# Contents

*What's New in the ORC Engine* ......................................................... v

## PART 1  Getting Started  1

**Chapter 1 / Introduction and Requirements** ................................... 3  
  About the ORC Engine ............................................................... 3  
  Requirements .............................................................................. 3

**Chapter 2 / Quick Tutorials** ......................................................... 7  
  Example: Read an ORC Table ...................................................... 7  
  Example: Create an ORC Table .................................................. 9  
  Example: View Information about an ORC Library and Table .......... 10  
  Example: View Information about a Table by Using ORC Tools ........ 13

## PART 2  Syntax  15

**Chapter 3 / LIBNAME Statement and LIBNAME Statement Options** .......... 17  
  Dictionary .................................................................................. 17

**Chapter 4 / Data Set Options** ......................................................... 31  
  Dictionary .................................................................................. 31

## PART 3  Usage  35

**Chapter 5 / Data Types** .............................................................. 37  
  Conversion between ORC and SAS Data Types .............................. 37  
  Restrictions on Data Types .......................................................... 42

**Chapter 6 / Restrictions** ............................................................ 43  
  Restrictions for SAS Features ..................................................... 43  
  Restrictions for ORC File Features .............................................. 44

**Chapter 7 / Optimizing Performance** ............................................. 45  
  Using Compression ..................................................................... 45  
  Setting the Stripe Size ............................................................... 45  
  Using the DROP= and KEEP= Data Set Options .......................... 46  
  Using the FIRSTOBS= and OBS= Data Set Options ...................... 46
What's New in the ORC Engine

Apply the Available Hot Fix

You are encouraged to download and apply a hot fix that is available for the ORC engine. The hot fix significantly improves performance. This document has been updated to reflect the behavior when the hot fix has been applied.
What's New in the ORC Engine
PART 1

Getting Started

Chapter 1
Introduction and Requirements ........................................... 3

Chapter 2
Quick Tutorials ................................................................. 7
Introduction and Requirements

About the ORC Engine

The ORC engine is a Base SAS LIBNAME engine that is supported in the SAS Viya programming environment. The ORC engine provides the following benefits:

- Apache ORC (Optimized Row Columnar) is an open-source file format that was created for use with Apache Hive. ORC uses column-based storage, which can be highly efficient for analysis of big data. In column-based storage, you quickly access only the columns that you need to query.

- In SAS, the ORC engine can create or access an ORC table that is stored in one of the following storage systems:
  - Microsoft Azure Data Lake Storage Gen2
  - Linux for x64 distributions that are supported by SAS.

- You can use SAS to manage your ORC data along with other data sources. You can use the Base SAS programming language to query ORC tables, similar to other data sources.

Requirements

SAS Requirements

You access ORC tables in the SAS Viya programming environment by using the ORC LIBNAME statement. A data connector is also available for SAS Cloud.

Here are the requirements to use the engine:

- SAS Viya 3.5 on Linux for x64 (data representation LINUX_X86_64).
- ORC files must conform to Apache ORC 1.5.6. (See also “Restrictions for ORC File Features” on page 44.)
- SAS session encoding must be UTF-8. (See also “Restrictions for SAS Features” on page 43.)

The following storage systems are supported:

- Microsoft Azure Data Lake Storage Gen2 (see “Azure Requirements” on page 4)
- Linux for x64 distributions that are supported by SAS

---

**Azure Requirements**

To use Microsoft Azure Data Lake Storage Gen2 as storage for ORC tables:

1. You must have access to a storage account. An on-site administrator can create the storage account and manage access by using the Microsoft Azure portal at portal.azure.com.

2. Specify the required options:
   - "AZURETENANTID= System Option” in SAS System Options: Reference
   - STORAGE_ACCOUNT_NAME= LIBNAME statement option
   - STORAGE_APPLICATION_ID= LIBNAME statement option
   - STORAGE_FILE_SYSTEM= LIBNAME statement option

   The following options have defaults and are optional:
   - "AZUREAUTHCACHELOC System Option” in SAS System Options: Reference
   - STORAGE_DNS_SUFFIX= LIBNAME statement option
   - STORAGE_TIMEOUT= LIBNAME statement option

   The first time you submit a LIBNAME statement to connect to Azure, you are required to authenticate with Microsoft.

3. Use Apache Hive in Azure HDInsight to create ORC tables that SAS can read. You can also create ORC tables by using a tool like Beeline and then upload the tables to Azure blob storage.

   **Note:** SAS can create ORC tables in Azure. However, a table is not created as a directory structure as in Hive, and SAS does not interact with the Hive Metastore.
Using Hive Directories as Tables

Typically in Hive, when you create a table, the data is stored within a directory (or folder). The directory name is the table name. The directory contains one or more files. The directory and its file or files constitute a single table.

SAS can read (but not create) directories as tables. You must specify the LIBNAME statement option DIRECTORIES_AS_DATA=YES and follow the restrictions. See the details in that topic for other important limitations.

Naming Conventions

You must follow these naming conventions for ORC tables:

- Unlike most SAS components, the ORC engine is case sensitive. Specify table and column names with exactly the same casing as in the data source. For example, if you request SAS to read mytable, and the table is named MyTable in the data source, then SAS cannot find the table. File extensions are also case sensitive.

- The ORC engine supports one default file extension, orc. To access files that have other extensions, you must specify the FILE_NAME_EXTENSION=LIBNAME statement option. See also “Using Hive Directories as Tables” on page 5.

- The engine can read ORC tables that contain a period in the table name when you specify DIRECTORIES_AS_DATA=YES. However, for tables that are not stored as directories, periods in a table name are not recommended. Using name-literal syntax or special characters in table names can cause unexpected results. Therefore it is recommended to use the default VALIDMEMNAME=COMPATIBLE system option.

- Names must follow any naming conventions for the open-source file format and for the storage system.
Example: Read an ORC Table

Example Code

This example reads an ORC table from an Azure location. The inventory table can be created by using the following Hive commands:

```
CREATE TABLE inventory(item INT, descr CHAR(32), price DOUBLE, day DATE) STORED AS ORC LOCATION '/mydata/inventory';
INSERT INTO inventory VALUES(4600, 'incandescent A19', 15.54, '2019-10-17');
INSERT INTO inventory VALUES(1023, 'led A19', 14.97, '2019-10-17');
INSERT INTO inventory VALUES(1023, 'incandescent B10', 5.49, '2019-10-16');
INSERT INTO inventory VALUES(4219, 'incandescent E12', 3.97, '2019-10-15');
SELECT * FROM inventory;
```

Here is the SAS code to read the table:

```
libname mylib orc '/data/user' storage_account_name = myacct
    storage_application_id = 'b1fc955d5c-e0e2-45b3-a3cc-a1cf54120f';
```
storage_file_system = myorcdata
directories_as_data=yes /*2*/
;
proc print data=mylib.inventory noobs; /*3*/
run;

1 The LIBNAME statement assigns the mylib libref to the orc engine and a location in Azure, followed by several connection options.

2 The LIBNAME statement option DIRECTORIES_AS_DATA=YES enables you to access Hive tables that are stored as a directory.

3 The PRINT procedure prints the ORC table mylib.inventory.

Output 2.1  PROC PRINT Output of ORC Table mylib.inventory

<table>
<thead>
<tr>
<th>item</th>
<th>descr</th>
<th>price</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>4500</td>
<td>incandescent A10</td>
<td>15.54</td>
<td>17OCT2019</td>
</tr>
<tr>
<td>1023</td>
<td>led A19</td>
<td>14.97</td>
<td>17OCT2019</td>
</tr>
<tr>
<td>1023</td>
<td>incandescent B10</td>
<td>5.49</td>
<td>16OCT2019</td>
</tr>
<tr>
<td>4219</td>
<td>incandescent E12</td>
<td>3.97</td>
<td>15OCT2019</td>
</tr>
</tbody>
</table>

Key Ideas

- Hive creates a directory for each table. To read these tables, specify the LIBNAME statement option DIRECTORIES_AS_DATA=YES.
- SAS can read ORC files that are stored in Microsoft Azure Data Lake Storage Gen2 or in Linux for x64. See “Requirements” on page 3.

See Also

- “Requirements” on page 3
- "LIBNAME Statement: ORC Engine" on page 17
- “PRINT Procedure” in Base SAS Procedures Guide
Example: Create an ORC Table

Example Code

In this example, a DATA step creates a simple ORC table in a Linux file system. Each variable uses a function to render a datetime, date, or time value when the program runs. The example also demonstrates the use of informats.

```sas
libname mylib orc "*/u/mydata" ;
data mylib.test;
  var1= datetime();
  format var1 datetime21.2;
  informat var1 datetime21.2; /*1*/
  var2= date();
  format var2 date9.;
  informat var2 date9.;       /*2*/
  var3= time();
  format var3 time.;
  informat var3 time.;        /*3*/
run;
proc contents data=mylib.test;    /*4*/
run;
proc print data=mylib.test;       /*5*/
run;
```

1. A SAS NUMERIC with DATETIME21.2 informat converts to an ORC TIMESTAMP data type.
2. A SAS NUMERIC with DATE9. informat converts to an ORC DATE data type.
3. A SAS NUMERIC with TIME. informat converts to an ORC DOUBLE data type.
4. The CONTENTS procedure shows that only var1 and var2 are assigned a format and informat.
5. In the output shown below for the PRINT procedure, notice that var3 is not displayed as a time value. The column was converted to a DOUBLE.

**Output 2.2**  Portion of PROC CONTENTS Output Showing Formats and Informats

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Format</th>
<th>Informat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>var1</td>
<td>Num</td>
<td>8</td>
<td>DATETIME21.2</td>
<td>DATETIME21.2</td>
</tr>
<tr>
<td>2</td>
<td>var2</td>
<td>Num</td>
<td>8</td>
<td>DATE9.</td>
<td>DATE9.</td>
</tr>
<tr>
<td>3</td>
<td>var3</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In the SAS Viya programming environment, you can use the ORC engine to read and write Apache ORC tables. The engine can create an ORC table in Microsoft Azure Data Lake Storage Gen2 or in Linux for x64. See “Requirements” on page 3.

- SAS uses an informat when writing to ORC. SAS uses a format when reading from ORC and when displaying data.

- When a format is not specified for a numeric value, SAS uses the BESTw. format as the default format. This format is not appropriate for displaying dates.

See Also

- Chapter 6, “Restrictions,” on page 43
- Chapter 5, “Data Types,” on page 37

Example: View Information about an ORC Library and Table

Example Code

The following DATASETS procedure prints information about an ORC table. The inventory table is created in “Example: Read an ORC Table” on page 7.

```
libname mylib orc '/data/user'        /* 1 */
    storage_account_name = myacct
    storage_application_id = 'b1fc955d5c-e0e2-45b3-a3cc-a1cf54120f'
    storage_file_system = myorcdata
    directories_as_data=yes            /* 2 */
    ;
proc datasets lib=mylib nolist;      /* 3 */
    contents data=inventory directory; /* 4 */
run;
quit;
```
The LIBNAME statement assigns the `mylib` libref to the orc engine and a location in Azure, followed by several connection options.

The LIBNAME statement option `DIRECTORIES_AS_DATA=YES` enables you to access Hive tables that are stored as directories.

In PROC DATASETS, the `LIB=` option specifies the `mylib` library.

The CONTENTS statement prints information about the `mylib.inventory` table. The `DIRECTORY` option specifies to print information about the contents of the `mylib` library.

In the Directory portion of the output, you can see that the directory contains three tables. One of the tables, `new`, is stored in a single file. The other two tables are stored as files within a directory. In Azure, the DATASETS procedure reports 0 for the file size of these tables, even though files within the directory contain data.

**Output 2.4  PROC DATASETS Output of Directory Information**

<table>
<thead>
<tr>
<th>Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Libref</td>
</tr>
<tr>
<td>Engine</td>
</tr>
<tr>
<td>Physical Name</td>
</tr>
<tr>
<td>File extension list</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Member Type</th>
<th>File Size</th>
<th>Last Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>inventory</td>
<td>DATA</td>
<td>0</td>
<td>10/16/2019 09:14:43</td>
</tr>
<tr>
<td>2</td>
<td>new</td>
<td>DATA</td>
<td>157505</td>
<td>10/16/2019 16:39:35</td>
</tr>
<tr>
<td>3</td>
<td>orders</td>
<td>DATA</td>
<td>0</td>
<td>10/17/2019 19:35:13</td>
</tr>
</tbody>
</table>

The following portions of output are for the `mylib.inventory` table.

**Output 2.5  Portion of CONTENTS Statement Output Showing Table Metadata**

<table>
<thead>
<tr>
<th>Data Set Name</th>
<th>MYLIB.inventory</th>
<th>Observations</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member Type</td>
<td>DATA</td>
<td>Variables</td>
<td>4</td>
</tr>
<tr>
<td>Engine</td>
<td>ORC</td>
<td>Indexes</td>
<td>0</td>
</tr>
<tr>
<td>Created</td>
<td>10/19/2019 05:14:43</td>
<td>Observation Length</td>
<td>56</td>
</tr>
<tr>
<td>Last Modified</td>
<td>10/19/2019 05:14:43</td>
<td>Deleted Observations</td>
<td>0</td>
</tr>
<tr>
<td>Protection</td>
<td>Compressed</td>
<td>ZLIB</td>
<td></td>
</tr>
<tr>
<td>Data Set Type</td>
<td>Sorted</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Representation</td>
<td>Default</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encoding</td>
<td>Default</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the Engine/Host portion, the file extension of `mylib.inventory` is listed as _DIR_ because the table is stored as a directory.
Output 2.6  Portion of CONTENTS Statement Output Showing Engine/Host Information

<table>
<thead>
<tr>
<th>Engine/Host Dependent Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File Extension</strong></td>
</tr>
<tr>
<td><strong>Format</strong></td>
</tr>
<tr>
<td><strong>File Version</strong></td>
</tr>
<tr>
<td><strong>Writer Version</strong></td>
</tr>
<tr>
<td><strong>File Length</strong></td>
</tr>
<tr>
<td><strong>Compression Block</strong></td>
</tr>
</tbody>
</table>

Output 2.7  Portion of CONTENTS Statement Output Showing Column Metadata

<table>
<thead>
<tr>
<th>Alphabetic List of Variables and Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Key Ideas

- Hive creates a directory for each table. To read these tables, specify the LIBNAME statement option `DIRECTORIES_AS_DATA=YES`.

- In the directory listing, PROC DATASETS provides a list of the file extensions that are currently specified for the library. If you have not specified the LIBNAME statement option `FILE_NAME_EXTENSION=`, then only the default file extension is shown. For ORC, the default is `orc`.

- In the directory listing, PROC DATASETS can list directories that are not ORC tables. In addition, PROC DATASETS does not report the true file size or file length for a table that is stored as a directory.

- Usual information such as the type of compression is shown in output from PROC CONTENTS or the CONTENTS statement of PROC DATASETS. Information that is specific to the engine and file system is reported in the Engine/Host Dependent Information portion of the output.

- You can also return some information about a SAS library by using the LIST option in the LIBNAME statement.

See Also

- “DIRECTORIES_AS_DATA= LIBNAME Statement Option” on page 22
- “FILE_NAME_EXTENSION= LIBNAME Statement Option” on page 25
Example: View Information about a Table by Using ORC Tools

Example Code

The Apache ORC project provides several tools for working with tables. (These tools are not shipped with SAS.)

The `orc-metadata` command returns the following information about the `test.orc` table. This table is created from the SAS programming environment in "Example: Create an ORC Table" on page 9. ORC data types are shown in the second line for the three variables.

Output 2.8  Portion of Metadata Returned from Apache ORC

```json
{
  "name": "/u/userid/test.orc",
  "type": "struct<var1:timestamp,var2:date,var3:double>",
  "rows": 1,
}
```

Key Ideas

- To see native data types in Hive, you can use the `orc-metadata` command. The command is a C++ tool that returns metadata in JSON format.
- Apache ORC also provides a Java version of this tool. Many other metadata tools exist in the open-source community.

See Also

Chapter 3
LIBNAME Statement and LIBNAME Statement Options ................................ 17

Chapter 4
Data Set Options .................................................................................. 31
LIBNAME Statement and LIBNAME Statement Options

Dictionary

LIBNAME Statement: ORC Engine
Assigns or deassigns a libref (a shortcut name) to a SAS library that contains Apache ORC files.

Valid in: Anywhere
Category: Data Access
Engine: ORC
See: To clear one or all librefs (CLEAR) or to list the characteristics of a SAS library (LIST), see “LIBNAME Statement” in SAS Global Statements: Reference.

Syntax

Form 1: LIBNAME libref ORC 'SAS-library' <options>;
Form 2: LIBNAME libref ORC (library-specification-1 <...library-specification-n>) <options>;
Syntax Description

libref

is a shortcut name for the aggregate storage location where the ORC files are stored. The libref must be a valid SAS name.

Length 1 to 8 bytes

Tip The SAS library assignment lasts only for the duration of the SAS session or until you change it or discontinue it with another LIBNAME statement.

See Other than the rules for length, a libref follows the same rules of syntax as any SAS name. For more information about SAS naming conventions, see “Rules for Most SAS Names” in SAS Language Reference: Concepts

ORC

especifies the ORC engine name.

'SAS-library'

is the physical name for the SAS library. The physical name is the path name that is recognized by the operating environment. Enclose the physical name in single or double quotation marks.

library-specification

is two or more SAS libraries that are specified by physical names, previously assigned librefs, or a combination of the two. If you specify a physical name, enclose it in single or double quotation marks. Separate each specification with either a blank or a comma and enclose the entire list in parentheses. This form is used for library concatenation (Form 2).

'SAS-library'

is the physical name of a SAS library, enclosed in quotation marks.

libref

is the name of a previously assigned libref.

options

are the LIBNAME statement options. See “LIBNAME Statement and LIBNAME Statement Options” on page 17.

Restrictions Options that are specific to an operating environment (host options) are not supported.

The DLCREATEDIR system option to create a subdirectory is not supported.

Details

Assigning a SAS Library (Form 1)

The SAS library assignment lasts only for the duration of the SAS session or until you change the libref or discontinue it with another LIBNAME statement. The simplest form of the LIBNAME statement specifies only a libref and the physical name of a SAS library. If you omit the engine name, you invoke the default engine.

See “Example: Create an ORC Table” on page 9.
Concatenating SAS Libraries (Form 2)

Use concatenation when you want to reference two or more libraries by using one libref. The libraries that you concatenate can have different LIBNAME statement options.

In the concatenation form of the LIBNAME statement, separate each specification with either a blank or a comma, and enclose the entire list in parentheses. Specify each library with its previously assigned libref or its physical path name.

Physical path names are supported in a concatenation LIBNAME statement for Linux file systems, but not for Microsoft Azure Data Lake Storage Gen2. If you specify a physical path name, enclose it in single or double quotation marks. You can use a combination of librefs and physical path names.

Do not assign LIBNAME statement options in the concatenation LIBNAME statement. Assign options in each LIBNAME statement where they are valid. Then refer to those librefs in the concatenation LIBNAME statement.

For more information, see “Library Concatenation” in SAS Language Reference: Concepts.

BINARY_BYTE_LIMIT= LIBNAME Statement Option

Specifies the maximum length for binary columns that do not have a defined length.

<table>
<thead>
<tr>
<th>Engine:</th>
<th>ORC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction:</td>
<td>This option is used for input only and is ignored for output.</td>
</tr>
</tbody>
</table>

Syntax

**BINARY_BYTE_LIMIT=**\( n \)

Syntax Description

<table>
<thead>
<tr>
<th>( n )</th>
<th>is the maximum length for binary columns that do not have a defined length in the data source.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>32767</td>
</tr>
<tr>
<td>Range</td>
<td>1–32767</td>
</tr>
</tbody>
</table>

Details

SAS reads a BINARY column from the data source as a CHAR column. SAS assigns a $HEXw. format. (See Chapter 5, “Data Types,” on page 37.)

In SAS, a CHAR column must have a defined length. Some open-source file formats do not store a length. In others, storing a length is optional. ORC does not store a length for BINARY. If the length of a BINARY column is not defined in the data source, then the $HEXw format uses the length of the BINARY_BYTE_LIMIT= option. The option is applied only to BINARY columns that do not have a defined length. If a column has a defined length that is reported by the data store, the engine uses that value.
A data value is truncated if the value is longer than the BINARY_BYTE_LIMIT= specification.

A value could be truncated in display (such as the PRINT procedure) because of the $HEXw. format. A hexadecimal value is stored and processed in one byte but requires two bytes to be displayed. For example, the hexadecimal character 0x31 is one byte. PROC PRINT displays that value as two characters, 3 and 1, which together require two bytes. Therefore, when you display the column in SAS, you might need to specify a $HEXw. format that is twice the size of the default format. For more information, see “FORMAT Statement” in SAS DATA Step Statements: Reference.

Comparisons
The data set option takes precedence over the LIBNAME statement option.

CHAR_COLUMN_LIMIT= LIBNAME Statement Option
Specifies the maximum length for character columns that do not have a defined length.

<table>
<thead>
<tr>
<th>Engine</th>
<th>ORC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction</td>
<td>This option is used for input only and is ignored for output.</td>
</tr>
</tbody>
</table>

Syntax
CHAR_COLUMN_LIMIT=n

Syntax Description

\[ n \]

\( n \) is the maximum length for character columns that do not have a defined length in the data source.

| Default  | 32767                |
| Range    | 1–32767              |

Details
SAS reads a CHAR, VARCHAR, or STRING data type from the data source as a CHAR data type. SAS does not assign a default format when reading these data types. (See Chapter 5, “Data Types,” on page 37.)

In SAS, a CHAR column must have a defined length. Some open-source file formats do not store a length. In others, storing a length is optional. If the length of a character column is not defined in the data source, then the CHAR_COLUMN_LIMIT= option is used.

This limit is applied only to character columns that do not have a defined length. If a column has a defined length that is reported by the data store, the engine uses that value.
A character value is truncated if the value is longer than the CHAR_COLUMN_LIMIT= specification.

Comparisons
The data set option takes precedence over the LIBNAME statement option.

COMPRESS= LIBNAME Statement Option
Specifies whether a new output table is compressed.

<table>
<thead>
<tr>
<th>Engine</th>
<th>ORC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default:</td>
<td>ZLIB</td>
</tr>
</tbody>
</table>

Syntax
COMPRESS=NO | ZLIB

Syntax Description

**NO**
specifies that a new table is not compressed.

**ZLIB**
specifies that a new table is compressed by using ZLIB compression.

| Alias | YES |

Details
For the ORC engine, the default is ZLIB compression because it is almost always the best choice. If you do not want compression, then you must specify the COMPRESS=NO data set option or LIBNAME statement option when the table is created. You cannot change compression unless you re-create the table.

SAS can also read compression types SNAPPY, LZO, LZ4, and ZSTD. However, these types are not supported for Write access.

If you use the COPY procedure (or the COPY statement of the DATASETS procedure) to copy an ORC table, be aware of the following interaction: If the output engine does not support ZLIB, the compression is changed to a compression that is supported by the output engine.

Comparisons
The data set option takes precedence over the LIBNAME statement option. The COMPRESS= system option is not supported by the ORC engine.

In other SAS engines, the default is no compression.
DATA_BLOCK_SIZE= LIBNAME Statement Option

Requests a stripe size for an ORC table during creation (output).

Engine: ORC

Syntax

DATA_BLOCK_SIZE=\(n\) | \(nK\) | \(nM\) | \(nG\)

Syntax Description

\(n\) | \(nK\) | \(nM\) | \(nG\)

is the requested stripe size for an ORC table during creation (output) in bytes, kilobytes, megabytes, or gigabytes. If \(n\) is specified without \(K\), \(M\), or \(G\), the value is in bytes. For example, \(data\_block\_size=32768\) requests a stripe size of 32,768 bytes.

Default 4 megabytes

Range 1024 bytes – 2 gigabytes

Details

In an ORC file, the data is grouped in stripes, which are separate and independent groups of row data. The size of the stripes can have an impact on performance. Here are some general guidelines:

- A large number of small stripes might be more efficient for the execution of subsetting queries.
- A small number of large stripes might be more efficient for sequential reads.

The actual stripe size in an ORC table does not exactly match the DATA_BLOCK_SIZE= value. The stripe size is also affected by factors such as compression. To determine the optimal value for DATA_BLOCK_SIZE=, performance testing is recommended.

Comparisons

The data set option takes precedence over the LIBNAME statement option.

DIRECTORIES_AS_DATA= LIBNAME Statement Option

Specifies whether file system directories are used as tables.

Engine: ORC

Valid in: DATASETS procedure

Default: NO
Restrictions:
This option is used for input only and is ignored for output.
The directory must contain ORC files only.
All of the ORC files in the directory must have the same schema.

Examples:
“Example: Read an ORC Table” on page 7
“Example: View Information about an ORC Library and Table” on page 10

Syntax
DIRECTORIES_AS_DATA=YES | NO

Syntax Description
YES
specifies that directories in the library location are accessed as tables.

Interaction
Do not confuse this option with the _NONE_ keyword in the
FILE_NAME_EXTENSION= LIBNAME statement option.

NO
specifies that directories in the library location are not accessed as tables.

Details
Typically in Hive, when you create a table, the data is stored within a directory (or
folder) that has the table name. The directory contains one or more files. The
directory and its file or files constitute a single table.

If you want to read directories as tables, specify the parent directory as the library
location. In the SAS library model, a library location can contain multiple tables.
Therefore, when DIRECTORIES_AS_DATA=YES, each subdirectory within the
library path is considered to be a table.

Here are the essential concepts when you set DIRECTORIES_AS_DATA=YES:

- A library can contain both types of tables: directories-as-data tables and single-
file tables. You can specify both FILE_NAME_EXTENSION= and
DIRECTORIES_AS_DATA=YES in one LIBNAME statement to access both
types of tables. If you specify a list of extensions in FILE_NAME_EXTENSION=,
the order of precedence in the list is left to right. If you do not specify
FILE_NAME_EXTENSION=, the default file extension is used. See
“FILE_NAME_EXTENSION= LIBNAME Statement Option” on page 25.

- Multiple tables can have the same root name but different extensions. For
example, in one library, you could store the directory-as-data table mytable, the
single-file table mytable.orc, and another single-file table mytable.ORC.

Behavior could be unpredictable. Here are some examples of the variety of
behavior when multiple tables have the same root name:

- Usually, the directory-as-data table takes precedence and is operated on.
However, if the directory-as-data table is corrupted (for example, from an
incomplete DELETE operation), then a single-file table that has the same root
name is operated on.

- Do not use the directory listing from PROC DATASETS or from the
DIRECTORY option in the CONTENTS statement. The directory listing
shows only one table for each root name, and the table selection can be
unpredictable. Instead, use the CONTENTS statement or the CONTENTS procedure to specify the table name. In the output, examine the file extension to make sure that you are reporting on the correct table.

- If you want to access a table that is stored as one file (in the example above, mytable.orc or mytable.ORC) and not the table that is stored as a directory (mytable), then you must submit a different library assignment with DIRECTORIES_AS_DATA=NO. Specify the file extension in the FILE_NAME_EXTENSION= LIBNAME statement option.

- If you want to avoid accessing a table that is stored as one file (in the example above, mytable.orc or mytable.ORC), and you want to be certain that you are accessing the table that is stored as a directory (mytable), then specify FILE_NAME_EXTENSION=_NONE_ and DIRECTORIES_AS_DATA=YES.

- In the directory listing, PROC DATASETS can list directories that are not ORC tables. In addition, PROC DATASETS does not report the true file size or file length for a table that is stored as a directory.

- The CONTENTS procedure, or the CONTENTS statement of the DATASETS procedure, shows a table’s file extension in the Engine/Host portion of output. For a table that is stored as a directory, the file extension is shown as _DIR_.

- Do not use SAS language elements to delete a table that is stored as a directory. Typical ways of deleting a table in SAS include the DELETE statement or KILL option in PROC DATASETS, or the DROP TABLE statement in the SQL procedure. After you submit these statements, SAS can no longer access the table but the table is not deleted from storage.

- In Hive, you can use a period (.) in a table name. The table is stored as a directory that contains a file. For example, if you name a table mytable.data.orc, then Hive creates a table that is stored in a directory named mytable.data.orc. Be aware that a period in the directory name might cause you to think that the object is a file instead of a directory.

When DIRECTORIES_AS_DATA=YES, SAS recognizes that the directory is a table. In this case, SAS considers a period and the text that follows a period to be part of the table name, not an extension.

When DIRECTORIES_AS_DATA=NO, periods in a table name are not recommended. Using name-literal syntax or special characters in table names can cause unexpected results. Therefore it is recommended to use the default VALIDMEMNAME=COMPATIBLE system option.

- The FILE_NAME_EXTENSION= LIBNAME statement option does not apply to directory names.

- DIRECTORIES_AS_DATA=YES supports reading, but not writing, a table as a directory. If you copy a table that exists as a directory, the copy is created as a single file, not as a directory that contains files.

---

**CAUTION**

A library can contain multiple files that have the same root name but different extensions. If you do not specify FILE_NAME_EXTENSION=, the default file extension is used. To be certain that you are accessing a directory-as-data table, specify FILE_NAME_EXTENSION=_NONE_ and DIRECTORIES_AS_DATA=YES.
FILE_NAME_EXTENSION= LIBNAME Statement Option

Specifies non-default file extensions for SAS to read and write Apache ORC files.

| Engine:     | ORC |
| Default:    | orc |

Syntax

FILE_NAME_EXTENSION=extension-type | _NONE_
FILE_NAME_EXTENSION=(extension-type-1 ... extension-type-n <_NONE_>)

Syntax Description

extension-type

is one extension or a list of extensions. Do not include the period (.) character in an extension.

If you specify a list of extensions, separate the values by one or more blanks, and enclose the list in parentheses. The order of precedence in a list is left to right. You can include the _NONE_ keyword in the list of extensions.

| Length       | The maximum length for each extension value is 32 bytes. |
| Restriction  | Only alphanumeric characters and the underscore character (_) are supported. |

_NONE_

specifies that files that have no extension are valid. You can specify the _NONE_ keyword alone or in a list of extensions. If you specify a list of extensions, place the _NONE_ keyword in the desired order of precedence.

| Interaction | The _NONE_ keyword enables SAS to access any table that is stored as one file that has no extension. If you want SAS to access tables that are stored as a directory, then use the LIBNAME statement option DIRECTORIES_AS_DATA=YES instead. |

Details

The ORC engine supports one default file extension, orc. If your files have an extension other than the default, then use the LIBNAME statement option FILE_NAME_EXTENSION= to specify the extension or a list of extensions.

Here are the rules for file extensions:

- If you specify an extension or a list of extensions, then they replace the default. If you want to also use the default extension, then you must include it in your list.
- If you want to read or write files that have no extension, then specify the _NONE_ keyword.
- File extensions, like table and column names, are case sensitive.
The file extension or file extension list (or the default if not specified) is used for both reading and writing.

In the directory listing, PROC DATASETS provides a list of the file extensions that are currently specified for the library. If you have not specified the LIBNAME statement option `FILE_NAME_EXTENSION=`, then only the default file extension is shown. For ORC, the default is `orc`.

The CONTENTS procedure, or the CONTENTS statement in the DATASETS procedure, shows a table’s file extension in the Engine/Host portion of output.

The `FILE_NAME_EXTENSION=` option does not apply to directory names. See “DIRECTORIES_AS_DATA= LIBNAME Statement Option” on page 22.

A library can contain multiple files that have the same root name but different extensions. For that reason, you can specify multiple extensions, and _NONE_ if applicable, in a file extension list. SAS uses the file extension list in the following ways:

- In a Write operation, the first value in the extension list is used.
- In a Read operation, only one file is read. When you specify a file, SAS looks for the specified file that has the first value in the extension list. If that file does not exist, then SAS looks for the specified file that has the second value in the extension list, and so on.
- In output from the DATASETS procedure, only one file with the same root name is displayed. The order of precedence is the same as in a Read operation.
- In a Delete operation, the behavior depends on whether you delete a library or a table:
  - If you specify the DELETE statement in PROC DATASETS, SAS deletes one file. The order of precedence for file extensions is the same as in a Read operation.
  - If you specify the KILL option in PROC DATASETS, or the DROP TABLE statement in the SQL procedure, SAS could delete multiple files. In this case, SAS deletes all of the specified files that have a value in the extension list.

If the library also contains tables that are stored as a directory, you can include `DIRECTORIES_AS_DATA=YES` in the LIBNAME statement. See “DIRECTORIES_AS_DATA= LIBNAME Statement Option” on page 22.

**CAUTION**

If you specify a list of extensions, recall that a library can contain multiple files that have the same root name but different extensions. Therefore, SAS could operate on a different file than you intend. Be aware of the order of precedence.

**Example**

In the following example, only files that do not have an extension are valid.

```
file_name_extension=_none_
```

In the following example, only files that have the extension ORC are valid.

```
file_name_extension=(ORC)
```

The following example replaces the default extension with a list of extensions. However, in order to include the default extension, it is specified first in the list. Then
files that have extension ORC, no extension, and OptRC are accessed, in that order of precedence. If SAS writes any new files, the files have the extension orc.

\[
\text{file\_name\_extension=} (\text{orc ORC _none_ OptRC})
\]

In the following example, the _NONE_ keyword is listed first. SAS first looks for files that do not have an extension. Then SAS looks for files that have the extension orc, ORC, BigData, and Optrc, in that order of precedence. If SAS writes any new files, the files do not have an extension.

\[
\text{file\_name\_extension=} (_\text{none_ orc ORC BigData Optrc})
\]

### STORAGE_ACCOUNT_NAME= LIBNAME Statement Option

Specifies the account name for connecting to a cloud storage system.

**Engine:** ORC

**Requirement:** This option is required if a cloud storage system is used. For other requirements, see “Azure Requirements” on page 4.

**Syntax**

\[
\text{STORAGE\_ACCOUNT\_NAME=} name
\]

**Syntax Description**

- **name**
  - specifies the account name for connecting to a cloud storage system.

**Restriction**

If the name includes spaces or non-alphanumeric characters, then enclose the name in single or double quotation marks.

**Example**

```plaintext
libname mylib orc '/data/user'
    storage_account_name = myacct
    storage_application_id = 'b1fc955d5c-e0e2-45b3-a3cc-a1cf54120f'
    storage_file_system = myorcdata
;
```

### STORAGE_APPLICATION_ID= LIBNAME Statement Option

Specifies the application ID for connecting to a cloud storage system.

**Engine:** ORC
This option is required if a cloud storage system is used. For other requirements, see “Azure Requirements” on page 4.

Syntax

```
STORAGE_APPLICATION_ID="string"
```

Syntax Description

"string"

specifies the application ID (a GUID string) from the assignment in the Microsoft Azure portal.

Restriction

If the name includes spaces or non-alphanumeric characters, then enclose the name in single or double quotation marks. Usually this value includes the hyphen (-) character.

Details

This option is used for OAuth 2.0 device flow authentication, which is supported by Microsoft Active Directory (Azure AD).

Example

Here is an example of the option:

```
libname mylib orc '/data/user'
   storage_account_name = myacct
   storage_application_id = 'b1fc955d5c-e0e2-45b3-a3cc-a1cf54120f'
   storage_file_system = myorcdata
;  
```

STORAGE_DNS_SUFFIX= LIBNAME Statement Option

Specifies the network host name for connecting to a cloud storage system.

Engine: ORC

Requirement: Although this option is not required, others are. See “Azure Requirements” on page 4.

Syntax

```
STORAGE_DNS_SUFFIX="name"
```

Syntax Description

"name"

specifies the network host name for connecting to a cloud storage system. Typically, this value is appended to the end of the storage account name and is called a suffix.
If the name includes spaces or non-alphanumeric characters, then enclose the name in single or double quotation marks. Usually this value includes the period (.) character.

Example

Here is an example of the option:

```r
libname mylib orc '/data/user'
   storage_account_name = myacct
   storage_application_id = 'b1fc955d5c-e0e2-45b3-a3cc-a1cf54120f'
   storage_file_system = myorcdata
   storage_dns_suffix = 'dfs.core.windows.net'
;
```

STORAGE_FILE_SYSTEM= LIBNAME Statement Option

Specifies the name of the file system for connecting to a cloud storage system.

**Engine:** ORC

**Requirement:** This option is required if a cloud storage system is used. For other requirements, see “Azure Requirements” on page 4.

**Syntax**

```
STORAGE_FILE_SYSTEM= name
```

**Syntax Description**

`name`

specifies the name of the file system, within the cloud storage system, that files are to be read from and written to.

**Restriction**

If the name includes spaces or non-alphanumeric characters, then enclose the name in single or double quotation marks.

**Example**

Here is an example of the option:

```r
libname mylib orc '/data/user'
   storage_account_name = myacct
   storage_application_id = 'b1fc955d5c-e0e2-45b3-a3cc-a1cf54120f'
   storage_file_system = myorcdata
;
```
STORAGE_TIMEOUT= LIBNAME Statement Option

Specifies the maximum completion time for connecting to a cloud storage system.

**Engine:** ORC

**Requirement:** Although this option is not required, others are. See “Azure Requirements” on page 4.

### Syntax

```
STORAGE_TIMEOUT=n
```

### Syntax Description

- **n** specifies the maximum number of seconds for a storage task to complete. The time-out is not enforced by SAS. If the storage system is enabled to enforce a time-out, then the storage system returns an error if the task does not complete within the specified time.

**Default**

The cloud storage system might have a default.

**Restrictions**

- Only numeric values are supported.
- Do not enclose the value in quotation marks.

### Example

In the following LIBNAME statement, the time-out is 5000 seconds:

```
libname mylib orc '/data/user'
    storage_account_name = myacct
    storage_application_id = 'b1fc955d5c-e0e2-45b3-a3cc-a1cf54120f'
    storage_file_system = myorcdata
    **storage_timeout=5000**
;
```
Data Set Options

Dictionary

**BINARY_BYTE_LIMIT= Data Set Option**

Specifies the maximum length for binary columns that do not have a defined length.

**Engine:** ORC

**Restriction:** This option is used for input only and is ignored for output.

**Syntax**

```
BINARY_BYTE_LIMIT=n
```

**Syntax Description**

- **n** is the maximum length for binary columns that do not have a defined length in the data source.

  **Default** 32767

  **Range** 1–32767

**Details**

SAS reads a BINARY column from the data source as a CHAR column. SAS assigns a $HEXw. format. (See Chapter 5, “Data Types,” on page 37.)

In SAS, a CHAR column must have a defined length. Some open-source file formats do not store a length. In others, storing a length is optional. ORC does not store a length for BINARY. If the length of a BINARY column is not defined in the data source, it is treated as 32767.
source, then the $HEXw format uses the length of the BINARY_BYTE_LIMIT=
option. The option is applied only to BINARY columns that do not have a defined
length. If a column has a defined length that is reported by the data store, the
engine uses that value.

A data value is truncated if the value is longer than the BINARY_BYTE_LIMIT=
specification.

A value could be truncated in display (such as the PRINT procedure) because of the
$HEXw format. A hexadecimal value is stored and processed in one byte but
requires two bytes to be displayed. For example, the hexadecimal character 0x31 is
one byte. PROC PRINT displays that value as two characters, 3 and 1, which
together require two bytes. Therefore, when you display the column in SAS, you
might need to specify a $HEXw format that is twice the size of the default format.
For more information, see “FORMAT Statement” in SAS DATA Step Statements:
Reference.

Comparisons

The data set option takes precedence over the LIBNAME statement option.

---

CHAR_COLUMN_LIMIT= Data Set Option

Specifies the maximum length for character columns that do not have a defined length.

Engine: ORC

Restriction: This option is used for input only and is ignored for output.

Syntax

CHAR_COLUMN_LIMIT=n

Syntax Description

n
is the maximum length for character columns that do not have a defined length in
the data source.

Default 32767
Range 1–32767

Details

SAS reads a CHAR, VARCHAR, or STRING data type from the data source as a
CHAR data type. SAS does not assign a default format when reading these data
types. (See Chapter 5, “Data Types,” on page 37.)

In SAS, a CHAR column must have a defined length. Some open-source file formats
do not store a length. In others, storing a length is optional. If the length of a
character column is not defined in the data source, then the
CHAR_COLUMN_LIMIT= option is used.
This limit is applied only to character columns that do not have a defined length. If a column has a defined length that is reported by the data store, the engine uses that value.

A character value is truncated if the value is longer than the CHAR_COLUMN_LIMIT= specification.

Comparisons
The data set option takes precedence over the LIBNAME statement option.

COMPRESS= Data Set Option
Specifies whether a new output table is compressed.

<table>
<thead>
<tr>
<th>Engine:</th>
<th>ORC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default:</td>
<td>ZLIB</td>
</tr>
</tbody>
</table>

Syntax
COMPRESS=NO | ZLIB

Syntax Description
- **NO** specifies that a new table is not compressed.
- **ZLIB** specifies that a new table is compressed by using ZLIB compression.

Details
For the ORC engine, the default is ZLIB compression because it is almost always the best choice. If you do not want compression, then you must specify the COMPRESS=NO data set option or LIBNAME statement option when the table is created. You cannot change compression unless you re-create the table.

SAS can also read compression types SNAPPY, LZO, LZ4, and ZSTD. However, these types are not supported for Write access.

If you use the COPY procedure (or the COPY statement of the DATASETS procedure) to copy an ORC table, be aware of the following interaction: If the output engine does not support ZLIB, the compression is changed to a compression that is supported by the output engine.

Comparisons
The data set option takes precedence over the LIBNAME statement option. The COMPRESS= system option is not supported by the ORC engine.

In other SAS engines, the default is no compression.
DATA_BLOCK_SIZE= Data Set Option

Requests a stripe size for an ORC table during creation (output).

Engine: ORC

Restriction: This option is used for output only and is ignored for input.

Syntax

DATA_BLOCK_SIZE=n | nK | nM | nG

Syntax Description

n | nK | nM | nG

is the requested stripe size for an ORC table during creation (output) in bytes, kilobytes, megabytes, or gigabytes. If n is specified without K, M, or G, the value is in bytes. For example, data_block_size=32768 requests a stripe size of 32,768 bytes.

Default 4 megabytes

Range 1024 bytes – 2 gigabytes

Details

In an ORC file, the data is grouped in stripes, which are separate and independent groups of row data. The size of the stripes can have an impact on performance. Here are some general guidelines:

- A large number of small stripes might be more efficient for the execution of subsetting queries.
- A small number of large stripes might be more efficient for sequential reads.

The actual stripe size in an ORC table does not exactly match the DATA_BLOCK_SIZE= value. The stripe size is also affected by factors such as compression. To determine the optimal value for DATA_BLOCK_SIZE=, performance testing is recommended.

Comparisons

The data set option takes precedence over the LIBNAME statement option.
PART 3

Usage

Chapter 5
Data Types ................................................................. 37

Chapter 6
Restrictions ............................................................... 43

Chapter 7
Optimizing Performance ............................................. 45
Conversion between ORC and SAS Data Types

The following table explains how SAS uses formats and informats for data type conversion. Here are the essential concepts:

- Formats define how ORC data types convert to SAS data types when SAS reads from ORC.
- Informats (combined with the SAS data type) define how SAS data types convert to ORC data types when SAS writes to ORC. For example, a SAS NUMERIC variable that has a DATETIME informat is written to an ORC table as a TIMESTAMP data type.
- SAS automatically applies formats and informats to columns as they are read from ORC, as shown in the table below. (Notice that some data types are not assigned a format or informat.) For Read operations from ORC, you cannot specify a different format. For example, you cannot apply a format to an ORC column in the PRINT procedure. However, you can specify a different format if you use an ORC table to create a SAS data set. See SAS Formats and Informats: Reference.
- When you create an ORC table from SAS, you must specify the correct informat in order to create the desired ORC data type.
- If you read and replace an ORC table, be aware that changing a column’s informat could change the data type. Also note that some data types can be read but not written. In that case, a different data type is assigned when the table is replaced.
- All lengths are specified in bytes.
### Table 5.1 Default Formats and Informs

<table>
<thead>
<tr>
<th>ORC Data Type</th>
<th>SAS Data Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR</td>
<td>CHAR</td>
<td>(none) ORC might not store a defined column length.</td>
</tr>
</tbody>
</table>
|               |               | When reading from ORC, keep the following in mind:
|               |               | □ If a column length is not defined in the data source, then the format uses the length of the CHAR_COLUMN_LIMIT= data set option or CHAR_COLUMN_LIMIT= LIBNAME statement option. If the option is not specified, then the length is 32,767. |
|               |               | □ Values shorter than the specified length are padded with blanks. |
|               |               | □ Values longer than 32,767 are truncated. |
|               |               | □ Character-based columns (CHAR, VARCHAR, STRING) that contain date, time, or timestamp values are not automatically converted to their respective SAS format types. They remain as character string values. |
|               |               | When writing to ORC, keep the following in mind: |
|               |               | □ If the length of the SAS CHAR is 255 or less, the ORC data type is CHAR. |
|               |               | □ If the length is greater than 255, the ORC data type is VARCHAR. |
## Conversion between ORC and SAS Data Types

<table>
<thead>
<tr>
<th>ORC Data Type</th>
<th>SAS Data Type</th>
<th>Notes</th>
</tr>
</thead>
</table>
| VARCHAR       | CHAR          | - The maximum length in ORC is 65,535.  
|               |               | - ORC might not store a defined column length.  
|               |               | - When reading from ORC, keep the following in mind:  
|               |               |   - If a column length is not defined in the data source, then the format uses the length of the `CHAR_COLUMN_LIMIT=` data set option or `CHAR_COLUMN_LIMIT=` LIBNAME statement option. If the option is not specified, then the length is 32,767.  
|               |               |   - Values shorter than the specified length are padded with blanks.  
|               |               |   - Values longer than 32,767 are truncated.  
|               |               |   - Character-based columns (CHAR, VARCHAR, STRING) that contain date, time, or timestamp values are not automatically converted to their respective SAS format types. They remain as character string values.  
|               |               | - When writing to ORC, keep the following in mind:  
|               |               |   - If the length of the SAS CHAR is 255 or less, the ORC data type is CHAR.  
|               |               |   - If the length is greater than 255, the ORC data type is VARCHAR.  
| STRING        | CHAR          | - ORC does not store a defined column length.  
|               |               | - When reading from ORC, keep the following in mind:  
|               |               |   - If a column length is not defined in the data source, then the format uses the length of the `CHAR_COLUMN_LIMIT=` data set option or `CHAR_COLUMN_LIMIT=` LIBNAME statement option. If the option is not specified, then the length is 32,767.  
|               |               |   - Values shorter than the specified length are padded with blanks.  
|               |               |   - Values longer than 32,767 are truncated.  
|               |               |   - Character-based columns (CHAR, VARCHAR, STRING) that contain date, time, or timestamp values are not automatically converted to their respective SAS format types. They remain as character string values.  
|               |               | - SAS cannot write a STRING data type to ORC.  
|               |               | (none)  

### Notes
- The maximum length in ORC is 65,535.
- ORC might not store a defined column length.
- Character-based columns (CHAR, VARCHAR, STRING) that contain date, time, or timestamp values are not automatically converted to their respective SAS format types. They remain as character string values.
<table>
<thead>
<tr>
<th>ORC Data Type</th>
<th>SAS Data Type</th>
<th>Notes</th>
</tr>
</thead>
</table>
| BINARY | CHAR | 1. ORC does not store a defined column length.  
2. When reading from ORC, the format uses the length of the `BINARY_BYTE_LIMIT=` data set option or `BINARY_BYTE_LIMIT=` LIBNAME statement option if the option is specified. If the option is not specified, then the length is 32,767. (The `BINARY_BYTE_LIMIT=` option affects the format only, not the informat.) Be aware of the documented way to avoid truncation when the data is displayed.  
3. Values longer than 32,767 are truncated.  
4. Values shorter than the specified length are padded with blanks.  
5. When writing to ORC, the length is 32,767. |
| DOUBLE | NUMERIC | 1. When writing to ORC, a SAS NUMERIC that has one of the informats in this table is written as the indicated ORC data type. If a NUMERIC does not have one of the indicated informats, it is written to ORC as a DOUBLE.  
For example, if a NUMERIC has an informat of 11., it is written to ORC as an INT. If a NUMERIC has an informat of 11.3, it is written to ORC as a DOUBLE. |
<p>| FLOAT | NUMERIC | SAS cannot write a FLOAT data type to ORC. |
| TINYINT (8 bit) | NUMERIC | 4. |
| LITTLEINT (16 bit) | NUMERIC | 6. |
| INT (32 bit) | NUMERIC | 11. |</p>
<table>
<thead>
<tr>
<th>ORC Data Type</th>
<th>SAS Data Type</th>
<th>SAS Format (Reading from ORC) and Informat (Writing to ORC)</th>
<th>Notes</th>
</tr>
</thead>
</table>
| BIGINT (64 bit) | NUMERIC | 20. | - When reading from ORC, converting a BIGINT to a NUMERIC can result in loss of precision. The internal SAS eight-byte, floating-point format accurately preserves only 15 digits of precision, and a BIGINT can preserve up to 19 digits of precision. See “Numerical Accuracy in SAS Software” in SAS Language Reference: Concepts.  
- Loss of precision can cause unexpected results from a WHERE clause. The expected rows might not be selected. |
| BOOLEAN (1 bit) | NUMERIC | 1. | |
| DATE | NUMERIC | DATE9. | - The supported date range is the year 1582 through the year 9999. |
| TIMESTAMP | NUMERIC | DATETIME21.2 | - The time zone is ignored.  
- The supported date range is the year 1582 through the year 9999.  
- Values are stored in ORC as an offset from the UNIX epoch. |
| DECIMAL | NUMERIC | w.d | - In the w.d format, w is the precision and d is the scale from the DECIMAL(p,s) data type.  
- When reading from ORC, converting a DECIMAL to a NUMERIC can result in loss of precision or scale for very large or very small values.  
- SAS cannot write a DECIMAL data type to ORC. A SAS NUMERIC that has a w.d informat is written to ORC as a DOUBLE. |
| LIST | | | Not supported. |
| MAP | | | Not supported. |
| STRUCT | | | Not supported. |
| UNION | | | Not supported. |
Restrictions on Data Types

- Compound data types are not supported: LIST, MAP, STRUCT, UNION. Tables that have these compound types are not usable in SAS. (The STRUCT root object type for table is supported. A STRUCT is the only supported schema.)

- See the notes in Table 5.1 for some limitations on columns that are converted to SAS data types. Many data types can be read by SAS but not written by SAS. For the data types that can be written by SAS, you must specify the indicated format.
Restrictions

Restrictions for SAS Features

The following behaviors differ from Base SAS. See also “Requirements” on page 3.

- Updating data is not currently supported. Therefore, renaming columns, inserting rows, and appending rows are not supported. For example, the APPEND procedure is supported only if the BASE= data set does not exist.

- Open-source file formats such as ORC do not store a sort indicator. Therefore, the sort information cannot be reported by the CONTENTS procedure (or the CONTENTS statement of PROC DATASETS). In addition, you might encounter SAS log messages stating that a sort indicator cannot be set. The log messages do not necessarily indicate an error. If the table is in sorted order, the processing completes without error. If the table is not in sorted order, then the processing stops and an error is written to the log.

- In the SORT procedure, the OVERWRITE option is not supported.

- See also some restrictions in “Naming Conventions” on page 5.

- See the language elements that are documented in this book for additional restrictions and interactions. Some restrictions could be removed in a future release.

The following SAS features are not supported:

- SAS indexes
- file or session encoding other than UTF-8
- special missing values that are created by using the MISSING statement
- encryption
- DS2 or FedSQL procedures
- features that are specific to the V9 engine, such as integrity constraints, audit trails, generation data sets, catalogs, and extended attributes
Restrictions for ORC File Features

- Compound data types are not supported. See Chapter 5, “Data Types,” on page 37.

- Hive data partitioning is not supported. (SAS does support tables that consist of multiple files. See “DIRECTORIES_AS_DATA= LIBNAME Statement Option” on page 22.)

- The only supported schema is a struct as the root object type, which contains a field for each of the top-level columns.

- If you use the COPY procedure (or the COPY statement of the DATASETS procedure) to copy an ORC table, be aware of the following interaction: If the output engine does not support ZLIB, the compression is changed to a compression that is supported by the output engine.

- SAS does not use ORC predicate pushdown for WHERE clause optimization.

- User-defined metadata is ignored. ORC tables can store user-defined metadata values that are created by certain software applications. Because the metadata is not a standard part of the schema definition or data storage, it is ignored by the engine.
Using Compression

In the ORC engine, ZLIB compression is the default. In other engines, compression is turned off by default.

For ORC, turning off compression almost always reduces performance, especially in Microsoft Azure Data Lake Storage Gen2.

SAS can also read compression types SNAPPY, LZO, LZ4, and ZSTD. However, these types are not supported for Write access.

- “COMPRESS= Data Set Option” on page 33
- “COMPRESS= LIBNAME Statement Option” on page 21

Setting the Stripe Size

In an ORC file, the data is grouped in stripes, which are separate and independent groups of row data. The size of the stripes can have an impact on performance.

To request a stripe size when you create an ORC table from SAS, use the DATA_BLOCK_SIZE= data set option on page 34 or the DATA_BLOCK_SIZE= LIBNAME statement option on page 22.

The actual stripe size in an ORC table does not exactly match the DATA_BLOCK_SIZE= value. The stripe size is also affected by factors such as compression. To determine the optimal value for DATA_BLOCK_SIZE=, performance testing is recommended.
Using the DROP= and KEEP= Data Set Options

You can use either the DROP= or KEEP= data set option to limit the columns that SAS reads from or writes to a table. When you read from a column-based file format such as ORC, limiting the columns is especially beneficial for performance.

See the following documentation:
- “DROP= Data Set Option” in SAS Data Set Options: Reference
- “KEEP= Data Set Option” in SAS Data Set Options: Reference

Using the FIRSTOBS= and OBS= Data Set Options

Use the SAS data set options FIRSTOBS= or OBS= (or both) to limit the number of rows that SAS reads from a table. Limiting the rows of a large table can be helpful while you are developing a query.

See the following documentation:
- “FIRSTOBS= Data Set Option” in SAS Data Set Options: Reference
- “OBS= Data Set Option” in SAS Data Set Options: Reference