quotes, but inserts a leading space, Snowflake reads the leading space rather than the opening quotation character as the beginning of the field (i.e. the quotation marks are interpreted as part of the string of field data). Set this option to `TRUE` to remove undesirable spaces during the data load.

This file format option is applied to the following actions only when loading Parquet data into separate columns using the MATCH_BY_COLUMN_NAME copy option.

Default `FALSE`

`USE_LOGICAL_TYPE = TRUE | FALSE`

Use Data loading, data querying in staged files, and schema detection.

Definition Boolean that specifies whether to use Parquet logical types. With this file format option, Snowflake can interpret Parquet logical types during data loading. For more information, see Parquet Logical Type Definitions. To enable Parquet logical types, set USE_LOGICAL_TYPE as TRUE when you create a new file format option.

Limitations Not supported for data unloading.

`USE_VECTORIZED_SCANNER = TRUE | FALSE`

Use Data loading and data querying in staged files

Definition Boolean that specifies whether to use a vectorized scanner for loading Parquet files.

Default `FALSE`. In a future BCR, the default value will be `TRUE`.

Using the vectorized scanner can significantly reduce the latency for loading Parquet files, because this scanner is well suited for the columnar format of a Parquet file. The scanner only downloads relevant sections of the Parquet file into memory, such as the subset of selected columns.

You can only enable the vectorized scanner if the following conditions are met:

- The `ON_ERROR` option must be set to `ABORT_STATEMENT` or `SKIP_FILE`.

  The other values, `CONTINUE`, `SKIP_FILE_<num>`, `'SKIP_FILE_<num>%'` are not supported.

If `USE_VECTORIZED_SCANNER` is set to `TRUE`, the vectorized scanner has the following behaviors:

- The `BINARY_AS_TEXT` option is always treated as `FALSE` and the `USE_LOGICAL_TYPE` option is always treated as `TRUE`, no matter what the actual value is being set to.

- The vectorized scanner supports Parquet map types. The output of scanning a map type is as follows:

```
"my_map":
  {
   "k1": "v1",
   "k2": "v2"
  }
```

- The vectorized scanner shows `NULL` values in the output, as the following example demonstrates:

```
"person":
 {
  "name": "Adam",
  "nickname": null,
  "age": 34,
  "phone_numbers":
  [
    "1234567890",
    "0987654321",
    null,
    "6781234590"
  ]
  }
```

- The vectorized scanner handles Time and Timestamp as follows:

Parquet	Snowflake vectorized scanner
TimeType(isAdjustedToUtc=True/False, unit=MILLIS/MICROS/NANOS)	TIME
TimestampType(isAdjustedToUtc=True, unit=MILLIS/MICROS/NANOS)	TIMESTAMP_LTZ
TimestampType(isAdjustedToUtc=False, unit=MILLIS/MICROS/NANOS)	TIMESTAMP_NTZ
INT96	TIMESTAMP_LTZ

If `USE_VECTORIZED_SCANNER` is set to `FALSE`, the scanner has the following behaviors:

- This option does not support Parquet maps. The output of scanning a map type is as follows:

```
"my_map":
 {
  "key_value":
  [
```

```
    {
            "key": "k1",
            "value": "v1"
        },
        {
            "key": "k2",
            "value": "v2"
        }
      ]
    }
```

- This option does not explicitly show `NULL` values in the scan output, as the following example demonstrates:

```
"person":
 {
  "name": "Adam",
  "age": 34
  "phone_numbers":
  [
   "1234567890",
   "0987654321",
   "6781234590"
  ]
 }
```

- This option handles Time and Timestamp as follows:

Parquet	When USE_LOGICAL_TYPE = TRUE	When USE_LOGICAL_TYPE = FALSE
TimeType(isAdjustedToUtc=True/False, unit=MILLIS/MICROS)	TIME	• TIME (If ConvertedType present) • INTEGER (If ConvertedType not present)
TimeType(isAdjustedToUtc=True/False, unit=NANOS)	TIME	INTEGER
TimestampType(isAdjustedToUtc=True, unit=MILLIS/MICROS)	TIMESTAMP_LTZ	TIMESTAMP_NTZ
TimestampType(isAdjustedToUtc=True, unit=NANOS)	TIMESTAMP_LTZ	INTEGER

Parquet	When USE_LOGICAL_TYPE = TRUE	When USE_LOGICAL_TYPE = FALSE
TimestampType(isAdjustedToUtc=False, unit=MILLIS/MICROS)	TIMESTAMP_NTZ	• TIMESTAMP_LTZ (If ConvertedType present) • INTEGER (If ConvertedType not present)
TimestampType(isAdjustedToUtc=False, unit=NANOS)	TIMESTAMP_NTZ	INTEGER
INT96	TIMESTAMP_NTZ	TIMESTAMP_NTZ

`REPLACE_INVALID_CHARACTERS = TRUE | FALSE`

Use Data loading and external table

Definition Boolean that specifies whether to replace invalid UTF-8 characters with the Unicode replacement character (`�`). This option performs a one-to-one character replacement.

Values If set to `TRUE`, Snowflake replaces invalid UTF-8 characters with the Unicode replacement character.

If set to `FALSE`, the load operation produces an error when invalid UTF-8 character encoding is detected.

Default `FALSE`

`NULL_IF = ( '<string1>' [ , '<string2>' , ... ] )`

Use Data loading only

Definition String used to convert to and from SQL NULL. Snowflake replaces these strings in the data load source with SQL NULL. To specify more than one string, enclose the list of strings in parentheses and use commas to separate each value.

This file format option is applied to the following actions only when loading Parquet data into separate columns using the MATCH_BY_COLUMN_NAME copy option.

Note that Snowflake converts all instances of the value to NULL, regardless of the data type. For example, if `2` is specified as a value, all instances of `2` as either a string or number are converted.

For example:

```
NULL_IF = ('\N', 'NULL', 'NUL', '')
```

Note that this option can include empty strings.

Default `\\N` (i.e. NULL, which assumes the `ESCAPE_UNENCLOSED_FIELD` value is `\\`)

TYPE = XML

PREVIEW FEATURE — OPEN

Available to all accounts.

`COMPRESSION = AUTO | GZIP | BZ2 | BROTLI | ZSTD | DEFLATE | RAW_DEFLATE | NONE`

Use Data loading only

Definition

- When loading data, specifies the current compression algorithm for the data file. Snowflake uses this option to detect how an ***already-compressed*** data file was compressed so that the compressed data in the file can be extracted for loading.
- When unloading data, compresses the data file using the specified compression algorithm.

Values

Supported Values	Notes
`AUTO`	When loading data, compression algorithm detected automatically, except for Brotli-compressed files, which cannot currently be detected automatically. When unloading data, files are automatically compressed using the default, which is gzip.
`GZIP`	
`BZ2`	
`BROTLI`	Must be specified if loading/unloading Brotli-compressed files.

Supported Values	Notes
`ZSTD`	Zstandard v0.8 (and higher) is supported.
`DEFLATE`	Deflate-compressed files (with zlib header, RFC1950).
`RAW_DEFLATE`	Raw Deflate-compressed files (without header, RFC1951).
`NONE`	When loading data, indicates that the files have not been compressed. When unloading data, specifies that the unloaded files are not compressed.

Default `AUTO`

`IGNORE_UTF8_ERRORS = TRUE | FALSE`

Use Data loading and external table

Definition Boolean that specifies whether UTF-8 encoding errors produce error conditions. It is an alternative syntax for `REPLACE_INVALID_CHARACTERS`.

Values If set to `TRUE`, any invalid UTF-8 sequences are silently replaced with the Unicode character `U+FFFD` (i.e. "replacement character").

If set to `FALSE`, the load operation produces an error when invalid UTF-8 character encoding is detected.

Default `FALSE`

`PRESERVE_SPACE = TRUE | FALSE`

Use Data loading only

Definition Boolean that specifies whether the XML parser preserves leading and trailing spaces in element content.

Default `FALSE`

`STRIP_OUTER_ELEMENT = TRUE | FALSE`

Use Data loading only

Definition Boolean that specifies whether the XML parser strips out the outer XML element, exposing 2nd level elements as separate documents.

Default `FALSE`

`DISABLE_SNOWFLAKE_DATA = TRUE | FALSE`

Use Data loading only

Definition Boolean that specifies whether the XML parser disables recognition of Snowflake semi-structured data tags.

Default `FALSE`

`DISABLE_AUTO_CONVERT = TRUE | FALSE`

Use Data loading only

Definition Boolean that specifies whether the XML parser disables automatic conversion of numeric and Boolean values from text to native representation.

Default `FALSE`

`REPLACE_INVALID_CHARACTERS = TRUE | FALSE`

Use Data loading and external table

Definition Boolean that specifies whether to replace invalid UTF-8 characters with the Unicode replacement character (`�`). This option performs a one-to-one character replacement.

Values If set to `TRUE`, Snowflake replaces invalid UTF-8 characters with the Unicode replacement character.

If set to `FALSE`, the load operation produces an error when invalid UTF-8 character encoding is detected.

Default `FALSE`

`SKIP_BYTE_ORDER_MARK = TRUE | FALSE`

Use Data loading only

Definition Boolean that specifies whether to skip any BOM (byte order mark) present in an input file. A BOM is a character code at the beginning of a data file that defines the byte order and encoding form.

If set to `FALSE`, Snowflake recognizes any BOM in data files, which could result in the BOM either causing an error or being merged into the first column in the table.

Default `TRUE`

Required parameters

`<internal_stage_name>` ***or*** ***`<external_stage_name>`***

Specifies the identifier for the stage; must be unique for the schema in which the stage is created.

In addition, the identifier must start with an alphabetic character and cannot contain spaces or special characters unless the entire identifier string is enclosed in double quotes (e.g. `"My object"`). Identifiers enclosed in double quotes are also case-sensitive.

For more details, see Identifier requirements.

Note
When creating an external stage, a URL is also required. For more details, see External Stage Parameters (in this topic).

If a URL is not specified, Snowflake creates an internal stage by default.

Optional parameters

`{ TEMP | TEMPORARY }`

Specifies that the stage created is temporary and will be dropped at the end of the session in which it was created. Note:

- When a temporary *external* stage is dropped, only the stage itself is dropped; the data files are not removed.
- When a temporary *internal* stage is dropped, all of the files in the stage are purged from Snowflake, regardless of their load status. This prevents files in temporary internal

stages from using data storage and, consequently, accruing storage charges. However, this also means that the staged files cannot be recovered through Snowflake once the stage is dropped.

> **Tip**
> If you plan to create and use temporary internal stages, you should maintain copies of your data files outside of Snowflake.

`FILE_FORMAT = ( FORMAT_NAME = '<file_format_name>' )` *or*
`FILE_FORMAT = ( TYPE = CSV | JSON | AVRO | ORC | PARQUET | XML | CUSTOM [ ... ] )`

Specifies the file format for the stage, which can be either:

`FORMAT_NAME = '<file_format_name>'`

Specifies an existing named file format to use for the stage. The named file format determines the format type (CSV, JSON, etc.), as well as any other format options, for the data files loaded using this stage. For more details, see CREATE FILE FORMAT.

`TYPE = CSV | JSON | AVRO | ORC | PARQUET | XML | CUSTOM [ ... ]`

Specifies the type of files for the stage:

- Loading data from a stage (using COPY INTO <table>) accommodates all of the supported format types.
- Unloading data into a stage (using COPY INTO <location>) accommodates `CSV`, `JSON`, or `PARQUET`.

If a file format type is specified, additional format-specific options can be specified. For more details, see Format type options (formatTypeOptions) (in this topic).

The `CUSTOM` format type specifies that the underlying stage holds unstructured data and can only be used with the `FILE_PROCESSOR` copy option.

Default: `TYPE = CSV`

> **Note**

`FORMAT_NAME` and `TYPE` are mutually exclusive; you can only specify one or the other for a stage.

`COMMENT = '<string_literal>'`

Specifies a comment for the stage.

Default: No value

`TAG ( <tag_name> = '<tag_value>' [ , <tag_name> = '<tag_value>' , ... ] )`

Specifies the tag name and the tag string value.

The tag value is always a string, and the maximum number of characters for the tag value is 256.

For information about specifying tags in a statement, see Tag quotas for objects and columns.

Internal stage parameters (`internalStageParams`)

`[ ENCRYPTION = (TYPE = 'SNOWFLAKE_FULL' | TYPE = 'SNOWFLAKE_SSE') ]`

Specifies the type of encryption supported for all files stored on the stage. You cannot change the encryption type after you create the stage.

`TYPE = ...`

Specifies the encryption type used.

Important
If you require Tri-Secret Secure for security compliance, use the `SNOWFLAKE_FULL` encryption type for internal stages. `SNOWFLAKE_SSE` does not support Tri-Secret Secure.

Possible values are:

- `SNOWFLAKE_FULL`: Client-side and server-side encryption. The files are encrypted by a client when it uploads them to the internal stage using PUT. Snowflake uses a 128-bit encryption key by default. You can configure a 256-bit key by setting the CLIENT_ENCRYPTION_KEY_SIZE parameter.

  All files are also automatically encrypted using AES-256 strong encryption on the server side.

- `SNOWFLAKE_SSE`: Server-side encryption only. The files are encrypted when they arrive on the stage by the cloud service where your Snowflake account is hosted.

  Specify server-side encryption if you plan to query pre-signed URLs for your staged files. For more information, see Types of URLs available to access files.

Default: `SNOWFLAKE_FULL`

External stage parameters (`externalStageParams`)

`URL = '<cloud_specific_url>'`

If this parameter is omitted, Snowflake creates an internal stage

Important

- Enclose the URL in single quotes (`''`) in order for Snowflake to identify the string. If the quotes are omitted, any credentials you supply may be displayed in plain text in the history. We strongly recommend verifying the syntax of the CREATE STAGE statement before you execute it.

  When you create a stage in the Snowflake web interface, the interface automatically encloses field values in quotation characters, as needed.

- Append a forward slash (`/`) to the URL to filter to the specified folder path. If the forward slash is omitted, all files and folders starting with the prefix for the specified path are included.

  Note that the forward slash is ***required*** to access and retrieve unstructured data files in the stage.

Amazon S3

`URL = '<protocol>://<bucket>[/<path>/]'`

Specifies the URL for the external location (existing S3 bucket) used to store data files for loading/unloading, where:

- `<protocol>` is one of the following:
  - `s3` refers to S3 storage in public AWS regions outside of China.
  - `s3china` refers to S3 storage in public AWS regions in China.
  - `s3gov` refers to S3 storage in government regions.

  Accessing cloud storage in a government region using a storage integration is limited to Snowflake accounts hosted in the same government region.

  Similarly, if you need to access cloud storage in a region in China, you can use a storage integration only from a Snowflake account hosted in the same region in

China.

In these cases, use the CREDENTIALS parameter in the CREATE STAGE command (rather than using a storage integration) to provide the credentials for authentication.

- `<bucket>` is the name of the S3 bucket.
- `<path>` is an optional case-sensitive path for files in the cloud storage location (i.e. files have names that begin with a common string) that limits the set of files. Paths are alternatively called *prefixes* or *folders* by different cloud storage services.

Google Cloud Storage

`URL = 'gcs://<bucket>[/<path>/]'`

Specifies the URL for the external location (existing GCS bucket) used to store data files for loading/unloading, where:

- `<bucket>` is the name of the GCS bucket.
- `<path>` is an optional case-sensitive path for files in the cloud storage location (i.e. files have names that begin with a common string) that limits the set of files. Paths are alternatively called *prefixes* or *folders* by different cloud storage services.

Microsoft Azure

`URL = 'azure://<account>.blob.core.windows.net/<container>[/<path>/]'`

Specifies the URL for the external location (existing Azure container) used to store data files for loading, where:

- `<account>` is the name of the Azure account (e.g. `myaccount`). Use the `blob.core.windows.net` endpoint for all supported types of Azure blob storage accounts, including Data Lake Storage Gen2.

  Note that currently, accessing Azure blob storage in government regions using a storage integration is limited to Snowflake accounts hosted on Azure in the same government region. Accessing your blob storage from an account hosted outside of the government region using direct credentials is supported.
- `<container>` is the name of the Azure container (e.g. `mycontainer`).
- `<path>` is an optional case-sensitive path for files in the cloud storage location (i.e. files have names that begin with a common string) that limits the set of files. Paths are alternatively called *prefixes* or *folders* by different cloud storage services.

Default: No value (an internal stage is created)

`STORAGE_INTEGRATION = <integration_name>` *or*
`CREDENTIALS = ( <cloud_specific_credentials> )`

Required only if the storage location is private/protected; not required for public buckets/containers

Amazon S3

`STORAGE_INTEGRATION = <integration_name>`

Specifies the name of the storage integration used to delegate authentication responsibility for external cloud storage to a Snowflake identity and access management (IAM) entity. For more details, see CREATE STORAGE INTEGRATION.

Note

- We highly recommend the use of storage integrations. This option avoids the need to supply cloud storage credentials using the CREDENTIALS parameter when creating stages or loading data.
- Accessing S3 storage in government regions using a storage integration is limited to Snowflake accounts hosted on AWS in the same government region. Accessing your S3 storage from an account hosted outside of the government region using direct credentials is supported.

`CREDENTIALS = ( AWS_KEY_ID = '<string>' AWS_SECRET_KEY = '<string>' [ AWS_TOKEN = '<string>' ] )` *or*
`CREDENTIALS = ( AWS_ROLE = '<string>' )`

Specifies the security credentials for connecting to AWS and accessing the private/protected S3 bucket where the files to load/unload are staged. For more information, see Configuring secure access to Amazon S3.

The credentials you specify depend on whether you associated the Snowflake access permissions for the bucket with an AWS IAM (Identity & Access Management) user or role:

- **IAM user:** IAM credentials are required. Temporary (aka “scoped”) credentials are generated by AWS Security Token Service (STS) and consist of three components:
  - `AWS_KEY_ID`
  - `AWS_SECRET_KEY`
  - `AWS_TOKEN`

  All three are ***required*** to access a private/protected bucket. After a designated period of time, temporary credentials expire and can no longer be used. You must then generate a new set of valid temporary credentials.

Important
The COPY command also allows permanent (aka “long-term”) credentials to be used; however, for security reasons, Snowflake does ***not*** recommend using them. If you must use permanent credentials, Snowflake recommends periodically generating new permanent credentials for external stages.

- **IAM role:** Omit the security credentials and access keys and, instead, identify the role using `AWS_ROLE` and specify the AWS role ARN (Amazon Resource Name).

Google Cloud Storage

`STORAGE_INTEGRATION = <integration_name>`

Specifies the name of the storage integration used to delegate authentication responsibility for external cloud storage to a Snowflake identity and access management (IAM) entity. For more details, see CREATE STORAGE INTEGRATION.

Microsoft Azure

`STORAGE_INTEGRATION = <integration_name>`

Specifies the name of the storage integration used to delegate authentication responsibility for external cloud storage to a Snowflake identity and access management (IAM) entity. For more details, see CREATE STORAGE INTEGRATION.

Note

- We highly recommend the use of storage integrations. This option avoids the need to supply cloud storage credentials using the CREDENTIALS parameter when creating stages or loading data.
- Accessing Azure blob storage in government regions using a storage integration is limited to Snowflake accounts hosted on Azure in the same government region. Accessing your blob storage from an account hosted outside of the government region using direct credentials is supported.

`CREDENTIALS = ( AZURE_SAS_TOKEN = '<string>' )`

Specifies the SAS (shared access signature) token for connecting to Azure and accessing the private/protected container where the files containing loaded data are staged. Credentials are generated by Azure.

Default: No value (no credentials are provided for the external stage)

`ENCRYPTION = ( <cloud_specific_encryption> )`

Required only for loading from/unloading into encrypted files; not required if storage location and files are unencrypted

Data loading Modifies the encryption settings used to decrypt encrypted files in the storage location and extract data.

Data unloading Modifies the encryption settings used to encrypt files unloaded to the storage location.

Amazon S3

```
ENCRYPTION = ( [ TYPE = 'AWS_CSE' ] [ MASTER_KEY = '<string>' ] | [ TYPE = 'AWS_SSE_S3' ] | [ TYPE = 'AWS_SSE_KMS' [ KMS_KEY_ID = '<string>' ] ] | [ TYPE = 'NONE' ] )
```

`TYPE = ...`

Specifies the encryption type used. Possible values are:

- `AWS_CSE`: Client-side encryption (requires a `MASTER_KEY` value). Currently, the client-side master key you provide can only be a symmetric key. Note that, when a `MASTER_KEY` value is provided, Snowflake assumes `TYPE = AWS_CSE` (i.e. when a `MASTER_KEY` value is provided, `TYPE` is not required).
- `AWS_SSE_S3`: Server-side encryption that requires no additional encryption settings.
- `AWS_SSE_KMS`: Server-side encryption that accepts an optional `KMS_KEY_ID` value.

For more information about the encryption types, see the AWS documentation for client-side encryption or server-side encryption.

- `NONE`: No encryption.

`MASTER_KEY = '<string>'` (applies to `AWS_CSE` encryption only)

Specifies the client-side master key used to encrypt the files in the bucket. The master key must be a 128-bit or 256-bit key in Base64-encoded form.

`KMS_KEY_ID = '<string>'` (applies to `AWS_SSE_KMS` encryption only)

Optionally specifies the ID for the AWS KMS-managed key used to encrypt files ***unloaded*** into the bucket. If no value is provided, your default KMS key ID is used to encrypt files on unload.

Note that this value is ignored for data loading.

Google Cloud Storage

```
ENCRYPTION = ( [ TYPE = 'GCS_SSE_KMS' | 'NONE' ] [ KMS_KEY_ID = '<string>' ] )
```

`TYPE = ...`

Specifies the encryption type used. Possible values are:

- `GCS_SSE_KMS`: Server-side encryption that accepts an optional `KMS_KEY_ID` value.

  For more information, see the Google Cloud Platform documentation:

  - https://cloud.google.com/storage/docs/encryption/customer-managed-keys
  - https://cloud.google.com/storage/docs/encryption/using-customer-managed-keys
- `NONE`: No encryption.

`KMS_KEY_ID = '<string>'` (applies to `GCS_SSE_KMS` encryption only)

Optionally specifies the ID for the Cloud KMS-managed key that is used to encrypt files ***unloaded*** into the bucket. If no value is provided, your default KMS key ID set on the bucket is used to encrypt files on unload.

Note that this value is ignored for data loading. The load operation should succeed if the service account has sufficient permissions to decrypt data in the bucket.

Microsoft Azure

```
ENCRYPTION = ( [ TYPE = 'AZURE_CSE' | 'NONE' ] [ MASTER_KEY = '<string>' ] )
```

`TYPE = ...`

Specifies the encryption type used. Possible values are:

- `AZURE_CSE`: Client-side encryption (requires a MASTER_KEY value). For information, see the Client-side encryption information in the Microsoft Azure documentation.
- `NONE`: No encryption.

`MASTER_KEY = '<string>'` (applies to AZURE_CSE encryption only)

Specifies the client-side master key used to encrypt or decrypt files. The master key must be a 128-bit or 256-bit key in Base64-encoded form.

External stage parameters for Amazon S3-compatible storage (`externalStageParams`)

`URL = 's3compat://<bucket>[/<path>/]'`

Specifies the URL for the external location (existing bucket accessed using an S3-compatible API endpoint) used to store data files, where:

- `<bucket>` is the name of the bucket.
- `<path>` is an optional case-sensitive path (or *prefix* in S3 terminology) for files in the cloud storage location (i.e. files with names that begin with a common string).

`ENDPOINT = '<s3_api_compatible_endpoint>'`

Fully-qualified domain that points to the S3-compatible API endpoint.

Directory table parameters (`directoryTableParams`)

`ENABLE = < TRUE | FALSE >`

Specifies whether to add a directory table to the stage. When the value is TRUE, a directory table is created with the stage.

> **Note**
> Setting this parameter to TRUE is ***not*** supported for S3-compatible external stages. The metadata for S3-compatible external stages cannot be refreshed automatically.

Default: `FALSE`

External stages

Amazon S3

`REFRESH_ON_CREATE = < TRUE | FALSE >`

Specifies whether to automatically refresh the directory table metadata once, immediately after the stage is created. Refreshing the directory table metadata synchronizes the metadata with the current list of data files in the specified stage path. This action is required for the metadata to register any ***existing*** data files in the named stage specified in the `URL =` setting.

`TRUE`

Snowflake automatically refreshes the directory table metadata once after the stage creation.

> **Note**
> If the specified cloud storage URL contains close to 1 million files or more, we recommend that you set `REFRESH_ON_CREATE = FALSE`. After creating the stage, refresh the directory table metadata incrementally by executing ALTER STAGE ... REFRESH statements that specify subpaths in the storage location (i.e. subsets of files to include in the refresh) until the metadata includes all of the files in the location.

`FALSE`

Snowflake does not automatically refresh the directory table metadata. To register any data files that exist in the stage, you must manually refresh the directory table metadata once using ALTER STAGE ... REFRESH.

Default: `TRUE`

`AUTO_REFRESH = < TRUE | FALSE >`

Specifies whether Snowflake should enable triggering automatic refreshes of the directory table metadata when ***new or updated*** data files are available in the named external stage specified in the URL value.

`TRUE`

Snowflake enables triggering automatic refreshes of the directory table metadata.

`FALSE`

Snowflake does not enable triggering automatic refreshes of the directory table metadata. You must manually refresh the directory table metadata periodically using ALTER STAGE ... REFRESH to synchronize the metadata with the current list of files in the stage path.

Default: `FALSE`

Google Cloud Storage

`REFRESH_ON_CREATE = < TRUE | FALSE >`

Specifies whether to automatically refresh the directory table metadata once, immediately after the stage is created. Refreshing the directory table metadata synchronizes the metadata with the current list of data files in the specified stage path. This action is required for the metadata to register any ***existing*** data files in the named stage specified in the `URL =` setting.

`TRUE`

Snowflake automatically refreshes the directory table metadata once after the stage creation.

> **Note**
> If the specified cloud storage URL contains close to 1 million files or more, we recommend that you set `REFRESH_ON_CREATE = FALSE`. After creating the stage, refresh the directory table metadata incrementally by executing ALTER STAGE ... REFRESH statements that specify subpaths in the storage location (i.e. subsets of files to include in the refresh) until the metadata includes all of the files in the location.

`FALSE`

Snowflake does not automatically refresh the directory table metadata. To register any data files that exist in the stage, you must manually refresh the directory table metadata once using ALTER STAGE ... REFRESH.

Default: `TRUE`

`AUTO_REFRESH = < TRUE | FALSE >`

Specifies whether Snowflake should enable triggering automatic refreshes of the directory table metadata when ***new or updated*** data files are available in the named external stage specified in the `[ WITH ] LOCATION =` setting.

`TRUE`

Snowflake enables triggering automatic refreshes of the directory table metadata.

`FALSE`

Snowflake does not enable triggering automatic refreshes of the directory table metadata. You must manually refresh the directory table metadata periodically using ALTER STAGE ... REFRESH to synchronize the metadata with the current list of files in the stage path.

`NOTIFICATION_INTEGRATION = '<notification_integration_name>'`

Specifies the name of the notification integration used to automatically refresh the directory table metadata using GCS Pub/Sub notifications. A notification integration is a Snowflake object that provides an interface between Snowflake and third-party cloud message queuing services.

Microsoft Azure

`REFRESH_ON_CREATE = < TRUE | FALSE >`

Specifies whether to automatically refresh the directory table metadata once, immediately after the stage is created. Refreshing the directory table metadata synchronizes the metadata with the current list of data files in the specified stage path. This action is required for the metadata to register any ***existing*** data files in the named stage specified in the `URL =` setting.

`TRUE`

Snowflake automatically refreshes the directory table metadata once after the stage creation.

> **Note**
> If the specified cloud storage URL contains close to 1 million files or more, we recommend that you set `REFRESH_ON_CREATE = FALSE`. After creating the stage, refresh the directory table metadata incrementally by executing ALTER STAGE ... REFRESH statements that specify subpaths in the storage location (i.e. subsets of files to include in the refresh) until the metadata includes all of the files in the location.

`FALSE`

Snowflake does not automatically refresh the directory table metadata. To register any data files that exist in the stage, you must manually refresh the directory table metadata once using ALTER STAGE ... REFRESH.

Default: `TRUE`

`AUTO_REFRESH = < TRUE | FALSE >`

Specifies whether Snowflake should enable triggering automatic refreshes of the directory table metadata when ***new or updated*** data files are available in the named external stage specified in the `[ WITH ] LOCATION =` setting.

`TRUE`

Snowflake enables triggering automatic refreshes of the directory table metadata.

`FALSE`

Snowflake does not enable triggering automatic refreshes of the directory table metadata. You must manually refresh the directory table metadata periodically using ALTER STAGE ... REFRESH to synchronize the metadata with the current list of files in the stage path.

Default: `FALSE`

`NOTIFICATION_INTEGRATION = '<notification_integration_name>'`

Specifies the name of the notification integration used to automatically refresh the directory table metadata using Azure Event Grid notifications. A notification integration is a Snowflake object that provides an interface between Snowflake and third-party cloud message queuing services.

Access control requirements

A role used to execute this SQL command must have the following privileges at a minimum:

Privilege	Object	Notes
USAGE	Storage integration	Required only if accessing a cloud storage service using a storage integration.
CREATE STAGE	Schema	Required only if creating a permanent stage.
OWNERSHIP	Stage	A role must be granted or inherit the OWNERSHIP privilege on the object to create a temporary object that has the same name as the object that already exists in the schema. Note that in a managed access schema, only the schema owner (i.e. the role with the OWNERSHIP privilege on the schema) or a role with the MANAGE GRANTS privilege can grant or revoke privileges on objects in the schema, including future grants.

Note that operating on any object in a schema also requires the USAGE privilege on the parent database and schema.

For instructions on creating a custom role with a specified set of privileges, see Creating custom roles.

For general information about roles and privilege grants for performing SQL actions on securable objects, see Overview of Access Control.

Usage notes

Important
If you require Tri-Secret Secure for security compliance, use the `SNOWFLAKE_FULL` encryption type for internal stages. `SNOWFLAKE_SSE` does not support Tri-Secret Secure.

Caution
Recreating a stage (using CREATE OR REPLACE STAGE) has the following additional, potentially undesirable, outcomes:

- The existing directory table for the stage, if any, is dropped. If the stage is recreated with a directory table, the directory is empty by default.
- The association breaks between the stage and any external table that references it.

  This is because an external table links to a stage using a hidden ID rather than the name of the stage. Behind the scenes, the CREATE OR REPLACE syntax drops an object and recreates it with a different hidden ID.

  If you must recreate a stage after it has been linked to one or more external tables, you must recreate each of the external tables (using CREATE OR REPLACE EXTERNAL TABLE) to reestablish the association. Call the GET_DDL function to retrieve a DDL statement to recreate each of the external tables.
- Any pipes that reference the stage stop loading data. The execution status of the pipes changes to `STOPPED_STAGE_DROPPED`. To resume loading data, these pipe objects must be recreated (using the CREATE OR REPLACE PIPE syntax).

- CREATE STAGE does not check whether the specified URL or credentials are valid. If the credentials are not valid, when you attempt to use the stage, the system returns an error.
- Regarding metadata:

  Attention
  Customers should ensure that no personal data (other than for a User object), sensitive data, export-controlled data, or other regulated data is entered as metadata

when using the Snowflake service. For more information, see Metadata fields in Snowflake.

Examples

Internal stages

Create an internal stage and specify server-side encryption for the stage:

```
CREATE STAGE my_int_stage
  ENCRYPTION = (TYPE = 'SNOWFLAKE_SSE');
```

Create a temporary internal stage with all the same properties as the previous example:

```
CREATE TEMPORARY STAGE my_temp_int_stage;
```

Create a temporary internal stage that references a file format named `my_csv_format` (created using CREATE FILE FORMAT):

```
CREATE TEMPORARY STAGE my_int_stage
  FILE_FORMAT = my_csv_format;
```

When you reference the stage in a COPY INTO <table> statement, the file format options are automatically set.

Create an internal stage that includes a directory table. The stage references a file format named `myformat`:

```
CREATE STAGE mystage
   DIRECTORY = (ENABLE = TRUE)
   FILE_FORMAT = myformat;
```

External stages

Amazon S3

In the examples below, if the S3 bucket is in a region in China, use the `s3china://` protocol for the URL parameter.

Create an external stage using a private/protected S3 bucket named `load` with a folder path named `files`. Secure access to the S3 bucket is provided via the `myint` storage integration:

```
CREATE STAGE my_ext_stage
  URL='s3://load/files/'
  STORAGE_INTEGRATION = myint;
```

Create an external stage using a private/protected S3 bucket named `load` with a folder path named `files`. The Snowflake access permissions for the S3 bucket are associated with an IAM user; therefore, IAM credentials are required:

```
CREATE STAGE my_ext_stage1
  URL='s3://load/files/'
  CREDENTIALS=(AWS_KEY_ID='1a2b3c' AWS_SECRET_KEY='4x5y6z');
```

Note that the AWS_KEY_ID and AWS_SECRET_KEY values used in this example are for illustration purposes only.

Create an external stage using an S3 bucket named `load` with a folder path named `encrypted_files` and client-side encryption (default encryption type) with the master key to decrypt/encrypt files stored in the bucket:

```
CREATE STAGE my_ext_stage2
  URL='s3://load/encrypted_files/'
  CREDENTIALS=(AWS_KEY_ID='1a2b3c' AWS_SECRET_KEY='4x5y6z')
  ENCRYPTION=(MASTER_KEY = 'eSx...');
```

Create an external stage using an S3 bucket named `load` with a folder path named `encrypted_files` and AWS_SSE_KMS server-side encryption with the ID for the master key to decrypt/encrypt files stored in the bucket:

```
CREATE STAGE my_ext_stage3
  URL='s3://load/encrypted_files/'
  CREDENTIALS=(AWS_KEY_ID='1a2b3c' AWS_SECRET_KEY='4x5y6z')
  ENCRYPTION=(TYPE='AWS_SSE_KMS' KMS_KEY_ID = 'aws/key');
```

Same example as the immediately preceding example, except that the Snowflake access permissions for the S3 bucket as associated with an IAM role instead of an IAM user. Note that credentials are handled separately from other stage parameters such as `ENCRYPTION`. Support for these other parameters is the same regardless of the credentials used to access your external S3 bucket:

```
CREATE STAGE my_ext_stage3
  URL='s3://load/encrypted_files/'
  CREDENTIALS=(AWS_ROLE='arn:aws:iam::001234567890:role/mysnowflakerole')
  ENCRYPTION=(TYPE='AWS_SSE_KMS' KMS_KEY_ID = 'aws/key');
```

Create a stage with a directory table in the active schema for the user session. The cloud storage URL includes the path `files`. The stage references a storage integration named `my_storage_int`:

```
CREATE STAGE mystage
  URL='s3://load/files/'
  STORAGE_INTEGRATION = my_storage_int
  DIRECTORY = (
    ENABLE = true
    AUTO_REFRESH = true
  );
```

Google Cloud Storage

Create an external stage using a private/protected GCS bucket named `load` with a folder path named `files`. Secure access to the GCS bucket is provided via the `myint` storage integration:

```
CREATE STAGE my_ext_stage
  URL='gcs://load/files/'
  STORAGE_INTEGRATION = myint;
```

Create a stage named `mystage` with a directory table in the active schema for the user session. The cloud storage URL includes the path `files`. The stage references a storage integration named `my_storage_int`:

```
CREATE STAGE mystage
  URL='gcs://load/files/'
  STORAGE_INTEGRATION = my_storage_int
  DIRECTORY = (
    ENABLE = true
    AUTO_REFRESH = true
    NOTIFICATION_INTEGRATION = 'MY_NOTIFICATION_INT'
  );
```

Microsoft Azure

Create an external stage using a private/protected Azure container named `load` with a folder path named `files`. Secure access to the container is provided via the `myint` storage integration:

```
CREATE STAGE my_ext_stage
  URL='azure://myaccount.blob.core.windows.net/load/files/'
  STORAGE_INTEGRATION = myint;
```

Create an external stage using an Azure storage account named `myaccount` and a container named `mycontainer` with a folder path named `files` and client-side encryption enabled:

```
CREATE STAGE mystage
  URL='azure://myaccount.blob.core.windows.net/mycontainer/files/'
  CREDENTIALS=(AZURE_SAS_TOKEN='?sv=2016-05-
31&ss=b&srt=sco&sp=rwdl&se=2018-06-27T10:05:50Z&st=2017-06-
27T02:05:50Z&spr=https,http&sig=bgqQwoXwxzuD2GJfagRg7VOS8hzNr3QLT7rhS8OFRL
Q%3D')
  ENCRYPTION=(TYPE='AZURE_CSE' MASTER_KEY = 'kPx...');
```

(The `AZURE_SAS_TOKEN` and `MASTER_KEY` values used in this example are not actual values; they are provided for illustration purposes only.)

Create a stage with a directory table in the active schema for the user session. The cloud storage URL includes the path `files`. The stage references a storage integration named `my_storage_int`:

```
CREATE STAGE mystage
  URL='azure://myaccount.blob.core.windows.net/load/files/'
  STORAGE_INTEGRATION = my_storage_int
  DIRECTORY = (
    ENABLE = true
    AUTO_REFRESH = true
    NOTIFICATION_INTEGRATION = 'MY_NOTIFICATION_INT'
  );
```