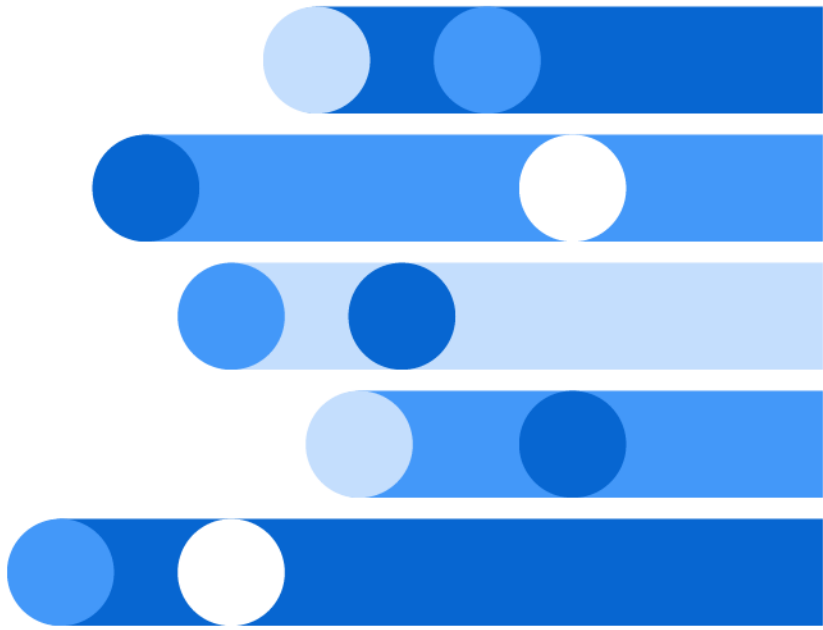
SAS® Cloud Analytic Services: User's Guide

2024.11 - 2026.09*



* This document might apply to additional versions of the software. Open this document in SAS Help Center and click on the version in the banner to see all available versions.

SAS® Documentation
September 15, 2026

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2024. *SAS® Cloud Analytic Services: User's Guide*. Cary, NC: SAS Institute Inc.

SAS® Cloud Analytic Services: User's Guide

Copyright © 2024, SAS Institute Inc., Cary, NC, USA

All Rights Reserved. Produced in the United States of America.

For a hard copy book: No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

For a web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

U.S. Government License Rights; Restricted Rights: The Software and its documentation is commercial computer software developed at private expense and is provided with RESTRICTED RIGHTS to the United States Government. Use, duplication, or disclosure of the Software by the United States Government is subject to the license terms of this Agreement pursuant to, as applicable, FAR 12.212, DFAR 227.7202-1(a), DFAR 227.7202-3(a), and DFAR 227.7202-4, and, to the extent required under U.S. federal law, the minimum restricted rights as set out in FAR 52.227-19 (DEC 2007). If FAR 52.227-19 is applicable, this provision serves as notice under clause (c) thereof and no other notice is required to be affixed to the Software or documentation. The Government's rights in Software and documentation shall be only those set forth in this Agreement.

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

September 2026

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

v_004-P1:casref

Contents

About This Book . . . . . *v*

Chapter 1 / Accessing Data with SAS Cloud Analytic Services **1**
Introduction 2
Terms to Be Familiar With 2
How Can I Access Data? 3
Common Tasks for Accessing Data 7
Upload a SAS Data Set with PROC CASUTIL 8
Upload a CSV with Simple Dates 10
Upload a Tab Delimited File with Complicated Dates 13
Upload a CSV File from a Server with PROC CASUTIL 17
Load a File from a Caslib with PROC CASUTIL 19
Load a Database Table 27
Save an In-Memory Table 29
Drop an In-Memory Table 30
Delete a File from a Caslib's Data Source 31
Data Compression 32

Chapter 2 / CAS LIBNAME Engine **37**
CAS LIBNAME Engine Overview 37
Getting Started 39
Data Types 41
Use of WHERE with the CAS LIBNAME Engine 52
Working with Temporary Columns 52
Variable Names and Data Set Names in CAS Engine 54
Troubleshooting CAS LIBNAME Engine Errors 54
Where to go from here 55

Chapter 3 / CASUTIL Procedure **57**
Overview: CASUTIL Procedure 58
Syntax: CASUTIL Procedure 59
Usage: CASUTIL Procedure 87
Results: CASUTIL Procedure 91
Examples: CASUTIL Procedure 94

Chapter 4 / MDSUMMARY Procedure **105**
Overview: MDSUMMARY Procedure 105
Syntax: MDSUMMARY Procedure 106
Usage: MDSUMMARY Procedure 109
Results: MDSUMMARY Procedure 109
Examples: MDSUMMARY Procedure 111

Chapter 5 / Platform Data Sources **125**
Data Redundancy 125
Dictionary 126

Chapter 6 / Platform File Types **153**
Dictionary 153

Chapter 7 / Data Connectors **167**
Working with SAS Data Connectors 168
Quick Reference for Data Connector Syntax 170
Where to Specify Data Connector Options 175
Using Wildcards for Pattern Matching 176
Using Data Transfer Modes with Data Connectors for DBMS Data Sources 177
Dictionary 179

Chapter 8 / Data Connector Options **383**
Dictionary 388

Chapter 9 / Data Types **609**
Overview of Data Types for CAS Table Columns 609
Summary of Data Types 610
Character Data 611
Numeric Data 613

Chapter 10 / CAS LIBNAME Options **615**
Dictionary 615

Chapter 11 / Data Set Options **631**
About Data Set Options for CAS 631

Chapter 12 / Macro Variables **633**
Dictionary 633

Chapter 13 / Session Options **645**
Setting Session Options 646
Session Options by Category 648
Dictionary 650

Chapter 14 / Statements **697**
Dictionary 697

About This Book

Audience

This book supports the use of SAS Cloud Analytic Services by participating SAS solutions. It explains key concepts and provides essential instructions. The emphasis is on server operation, data management, and security configuration.

1

Accessing Data with SAS Cloud Analytic Services

Introduction . . . 2

Terms to Be Familiar With . . . 2

How Can I Access Data? . . . 3
- Files on a PC . . . 3
- Files in a Shared File System . . . 4
- Tables in a Database Server . . . 6

Common Tasks for Accessing Data . . . 7

Upload a SAS Data Set with PROC CASUTIL . . . 8
- Example . . . 8
- Key Ideas . . . 9
- See Also . . . 9

Upload a CSV with Simple Dates . . . 10
- Example . . . 10
- Key Ideas . . . 12
- See Also . . . 12

Upload a Tab Delimited File with Complicated Dates . . . 13
- Example . . . 13
- Key Ideas . . . 17
- See Also . . . 17

Upload a CSV File from a Server with PROC CASUTIL . . . 17
- Example . . . 17
- Key Ideas . . . 19
- See Also . . . 19

Load a File from a Caslib with PROC CASUTIL . . . 19
- Prerequisites . . . 19
- Example . . . 20
- Results . . . 22
- Key Ideas . . . 26
- See Also . . . 27

Load a Database Table . . . 27
- Prerequisites . . . 27
- Example . . . 27

Key Ideas . . . 28
Save an In-Memory Table . . . **29**
Example . . . 29
Key Ideas . . . 30
Drop an In-Memory Table . . . **30**
Example . . . 30
Key Ideas . . . 31
Delete a File from a Caslib's Data Source . . . **31**
Example . . . 31
Key Ideas . . . 31
Data Compression . . . **32**
Overview of Data Compression . . . 32
Compressed Tables and the DATA Step . . . 32
Compressed Tables and the CASUTIL Procedure . . . 33
Performance Considerations . . . 34
Interactions . . . 34
See Also . . . 34

Introduction

Analytical procedures and some DATA step features use input data from in-memory tables on SAS Cloud Analytic Services only. This document takes you through common tasks for loading and accessing your input data with SAS Cloud Analytic Services. Here are the methods that these tasks use for loading and accessing:

- caslibs
- CASUTIL procedure
- DATA step

Terms to Be Familiar With

Before we begin, here are a few important terms:

active caslib
: your session must have a default location for server-side data access. This is the active caslib. The term *active caslib* is used rather than default caslib because the caslib that your session uses is modified as caslibs are added and dropped.

caslib
: the mechanism for accessing data with SAS Cloud Analytic Services. At its simplest, a caslib provides access to files in a data source, such as a database or file system directory, and to in-memory tables.

data connector
: a data connector is the software that is used with a caslib to read server-based data sources like databases and Hive. There are a few data connectors for file-based caslibs. These data connectors are used to control reading data files such as setting the file encoding.

file
: the source data that is in a caslib's data source. For a caslib that uses a path-based data source, this is natural. For a caslib that uses a database as a data source, the tables in the database are referred to as files.

session
: when you initially connect to SAS Cloud Analytic Services, your session is started on the server. Data access and communication is performed through the session. Your programs communicate with the session to request actions. Many sessions can operate concurrently, and actions execute serially within a session. In most cases, programmers start and use one session only.

table
: is used to refer to in-memory data. After a file (using the preceding definition) is loaded into the server, it is referred to as a table.

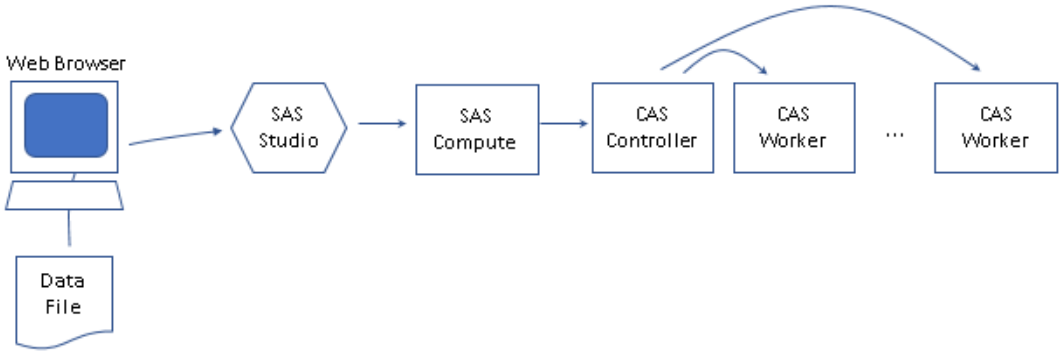
How Can I Access Data?

Files on a PC

Many times, the data to analyze is on your PC. You can use your browser with SAS Data and AI Studio (known as SAS Studio prior to 2026.06) to upload a file from your PC to CAS. The following figure represents this scenario.

For a distributed server, after the data is transferred to the controller, the controller distributes the data to workers. If you have a single-machine server, then the controller receives the data and can process the data.

Figure 1.1 Upload from a PC to CAS



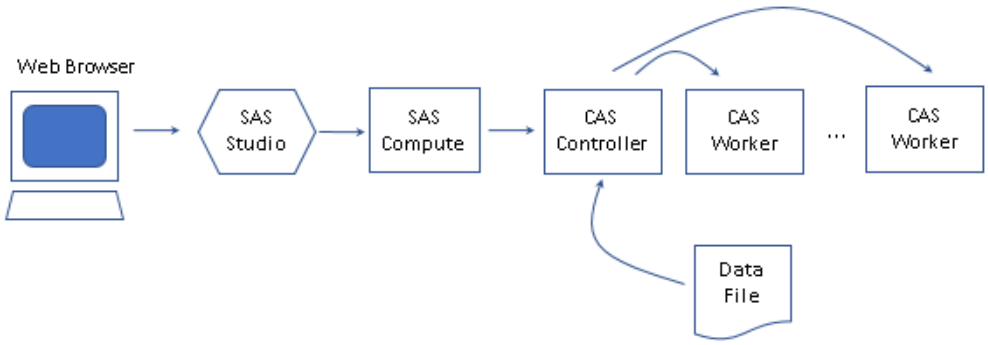
Method	Advantages	Disadvantages
If the file is small, less than 100MB, you can use SAS Data and AI Studio (known as SAS Studio prior to 2026.06) to upload the file.	■ For ad hoc analysis, uploading a small file is convenient.	■ This method relies on uploading each file manually and is tedious.
If the file is large or you have a large number of files to use, contact your IT administrator to move the files to a shared file system that CAS can access.	■ If you can move the file to a file system that CAS can access, you can add a caslib to use the directory and the server can load the data efficiently.	

Files in a Shared File System

For larger files or a greater number of files, it can be simpler to provide file system access between the server and the data.

The following figure represents the data movement for a Path caslib. The controller accesses the data and for distributed servers, the controller transfers rows to workers.

Figure 1.2 Path Caslib Data Access



If you have a distributed server and can provide file system access to the workers as well, then a DNFS caslib can be higher-performance choice. For a DNFS caslib, the workers access the file in parallel and this can improve data loading times.

Figure 1.3 DNFS Caslib Data Access

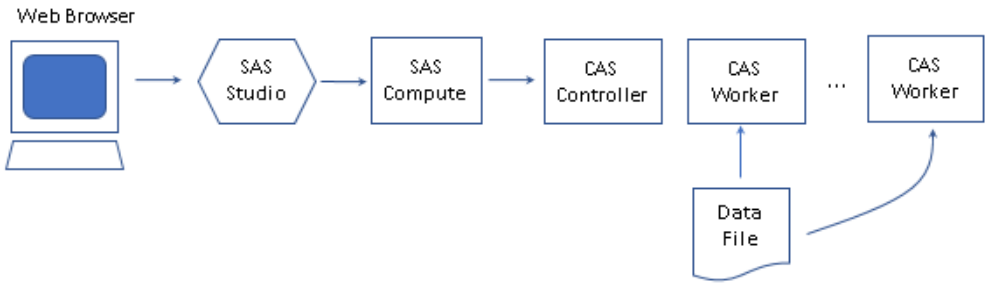


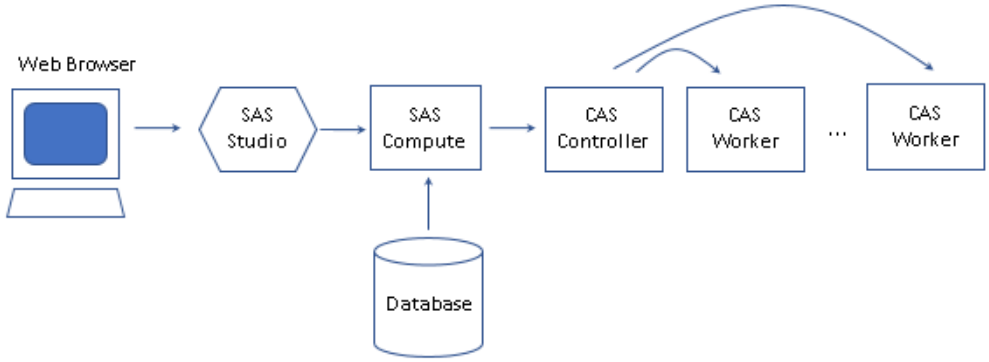
Table 1.1 On a Shared File System That CAS Can Access

Method	Advantages	Disadvantages
Assign a Path or DNFS caslib to access the directory.	■ The server accesses the data and avoids delays that are added by transferring the data through additional clients. ■ For a DNFS caslib, a distributed server can load the data in parallel from the directory.	■ If you need to load a CSV file that is not rigidly structured, CAS has limited ability to process record-by-record differences.
Assign a SAS libref to access the directory and then load from SAS to CAS.	■ To load a loosely structured CSV file, PROC IMPORT or a DATA step can perform sophisticated data type conversion and data manipulation.	■ Unless data manipulation is required, SAS transfers the data serially to CAS and can be slow due to network transfer speeds.

Tables in a Database Server

SAS/ACCESS engines enable traditional data access to databases. After SAS reads the data, it can modify it if that is required and then transfer the data to CAS.

Figure 1.4 *Access Data with a SAS/ACCESS Engine*



SAS data connectors are used with CAS to enable access between CAS and the database server. By default, data is transferred between the database and the controller. The controller can distribute rows to workers afterward. This scenario is shown with the solid lines.

SAS data connect accelerators enable parallel data transfer between CAS workers and either a single database server or a multi-node database server. This scenario is shown with the dashed lines.

Figure 1.5 *Access Data with a Data Connector or Data Connect Accelerator*

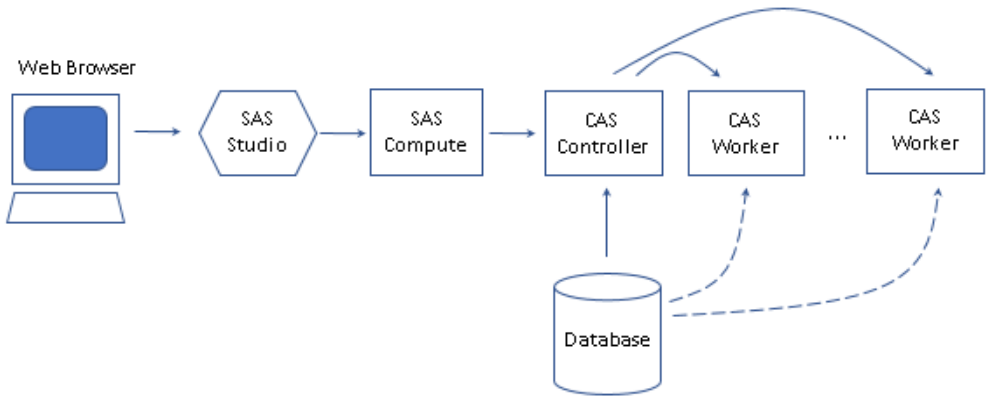


Table 1.2 *In a Database Server*

Method	Advantages	Disadvantages
If you are familiar with SAS LIBNAME statements, then you can assign a libref to the	■ Familiar for SAS programmers.	■ Transfer times increase with larger tables.

Method	Advantages	Disadvantages
database and then transfer the data from the database to the SAS session and then to CAS.	■ Performance is satisfactory for small tables.	
Assign a caslib to transfer the data from the database to the CAS controller with the loadTable action.	■ This method avoids the data transfer to the SAS session and reduces transfer times.	■ For very large tables, data transfer to the controller can be a bottleneck. ■ Less ability to manipulate data than the DATA step. You can specify only data types and labels.
Assign a caslib and use a data connect accelerator to transfer data between the database and multiple workers on a distributed CAS server.	■ Parallel data transfer offers the best performance.	■ Applies to distributed CAS servers only.

Common Tasks for Accessing Data

Table 1.3 *Common Tasks for Accessing Data*

Task	Sample Syntax

Upload a SAS data set.
TIP This is a good learning step if you are familiar with SAS and want to learn how SAS libraries, data sets, and SAS Cloud Analytic Services work together.
	`proc casutil;`
	`  load data=`*`libref.member-name`*
	`  casout="`*`table-name`*`";`
	`run;`

Upload a data file from SAS.	`proc casutil;`
	`  load file="`*`/path/to/file.suffix`*`"`
	`  casout="`*`table-name`*`";`
	`run;`

List caslibs. This shows you the server-side data sources that SAS Cloud Analytic Services can access.	`caslib _all_ list;`

Task	Sample Syntax
Add a file-based caslib. **TIP** Remember that the specified PATH= must be accessible from the host for the SAS Cloud Analytic Services controller.	`caslib data datasource=(srctype="path")` `  path="/data01";`

Determine the data files in a caslib that the server can access.	`proc casutil;`
	`  list files incaslib="name";`
	`run;`

Load a data file that SAS Cloud Analytic Services can access.	`proc casutil;`
	`  load casdata="file-name.suffix"`
	`  casout="table-name";`
	`run;`

Load a table from database.	`proc casutil;`
	`  load casdata="database-table-name"`
	`  casout="table-name";`
	`run;`

List in-memory tables.	`proc casutil;`
	`  list tables incaslib="name";`
	`run;`

Assign a CAS engine libref and bind it to a caslib. This is how you access tables with a SAS procedure or a DATA step.	`libname mycas cas caslib="name";`

Upload a SAS Data Set with PROC CASUTIL

Example

This example shows how to upload a SAS data set from a SAS library on the SAS Compute Server to CAS. This approach is appropriate for smaller tables and ad hoc data analysis.

```
cas casauto
sessopts=(caslib="public");                                    /* 1 */

proc casutil;
```

```
   load data=sashelp.iris
promote;                                        /* 2 */

   contents casdata="iris";

run;
```

1 Set the active caslib for the session to Public. This caslib is usually enabled at deployment time and is commonly used for ad hoc data access.

2 The DATA= argument indicates that the data from the SAS data set is transferred from the SAS Compute Server to SAS Cloud Analytic Services. Replace the Sashelp.Iris value with a libref and SAS data set that you want to use. The PROMOTE option makes the Iris table a global-scope table and available to other sessions that use the Public caslib. Use the CONTENTS statement to ensure that the table includes the column names and data types that you expect.

Output 1.1 *Results of the CONTENTS Statement*

Table Information for Caslib Public

Table Name	Label	Number of Rows	Number of Columns	Indexed Columns	NLS encoding	Created	Last Modified	Promoted Table	Repeated Table	View	Compressed
IRIS	Fisher's Iris Data (1936)	150	5	0	utf-8	2020-04-02T18:23:26+00:00	2020-04-02T18:23:26+00:00	Yes	No	No	No

Detail Information for iris in Caslib Public.

Node	Number of Blocks	Active Blocks	Rows	Data size	Variable Data size	Blocks Mapped	Memory Mapped	Blocks Unmapped	Memory Unmapped	Blocks Allocated	Memory Allocated	Index Size	Compressed Size	Compression Ratio
ALL	1	1	150	7200	0	1	7376	0	0	0	0	0	0	0

Column Information for IRIS in Caslib Public

Column	Label	Type	Length	Formatted Length	Format Width	Format Decimal
Species	Iris Species	char	10	10	0	0
SepalLength	Sepal Length (mm)	double	8	12	0	0
SepalWidth	Sepal Width (mm)	double	8	12	0	0
PetalLength	Petal Length (mm)	double	8	12	0	0
PetalWidth	Petal Width (mm)	double	8	12	0	0

Key Ideas

- The LOAD DATA= statement in the CASUTIL procedure is used for accessing SAS data sets.
- The CONTENTS statement is used to display information such as column names and data types.

See Also

- Chapter 3, “CASUTIL Procedure,” on page 57
- “CASLIB Statement” on page 745

Upload a CSV with Simple Dates

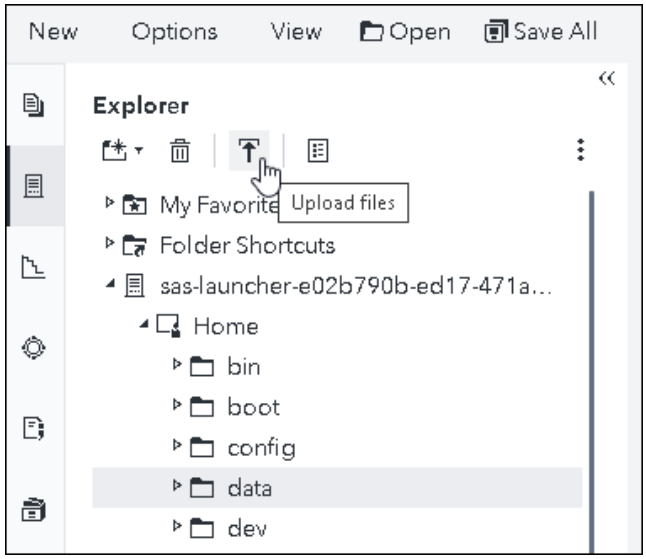
This example shows how to upload a CSV file with your browser and SAS Data and AI Studio (known as SAS Studio prior to 2026.06) to a persistent storage location. After the file is transferred to the persistent storage that the SAS Compute Server can access, SAS Data and AI Studio is used to transfer the entire file into CAS where CAS processes the file.

The CSV file that is used in the example, **`waterflow.csv`**, is available for download from https://support.sas.com/documentation/onlinedoc/viya/examples.htm.

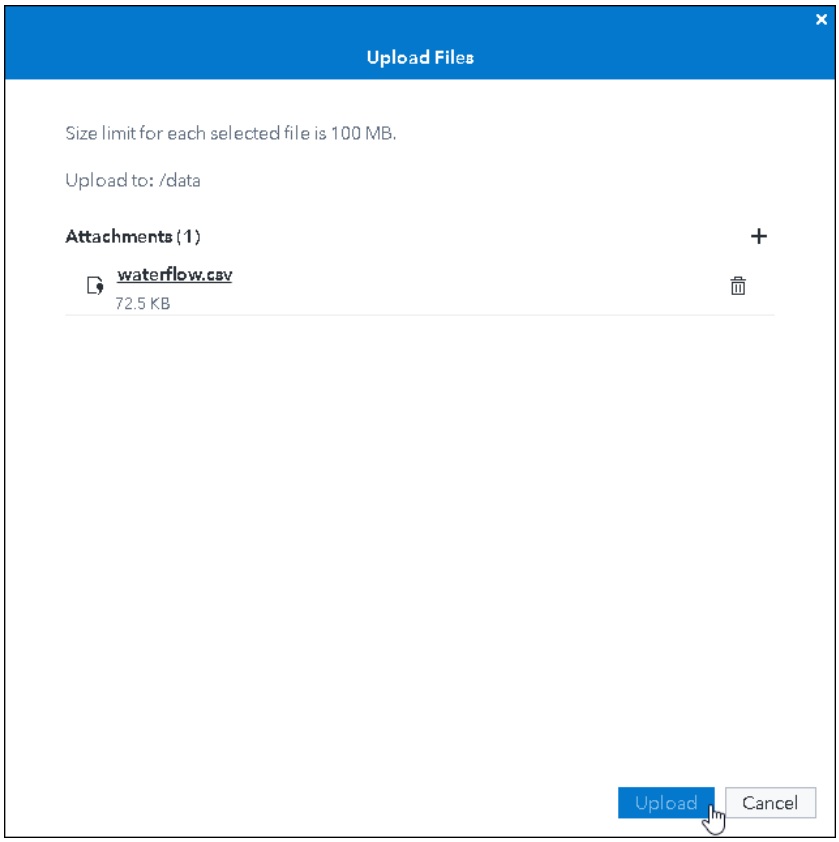
Example

The first step is to upload the file from your PC to a persistent storage location. Follow these steps in SAS Data and AI Studio (known as SAS Studio prior to 2026.06):

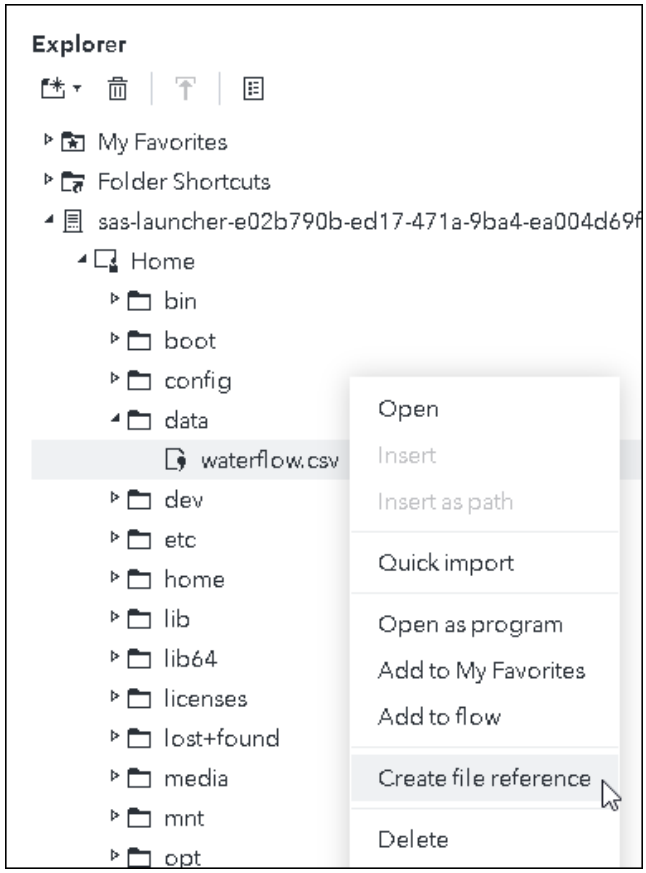
1 In the Explorer section of the navigation pane, navigate to a location such as **/ `data`** under the sas-launcher icon and then click ⤒.



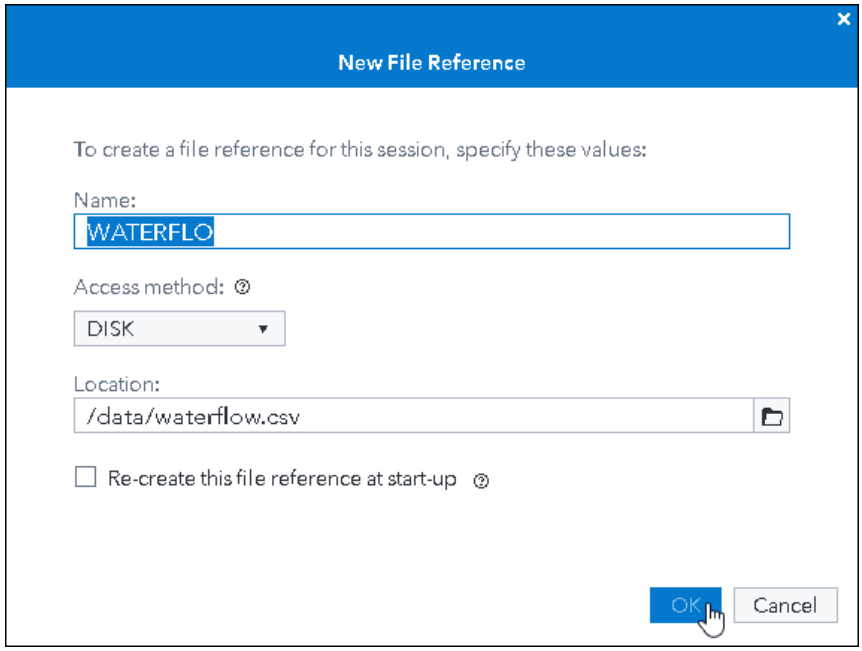
2 On the Upload Files window, click + and then navigate to the file. Afterward, click **Upload**.



3 Right-click on the uploaded file and select **Create file reference**.



4 On the New File Reference window, you can change the file reference name or select the check box so that the file reference is re-created.



5 Run the following code.

```
proc casutil;
  load file=waterflo casout="waterflow" replace
    importoptions=(
      filetype="csv"
      vars={
         dttm={informat="B8601DT19." format="B8601DT."}
      }
    );
run;
```

Key Ideas

- SAS Data and AI Studio (known as SAS Studio prior to 2026.06) can transfer small files from your PC to a location that the SAS Compute Server can access.
- The LOAD FILE= statement in the CASUTIL procedure can transfer a file as-is to CAS. After CAS receives the file, it can import the data from the file and create an in-memory table.

See Also

- "Introduction to SAS Data and AI Studio" in *SAS Data and AI Studio: User's Guide*

■ Chapter 3, “CASUTIL Procedure,” on page 57

Upload a Tab Delimited File with Complicated Dates

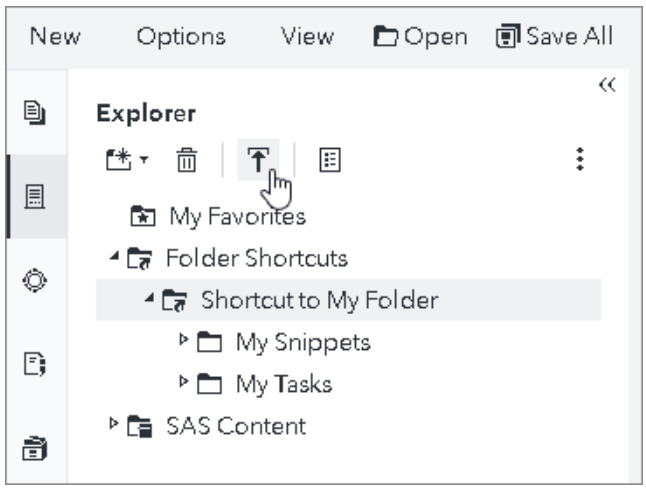
Example

The following sample of an energy usage file is tab delimited and is difficult to parse in CAS. A simpler strategy is to use a DATA step in SAS to pre-process the delimited file and then transfer the data to CAS.

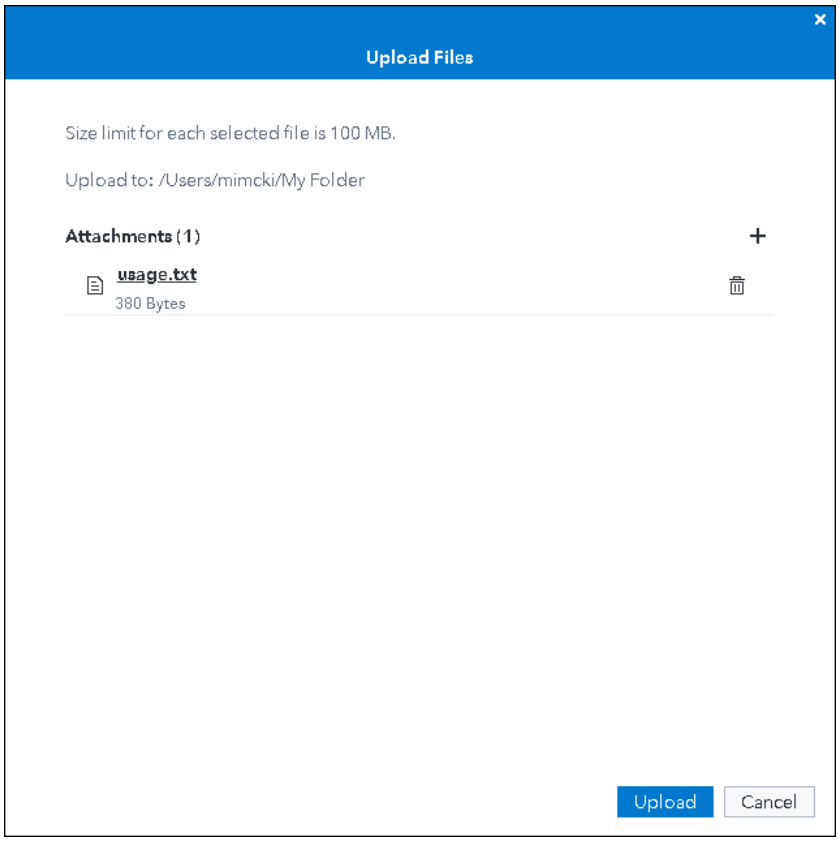
```
account    high    low    dttm    startdt    enddt    kwh
75185544    71    62    11/03/2013 04:11 AM    Fri 11/01/2013    Sat 11/02/2013    31
75185544    68    48    11/04/2013 04:05 AM    Sat 11/02/2013    Sun 11/03/2013    42
75185544    59    38    11/05/2013 04:11 AM    Sun 11/03/2013    Mon 11/04/2013    32
75185544    52    32    11/06/2013 04:11 AM    Mon 11/04/2013    Tue 11/05/2013    30
75185544    56    35    11/07/2013 04:07 AM    Tue 11/05/2013    Wed 11/06/2013    35
```

The first step is to upload the file from your PC to a location that the SAS Compute Server can access. In the following example, the file is uploaded to **My Folder**, a location in the files service and not an operating system directory. Follow these steps in SAS Data and AI Studio (known as SAS Studio prior to 2026.06):

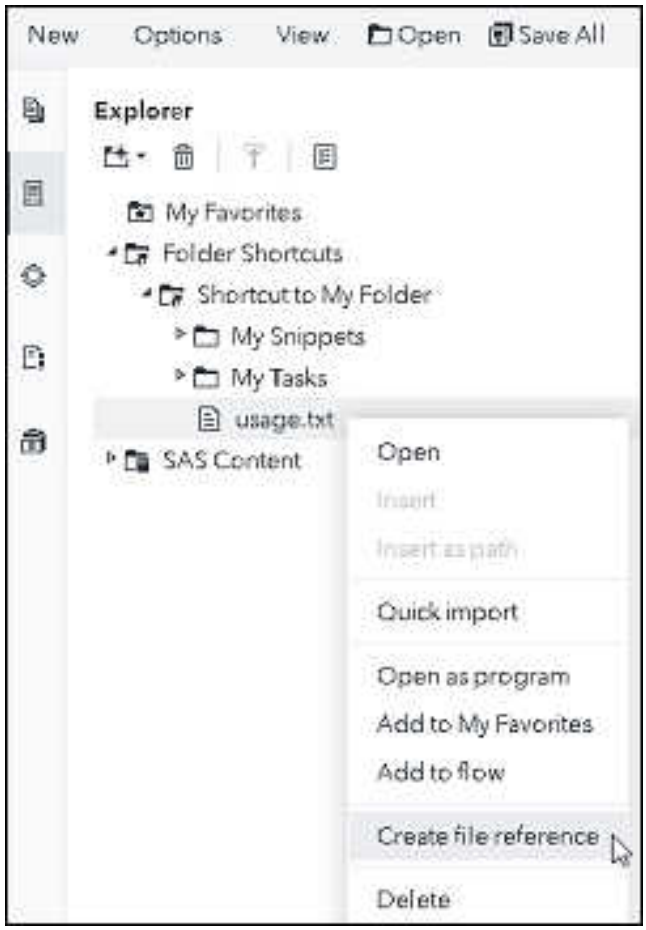
1 In the Explorer section of the navigation pane, navigate to a location such as **Shortcut to My Folder** and then click ⤒.



2 On the Upload Files window, click + and then navigate to the file. Afterward, click **Upload**.

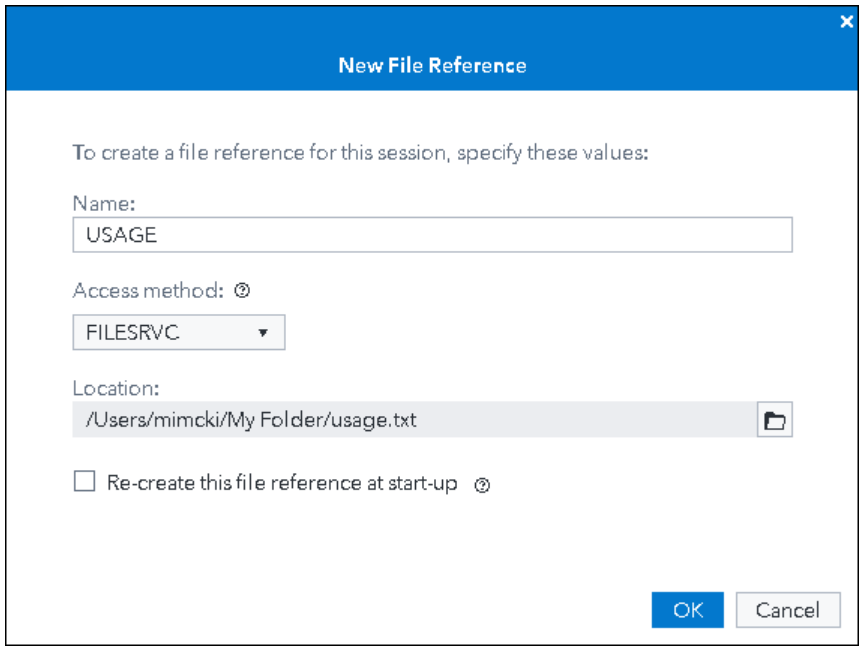


3 Right-click on the uploaded file and select **Create file reference**.



Note: The graphic shows that the Usage.txt file is stored in SAS content, in **My Folder**. Uploading small files to SAS content is acceptable if you plan to use a DATA step to read the file. This code example uses a DATA step.

4 On the New File Reference window, you can change the file reference name or select the check box so that the file reference is re-created.



5 Run the following code.

There are several possible ways to process the data. The following example is one way.

If you changed the name of the file reference from Usage to something else, then replace the file reference in the INFILE statement.

```
cas casauto;

data work.prep;
  infile usage delimiter='09'x firstobs=2;
  informat account high low kwh 8.;
  informat dttm anydtdtm19.;
  informat startdt enddt anydtdte14.;
  format dttm datetime19.;
  format startdt enddt date10.;
  input account    high low dttm sdow $char3. startdt edow $char3.
enddt kwh;
run;

proc contents data=work.prep;
run;

proc print data=work.prep(obs=2);
run;

proc casutil;
  load data=work.prep casout="usage" replace;
run;

proc cas;
  simple.summary /
    table={
      name="usage"
    }
```

```
    inputs={"high", "low", "kwh"};
run;
```

Key Ideas

- SAS Data and AI Studio (known as SAS Studio prior to 2026.06) can transfer small files from your PC to a location that the SAS Compute Server can access.
- The INFILE statement can read a file from SAS content in a DATA step.
- Informats and the INPUT statement excel at reading data from delimited files and interpreting the data into the correct numeric or character data type.

See Also

- "Introduction to SAS Data and AI Studio" in *SAS Data and AI Studio: User's Guide*
- "INFILE Statement" in *SAS DATA Step Statements: Reference*
- "INPUT Statement" in *SAS DATA Step Statements: Reference*

Upload a CSV File from a Server with PROC CASUTIL

Example

This example shows how to use the HTTP procedure with the SAS Compute Server. The procedure transfers a CSV file to a temporary location and then uploads the CSV file from the SAS Compute Server to CAS. This approach is appropriate for smaller tables and ad hoc data analysis.

```
cas casauto
sessopts=(caslib="public");                                    /* 1 */

%let data='http://support.sas.com/documentation/onlinedoc/viya/
exampledatasets/classfit.csv';
filename t temp;
```

```
proc http method='get' url=&data.
out=t;                                    /* 2 */
run;

%let temppath = %sysfunc(quote(%sysfunc(pathname(t))));

proc casutil;
  load file=&temppath.
casout="classfit"                                  /* 3 */
    importoptions=(filetype="csv") promote;
run;
  contents casdata="classfit";
run;
```

1 Set the active caslib for the session to Public. This caslib is usually enabled at deployment time and is commonly used for ad hoc data access.

2 The HTTP procedure reads the CSV file from the URL and stores it in a temporary file.

3 The CASUTIL procedure transfers the data from the SAS Compute Server to SAS Cloud Analytic Services.

The FILE= argument is used with the file reference. You must specify the CASOUT= argument. Specify the IMPORTOPTIONS= and FILETYPE= arguments to ensure that the file is read correctly by SAS Cloud Analytic Services. The PROMOTE option makes the Classfit table a global-scope table and available to other sessions that use the Public caslib. Use the CONTENTS statement to ensure that the table includes the column names and data types that you expect.

Output 1.2 *Results of the CONTENTS Statement*

Table Information for Caslib Public

Table Name	Number of Rows	Number of Columns	Indexed Columns	NLS encoding	Created	Last Modified	Promoted Table	Repeated Table	View	Compressed
CLASSFIT	19	10	0	utf-8	2020-04-02T18:11:12+00:00	2020-04-02T18:11:12+00:00	Yes	No	No	No

Detail Information for classfit in Caslib Public.

Node	Number of Blocks	Active Blocks	Rows	Data size	Variable Data size	Blocks Mapped	Memory Mapped	Blocks Unmapped	Memory Unmapped	Blocks Allocated	Memory Allocated	Index Size	Compressed Size	Compression Ratio
ALL	8	8	19	1946	122	8	2936	0	0	0	0	0	0	0

Column Information for CLASSFIT in Caslib Public

Column	Type	Length	Formatted Length	Format Width	Format Decimal
Name	varchar	7	7	0	0
Sex	varchar	1	1	0	0
Age	double	8	12	0	0
Height	double	8	12	0	0
Weight	double	8	12	0	0
predict	double	8	12	0	0
lowermean	double	8	12	0	0
uppermean	double	8	12	0	0
lower	double	8	12	0	0
upper	double	8	12	0	0

Key Ideas

- The LOAD statement in the CASUTIL procedure can transfer data from the compute server to CAS.
- The LOAD FILE= statement in the CASUTIL procedure is used for accessing files that are accessible by the compute server.
- The CONTENTS statement is used to display information such as column names and data types.

See Also

- Chapter 3, "CASUTIL Procedure," on page 57
- "CASLIB Statement" on page 745

Load a File from a Caslib with PROC CASUTIL

Prerequisites

The following example assumes the following:

- You can create a small CSV file in the file system that is associated with the directory for your personal caslib.
- The Microsoft Excel file used in this example has historical consumer price index (CPI) data for food from http://www.ers.usda.gov/data-products/food-price-outlook.

Example

This example shows how to access two files that CAS can access in a file system and then load the data into CAS:

- a CSV file that describes the performance of a toy catapult. The first line of the file does not contain column names. The example shows how to specify names.
- a Microsoft Excel file.

For the CSV file, a description for the data is shown in the example. The values are as follows:

```
5,10,10,11,10,11,3
5.5,16,3,16,1,15,6
6,23,0,18,7,20,5
6.5,23,3,28,6,26,0
7,27,3,25,10,23,1
```

The program is as follows:

```
cas casauto
sessopts=(caslib="casuser");                                    /* 1 */

libname mycas cas
caslib="casuser";                                          /* 2 */

/* first, load the data from the CSV file */
proc casutil incaslib="casuser" outcaslib="casuser";

  contents
casdata="catapult.csv";                                     /* 3 */

  load casdata="catapult.csv" casout="catapultraw"

importoptions=(filetype="csv"                                    /* 4
*/
          encoding="latin1"
          getnames="false"
          vars=(
            (name="turns",     label="Number of turns",
type="double"),
            (name="first_ft",  label="Feet for first try",
type="double"),
            (name="first_in",  label="Inches for first try",
type="double"),
            (name="second_ft", label="Feet for second try",
type="double"),
            (name="second_in", label="Inches for second try",
type="double"),
            (name="third_ft",  label="Feet for third try",
type="double"),
```

```
            (name="third_in",  label="Inches for third try",
type="double")
          )
      ) replace;

    save casdata="catapultraw"
replace;                                  /* 5 */
    contents casdata="catapultraw.sashdat";
quit;

data mycas.catapult (promote=yes) /
sessref=casauto;                    /* 6 */
   set mycas.catapultraw;
   first  = 12 * first_ft  + first_in;
   second = 12 * second_ft + second_in;
   third  = 12 * third_ft  + third_in;
run;

proc casutil incaslib="casuser";
   contents
casdata="catapult";                                          /* 7 */
   droptable
casdata="catapultraw";                                      /* 8 */
quit;

/* simple scatter plot */
proc sgplot
data=mycas.catapult;                                         /* 9 */
     scatter x=turns y=first;
     scatter x=turns y=second;
     scatter x=turns y=third;

     /*--X Axis--*/
     xaxis grid label="Number of turns";

     /*--Y Axis--*/
     yaxis grid label="Distance, in inches";;
run;

/* second, load the Excel file */
proc casutil incaslib="casuser" outcaslib="casuser";
   list
files;                                                              /* 10
*/
   contents casdata="historicalcpi.xls";

   load casdata="historicalcpi.xls"
casout="historicalcpi"              /* 11 */
        label="Historical CPI data, 1974 through 2014; updated
3/3/2015";

   save casdata="historicalcpi" replace;
quit;
```

1 The SESSOPTS= option is used with the CASLIB= session option to ensure that the Casuser personal caslib is set as the active caslib.

2 The CAS engine LIBNAME statement assigns the Mycas libref and binds it to the Casuser caslib.

3 The CONTENTS statement shows the file information and column information for the CSV file. See Figure 1.6 on page 23.

4 The CASDATA= argument indicates that the file is read from the caslib's data source. The IMPORTOPTIONS= specify how to read the file.

5 The SAVE statement makes a copy of the imported data as a SASHDAT file. This is part of the data life cycle. If the file is imported correctly, then subsequent analyses of the data can begin from the SASHDAT file. The CONTENTS statement shows that the column names and labels are applied. See Figure 1.7 on page 23.

6 The DATA step is used to combine each set of foot and inch measures into a single column. The PROMOTE= option is used to make the table available to other sessions that you start. The SESSREF= option is used to ensure that the DATA step runs in CAS.

7 The last CONTENTS statement is used to display the table information, table details, and column information for the in-memory table. See Figure 1.8 on page 24.

8 The DROPTABLE statement is used to free the memory resources that are used for the data from the CSV file. The copy of the data that was made with the SAVE statement is not deleted, only the in-memory resources are freed.

9 The SGPLOT procedure uses the Mycas CAS engine libref. When Mycas was assigned at the start of the program, the CASLIB= option bound it to the Casuser caslib. This ensures that the libref always accesses tables in that caslib.

10 The LIST FILES statement is used to list the files in the caslib's data source. In this case, the personal caslib, Casuser, uses the OS file system. See Figure 1.9 on page 24.

11 The LOAD CASDATA= argument specifies the Historicalcpi.xls file. The LABEL= option is used to specify a description of the data.

Results

The following display shows the results of the CONTENTS statement. Notice that the anticipated column names match the first line of the CSV file. This is corrected in the subsequent LOAD CASDATA= statement when the GETNAMES= option is set to false.

Figure 1.6 *CONTENTS Statement Results for the CSV File*

The CASUTIL Procedure

File Information for catapult.csv in caslib CASUSER( ).					
Name	**Permission**	**Owner**	**Group**	**File Size**	**Last Modified**
catapult.csv	-rw-r--r--			0.1KB	01Feb2016:14:50:35

Column Information for catapult.csv in Caslib CASUSER( )			
Column	**Type**	**Length**	**Formatted Length**
5	double	8	12
10	double	8	12
10	double	8	12
11	double	8	12
10	double	8	12
11	double	8	12
3	double	8	12

The following display applies to the second CONTENTS statement, after the LOAD CASDATA= statement that include the IMPORTOPTIONS= settings. Notice that the column names and labels are applied.

Figure 1.7 *CONTENTS Statement Results for the SASHDAT File*

File Information for catapultraw.sashdat in caslib CASUSER( ).					
Name	**Permission**	**Owner**	**Group**	**File Size**	**Last Modified**
catapultraw.sashdat	-rwxr-xr-x			24.2KB	01Feb2016:19:27:28

Column Information for catapultraw.sashdat in Caslib CASUSER( )				
Column	**Label**	**Type**	**Length**	**Formatted Length**
turns	Number of turns	double	8	12
first_ft	Feet for first try	double	8	12
first_in	Inches for first try	double	8	12
second_ft	Feet for second try	double	8	12
second_in	Inches for second try	double	8	12
third_ft	Feet for third try	double	8	12
third_in	Inches for third try	double	8	12

Figure 1.8 *CONTENTS Statement Results for the In-Memory Table*

Table Information for Caslib CASUSER(...)

Table Name	Number of Rows	Number of Columns	NLS encoding	Created	Last Modified	Promoted Table	Duplicated Rows	View	Source Name	Compressed
CATAPULT	5	10	utf-8	01Feb2016:19:28:25	01Feb2016:19:27:56	Yes	No	No	_T_1AAD1589_7FF56342BBA8	No

Detail Information for catapult in Caslib CASUSER(...).

Node	Number of Blocks	Active Blocks	Rows	Fixed Data size	Variable Data size	Blocks Mapped	Memory Mapped	Blocks Unmapped	Memory Unmapped	Blocks Allocated	Memory Allocated
ALL	5	5	5	400	0	0	0	5	400	0	0

Column Information for CATAPULT in Caslib CASUSER(...)

Column	Label	Type	Length	Formatted Length
turns	Number of turns	double	8	12
first_ft	Feet for first try	double	8	12
first_in	Inches for first try	double	8	12
second_ft	Feet for second try	double	8	12
second_in	Inches for second try	double	8	12
third_ft	Feet for third try	double	8	12
third_in	Inches for third try	double	8	12
first		double	8	12
second		double	8	12
third		double	8	12

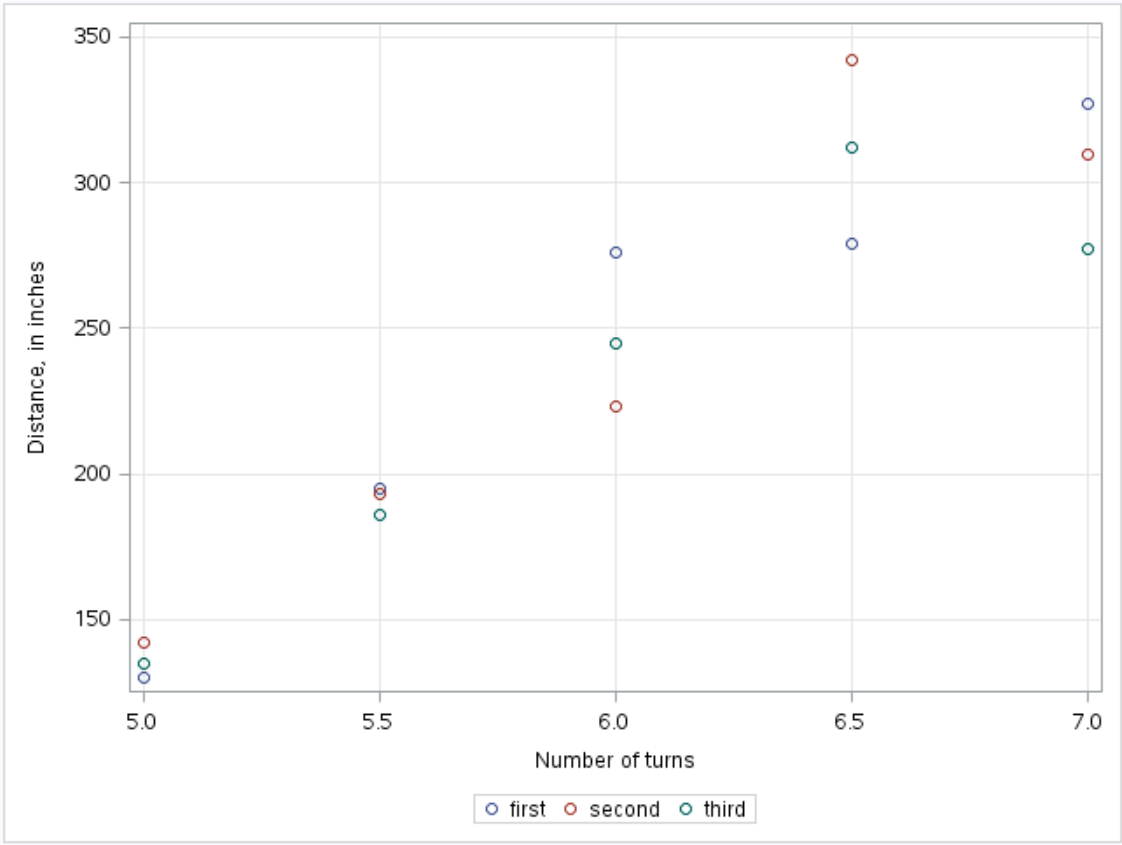
Figure 1.9 *LIST FILES Statement Results for a Path-Based Caslib*

The CASUTIL Procedure

File Information for root of caslib CASUSER(...).

Name	Permission	Owner	Group	File Size	Last Modified
catapult.csv	-rw-r--r--	...	...	0.1KB	01Feb2016:14:50:35
historicalcpi.xls	-rwxr-xr-x	...	...	56.5KB	02Feb2016:19:08:19
mycas.distinct.sashdat	-rwxr-xr-x	...	...	7.2KB	12Nov2015:16:48:12
cars_large_part.sashdat	-rwxr-xr-x	...	...	20.4MB	26Jan2016:12:44:45

Figure 1.10 *Scatter Plot Results for the Catapult Table*



The following display shows the results of the CONTENTS statement for the Historicalcpi.xls file. By default, the column names are read from a file.

Figure 1.11 CONTENTS Statement Results for the XLS File

The CASUTIL Procedure

File Information for historicalcpi.xls in caslib CASUSER([redacted]).					
Name	Permission	Owner	Group	File Size	Last Modified
historicalcpi.xls	-rwxr-xr-x	[redacted]	[redacted]	56.5KB	02Feb2016:19:08:19

Column Information for historicalcpi.xls in Caslib CASUSER([redacted])				
Column	Type	Length	Formatted Length	Format
Consumer price indexes historical data, 1974 through 2014	varchar	0	0	$
B	varchar	0	0	$
C	varchar	0	0	$
D	varchar	0	0	$
E	varchar	0	0	$
F				
	varchar	0	0	$
AO	varchar	0	0	$
AP	varchar	0	0	$
AQ	varchar	0	0	$
AR	varchar	0	0	$
AS	varchar	0	0	$
AT	varchar	0	0	$

Key Ideas

- Reading files from a caslib's data source is the most efficient way to access data. One key to recognizing that data is read from a caslib's data source is the presence of the CASDATA= argument.
- The CONTENTS statement in the CASUTIL procedure can display information for files, tables, and columns. To view information about a file, specify the file name, including the suffix in the CASDATA= option. After the table is loaded into memory, you drop the suffix or use the table name that you specified in the CASOUT= option.
- You can specify IMPORTOPTIONS= to describe how to load the data. For delimited files, the common options are to specify the file encoding and whether to get column names from the first line of the file.
- After the data for a file is imported, save a copy as a SASHDAT file.

See Also

- Chapter 3, "CASUTIL Procedure," on page 57
- "CASLIB Statement" on page 745
- *SAS Cloud Analytic Services: DATA Step Programming*

Load a Database Table

Prerequisites

The following example assumes the following:

- You are granted access to data in the data source.
- You know the connection information such as host, port, and so on.
- Your SAS Cloud Analytic Services installation is licensed and configured to use the client software for the data source vendor that you want to access. For installation-time configuration information, see *SAS Viya Platform: Deployment Guide*.

Example

Loading tables into the server from a caslib's data source is the most efficient way to load data. In this example, a table is read from Oracle and the in-memory table is kept in the same caslib.

```
caslib oralib
datasource=(                                                    /* 1 */
    srctype="oracle",
    uid="DBUSER",
    pwd="secret",
    path="//dbserver.example.com:1521/dbname",
    schema="DBUSER"
    );


proc casutil;
```

```
   list
files;                                                    /* 2 */

   droptable casdata="sales"
quiet;                                  /* 3 */
   contents
casdata="sales";                                          /* 4 */

   load casdata="sales" casout="sales"
promote                         /* 5 */
       label="Fact table for User-to-Item Analysis"
       varlist=(
            (name="USERID"  label="User ID"),
            (name="ITEMID"  label="Item ID")
       );

   contents
casdata="sales";                                          /* 6 */
quit;

run;
```

1 Add a caslib that uses Oracle as the data source. Oralib becomes the active caslib for the session and the subsequent programming statements use it for input and output.

2 The LIST FILES statement displays the tables that are available in the Oracle database.

3 The DROPTABLE statement includes the QUIET option. Running this statement is useful on repeated runs because it ensures that no table named Sales can be in-memory to interfere with the subsequent LOAD CASDATA= statement.

4 Because the first CONTENTS statement follows the DROPTABLE statement, this ensures that the table information and column information from Oracle are read.

5 The CASDATA= argument in the LOAD statement indicates that the Sales table is read from the caslib's data source (Oracle) into SAS Cloud Analytic Services. Options are specified to add labels to the table and columns.

6 Because the last CONTENTS statement follows the LOAD statement, table information and column information is displayed for the in-memory copy of the Sales table that was read from Oracle.

Key Ideas

- The CASLIB on page 745 statement adds a server-side data source to SAS Cloud Analytic Services.
- In this example, the active caslib is Oralib. Remember that when you add a caslib, by default, it becomes the active caslib.

- For information about data source connection parameters, see "Data Connectors" on page 167.

Save an In-Memory Table

Example

This example demonstrates the following:

- saving a table to caslib named Public. The table is saved as a SASHDAT file.
- saving a table from a caslib that uses Oracle to a path-based caslib named Public. The table is saved as a SASHDAT file.

```
proc casutil incaslib="public" outcaslib="public";
   load casdata="customers.sashdat" casout="customers";
run;

/* From some other application, or a DATA step, the */
/* Customers table is modified with a change that   */
/* is important to save.                            */
proc casutil  incaslib="public" outcaslib="public";
   save casdata="customers"
replace;                                  /* 1 */
run;

caslib orsales datasource=(
   srctype="oracle"
   username="DBUSER"
   password="secret"
   path="//dbserver.example.com:1521/sales"
   schema="DBUSER"
   );

proc casutil incaslib="orsales";
   load casdata="sales"
casout="sales";                                  /* 2 */

   save casdata="sales" outcaslib="public";
run;
```

1 The SAVE statement uses the Public caslib from the OUTCASLIB= procedure statement option. The next time a LOAD statement is used with the Customers table, the table includes the changes. You can use the CASOUT= option to specify an alternative file name other than the default, customers.sashdat.

2 The LOAD statement reads the table named Sales from the Orsales caslib that uses Oracle Database as a data source. To save a copy of the table in a different caslib, the SAVE statement uses the OUTCASLIB= option to specify the Public caslib that is a path-based data source.

Key Ideas

- You can save in-memory tables as SASHDAT files or CSV files in path-based caslibs only. If you need to save a table from a caslib that is not path-based, then you can specify the OUTCASLIB= option.
- For information about data source connection parameters to use in the CASLIB statement, see "Data Connectors" on page 167.

Drop an In-Memory Table

Example

When you drop an in-memory table, only the in-memory table is affected. If the table was loaded from a caslib's data source, the table in the data source is unaffected.

```
proc casutil incaslib="public" outcaslib="public";
   load casdata="sales.sashdat"
casout="sales";                          /* 1 */
run;

proc casutil incaslib="public";
   droptable
casdata="sales";                                    /* 2 */
run;
```

1 The LOAD statement loads a file named Sales.sashdat from the Public caslib.

2 The DROPTABLE statement drops the in-memory table.

Key Ideas

- If you drop a session-scope table, then only the session that loaded the table is affected.
- If you drop a global-scope table, then the table might be accessed from multiple sessions. The table is dropped after any actions that access the table are complete.
- Be aware that dropping a global-scope table can affect other sessions if the actions that are run by other sessions expect the table to be in memory.

Delete a File from a Caslib's Data Source

Example

When you delete a file from a data source, it does not affect an in-memory copy. The term "file" refers to a file in a caslib with a path-based data source or a table in a caslib with a server-based data source.

The following example demonstrates deleting a file named Sales from the data source associated with the Public caslib.

```
proc casutil;
   deletesource casdata="sales.sashdat" incaslib="public";
run;
```

Key Ideas

- Include the file name suffix in the CASDATA= argument.

Data Compression

Overview of Data Compression

SAS Cloud Analytic Services supports and performs all compression for in-memory tables. When you transfer a table to the server and request compression, rows are sent to the server as is and the server compresses them.

All data in a row, both character and numeric variables, are compressed. Every row in a table is compressed. The server does not support some rows in compressed form and others as uncompressed.

For matrices of computed doubles (those with many decimal places), compression might not reduce the storage requirements at all. For rows with many long character variables that consist mostly of blanks, the compression ratio can be very high. For rows with mixed variables, where most doubles do not have fractional parts and most character variables have a small amount of blank padding, the compression ratio is typically moderate. As with most cases of using compression, character variables tend to compress the most and the ratio depends on your data.

Compressed Tables and the DATA Step

This example shows how to use the COMPRESS= data set option for SAS Cloud Analytic Services.

Example Code 1.1 *Creating a Compressed Table with a DATA Step*

```
libname mycas cas host="cloud.example.com" port=5570;

data mycas.prdsale (compress=yes);
    set sashelp.prdsale;
run;
```

After the table is loaded into memory, you can access the compressed table with the mycas.prdsale table reference.

SAS Cloud Analytic Services supports the APPEND= data set option for compressed tables. This example shows how to add new (uncompressed) rows to the compressed table.

Example Code 1.2 *Appending Rows to a Compressed Table*

```
data mycas.prdsale(append=yes);
    somelib.newrows;
run;
```

Because the mycas.prdsale table is already compressed, the new rows are automatically compressed as they are appended to the table. Specifying COMPRESS= with APPEND= has no effect. If the table is compressed, the server compresses the new rows. If the table is not compressed, then the server does not compress the new rows even if you specify COMPRESS=YES. The compressed or uncompressed state of the table determines how rows are appended.

Partitioning and compression are supported together. This example creates a new in-memory table that is partitioned and compressed.

Example Code 1.3 *Creating a Partitioned and Compressed Table*

```
data mycas.iris (partition=(species) compress=yes);
    set sashelp.iris;
run;

data mycas.iris (append=yes);
    set somelib.moreirises;
run;
```

In the first DATA step statement, the Iris data set is loaded into memory on the server. The table is partitioned by the formatted values of the Species variable. The table is also compressed. In the second DATA step statement, the table is appended to with more rows. Because the in-memory table is already partitioned and compressed, the new rows are automatically partitioned and compressed when they are appended.

Compressed Tables and the CASUTIL Procedure

You can use the CASUTIL procedure to load data in memory on SAS Cloud Analytic Services.

```
proc casutil;
   load data=sashelp.prdsale casout="prdsale" compress;
quit;
```

This example uses the COMPRESS data set option to read the Prdsale data set from the sashelp library and compress it in-memory on the server. Be aware that you must specify the COMPRESS option for each table that you want to load in compressed form.

When you read SASHDAT tables into memory, compression depends on these factors for the resulting in-memory tables:

- whether a WHERE clause is used
- whether the SASHDAT table is compressed on disk

If you specify a WHERE clause when loading a SASHDAT file with compression, the server uncompresses the rows as it evaluates the WHERE clause. This results in an uncompressed in-memory table. The memory efficiencies of the SASHDAT table format are forfeited in this scenario because the server had to apply the WHERE clause.

If you do not specify a WHERE clause, the server ignores the COMPRESS option and relies on whether the SASHDAT file is compressed. If the SASHDAT file is compressed, the in-memory representation of the table is also compressed. If the SASHDAT file is not compressed, then neither is the in-memory representation. The server ignores the option so that it can keep the memory efficiencies of the SASHDAT file format: When a SASHDAT table is loaded in memory, the in-memory representation is identical to the on-disk representation.

Performance Considerations

Compression exchanges less memory use for more CPU use. It slows down any request that processes the data. An in-memory table consists of blocks of rows. When the server works with a compressed table, the blocks of rows must be uncompressed before the server can work with the variables. In some cases, a request can require five times longer to run with a compressed table rather than an uncompressed table.

For example, if you want to summarize two variables in a table that has 100 variables, all 100 columns must be uncompressed in order to locate the data for the two variables of interest. If you specify a WHERE clause, then the server must uncompress the data before the WHERE clause can be applied. Like the example where only two of 100 variables are used, if the WHERE clause is very restrictive, then there is a substantial performance penalty to filter out most of the rows.

Working with SASHDAT tables that are loaded by server is the most memory-efficient way to use the server. Using compressed SASHDAT tables preserves the memory efficiencies, but still incurs the performance penalty of uncompressing the rows as the server operates on each row.

Interactions

Here are the interactions for compressed tables and SAS programs.

- You can use a compressed table in programs like any other table.
- You can define calculated columns for compressed tables with the TEMPNAMES= data set option.
- You can append to compressed tables. This is also supported for compressed tables that have partitioning with or without ordering.

See Also

- “APPEND= Data Set Option” in *SAS Data Set Options: Reference*
- “COMPRESS= Data Set Option” in *SAS Data Set Options: Reference*

- "ORDERBY= Data Set Option" in *SAS Data Set Options: Reference*
- "TEMPNAMES= Data Set Option" in *SAS Data Set Options: Reference*

2

CAS LIBNAME Engine

CAS LIBNAME Engine Overview 37
Getting Started 39
Data Types 41
Summary of CAS LIBNAME Engine Data Types 41
VARCHAR Data Type 43
Support for Implicit Declaration of Data Types 51
Use of WHERE with the CAS LIBNAME Engine 52
Working with Temporary Columns 52
Variable Names and Data Set Names in CAS Engine 54
Troubleshooting CAS LIBNAME Engine Errors 54
The Call to Cloud Analytic Services Failed 54
Where to go from here 55

CAS LIBNAME Engine Overview

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. The CAS engine performs client/server communication. The engine is part of the SAS Compute Server and connects to the CAS server. For information about system requirements for the CAS engine, see *System Requirements for the SAS Viya Platform*.

The CAS LIBNAME engine connects a SAS session to a CAS session. The libref is your link between SAS and the in-memory tables on the CAS server. For information about usage, see this topic and those that follow. For the syntax, see "LIBNAME Statement: CAS Engine" on page 765.

When you assign a CAS engine libref, you are associating the libref with a CAS session in order to work with in-memory tables. A libref is associated with the session that you specify in the SESSREF= statement option.

The best practice for connecting to CAS is to first start a CAS session, and then specify the SESSREF=. See the below table for more information about how the LIBNAME Engine connects to CAS.

Table 2.1 *How the LIBNAME Engine Connects to CAS*

Number of Existing CAS Sessions	Specified SESSREF= Statement Option?	Connection Details
None	No	The CAS engine starts a new session when you specify the CAS LIBNAME statement. [1]
One	No	The libref is associated with the session in the SESSREF= system option. The system option is updated with the name of the most recently started CAS session.
One	Yes	The libref is associated with the session that you specify in the SESSREF= statement option.
More than one	No	The libref is associated with the session in the SESSREF= system option. The system option is updated with the name of the most recently started CAS session.

More than one	Yes	The libref is associated with the session that is specified in the SESSREF= statement option. You can view the setting with the following code:
		`proc options option=sessref;`
		`run;`

1 The CAS engine uses the CASHOST and CASPORT system options to make the connection, and the SESSREF system option to name the session.

By default, the libref uses the active caslib, which can change as caslibs are added and dropped. However, you can specify the CASLIB= LIBNAME option to bind the libref to a specific caslib. In this case, adding and dropping caslibs has no effect on the libref unless the bound caslib is dropped.

You can specify library options in the CAS LIBNAME statement or data set option in a DATA step or procedure step. When you specify a value for a data set option that has a corresponding CAS LIBNAME statement option (such as PROMOTE=) or a GLOBAL option, the data set option value takes precedence over the value for the CAS LIBNAME statement option.

To run procedures and the DATA step that use CAS processing or transfer data from CAS, you must do the following:

1. Connect to a server and start a CAS session.
2. Specify the CAS engine LIBNAME statement and use the libref with the input table name. When you assign a CAS libref, the CAS LIBNAME engine associates the libref with the active caslib unless you specify the LIBNAME statement

CASLIB= option. If you use the CASLIB= option, the CAS libref is bound to the caslib.

Here are two examples of the CAS LIBNAME statement:

```
/* The libref Mycas is associated */
/* with the active caslib.*/
/* If a new caslib is added, Mycas is associated */
/* with the new caslib.*/

libname mycas cas;

/* In the following statement, the libref Mycas is */
/* bound to the testTables caslib,*/
/* even if the active caslib changes */
/* to a caslib other than testTables.  */

libname mycas cas caslib=testTables;
```

3 Execute a procedure or DATA step.

4 At the end of your program, when you no longer need to access data in CAS, you can use the following statement to terminate your session to preserve resources:

```
cas casauto terminate;
```

Note: See *Base SAS Procedures Guide* for specific information about working with procedures in CAS.

Getting Started

The CAS engine makes it possible to run SAS Viya platform procedures and an in-memory DATA step on CAS in-memory tables. You must start a session with the CAS server and assign a CAS LIBNAME engine libref before the following types of data transfers can occur between SAS and the CAS server:

- reading SAS data sets from a loaded table and writing to a caslib as an in-memory table
- transferring CAS table data or table metadata as input to a DATA step or SAS procedure when the DATA step or procedure executes in SAS.

The example below demonstrates a few key points about the engine:

- The CAS engine is the key method for performing an in-memory DATA step.
- The CAS engine is required when using a SAS procedure to transfer data from the CAS server.

- The analytic procedures on the SAS Viya platform process in-memory tables with CAS actions, so the engine provides the way to identify the tables to use for analysis.
- The engine provides one way to perform data transfer between SAS and the server and is appropriate for smaller data sets. For larger data sets, the CASUTIL procedure can be used to perform a server-side load.

```
option casport=5570 cashost="cloud.example.com";                    /*1*/

cas casauto sessopts=(caslib=casuser);                              /*2*/

libname mycas cas;                                                  /*3*/

proc casutil;                                                       /*4*/
   load data=sashelp.cars casout="cars";
quit;

data mycas.bopc;                                        /*5*/
   set mycas.cars;
   where make in ("Buick", "Oldsmobile", "Pontiac", "Cadillac");
   wthp_ratio = round((weight / horsepower), .0001);
   drop msrp invoice;
run;

proc print data=mycas.bopc(obs=5);                                  /*6*/
run;

proc treesplit data=mycas.bopc;                                     /*7*/
   model mpg_highway = cylinders enginesize wheelbase length;
   class type drivetrain;
run;
```

1 Specify the connection information. The CASHOST= and CASPORT= system options are used to specify the connection information. As an alternative, your administrator might have set the default host and port for your server in a configuration file.

2 Start a session and set the active caslib to Casuser. You can start a session manually with the CAS statement. In this example, the CAS statement starts the CAS session named CASAUTO.

3 The LIBNAME statement assigns a CAS engine libref named Mycas, that you use to connect to the session CASAUTO. In subsequent procedure steps, input table names must begin with the Mycas libref.

   Some procedures require output table names to also begin with the CAS libref. See the documentation for your procedure for more information. In-memory DATA steps require a CAS engine libref on both the input and output table names.

4 Add the Sashelp.cars data set as an in-memory table.

5 Run an in-memory DATA step to create a new table with a subset of the rows and columns and add a new column. The input and output tables must begin with the CAS engine libref Mycas.

6 View five rows from the new data. The engine performs data transfer from the CAS server to SAS.

7 Use the Mycas libref to identify the input table for a procedure.

Data Types

Summary of CAS LIBNAME Engine Data Types

The CAS engine supports the storage of three data types. The following table provides information for each type:

Table 2.2 *CAS LIBNAME Engine Data Types*

Data Type	Description	Example	How Missing Values are Designated

CHAR $*n*	Stores a fixed-length character string. *n* is the number of bytes used to store all values. If the value stored contains fewer than *n* bytes, then SAS adds blanks (spaces) to the end of the value to fill the column.
	Length: 1 - 32767 bytes
		`data casuser.names;`
		`  length fname $20;`
		`  fname= "Sam";`
		`run;`
			zero or more blank spaces surrounded by single or double quotation marks:
			`fname="";`
			or
			`f='  ';`
			Note: The length of `fname="";` (defined with no blank spaces) is 1.

VARCHAR(*n*)[2]	Stores a varying-length character string. *n* is the maximum number of characters to store.
	Length: 0 - 536,870,911 characters (UTF-8 encoding)
		`data casuser.names;`
		`  length lname varchar(40);`
		`  lname="Adams";`
		`run;`
			zero or more blank spaces surrounded by single or double quotation marks:
			`lname="";`
			`lname='  ';`
			Note: The length of `lname="";` (defined with no blank spaces) is 0.

DOUBLE	Stores a numeric value, including dates and	`data casuser.names;`
		`  length num 8;`
		`  num = 25;`
			a single period
			`num1=.;`

Data Type	Description	Example	How Missing Values are Designated

	times, as a floating-point number.
	Length: 8 bytes for the CAS Engine[1]
		`run;`	For special numeric missing variables, the upper or lowercase letters A-Z, or the underscore (_) character:
			`.`
			`.A-.Z`
			`._`

1 If a shorter length is specified, then 8 bytes are used and a note is printed to the SAS log.

2 The CAS engine is required for storing VARCHAR variables in the output table of a DATA step. The DATA step can read CAS tables containing VARCHAR variables, but it cannot store them unless a CAS engine libref is specified on the output table.

The DATA step does not support zero-length character literals. A character literal consisting of two consecutive quotation marks ("" or '') is stored as a single blank character and has a length of 1. However, a CHAR variable can contain a zero-length string when it is assigned from an expression that returns a zero-length value, such as TRIMN("").

The VARCHAR data type supports zero-length character strings. However, an assignment of "" or '' does not create a zero-length string. In the DATA step, these literals represent a single blank character. A zero-length VARCHAR value must be produced by an expression that returns a zero-length string, such as TRIMN("").

To determine whether a CHAR variable is missing, use the MISSING function:

```
 /*  lc=1, lvc=0 */
data _null_;
  length vc varchar(32);
  c = '';   /* no blanks */
  vc = '';
  lc = lengthc(c);
  lvc= lengthc(vc);
  put 'char length is ' lc;
  put 'varchar length is ' lvc;
  if missing(c) then put 'c is missing';
  if missing(vc) then put 'vc is missing';
run;
```

The DATA step above returns `lc=1` and `lvc=0`.

For more information about data types in CAS, see Chapter 9, "Data Types," on page 609.

The SAS V9 engine supports only the CHAR and NUMERIC types. For more information about data types that are supported by the SAS V9 Engine, see "Data Types" in *SAS Programmer's Guide: Essentials*.

SAS Cloud Analytic Services does not support the LENGTH statement for numeric data types and does not support DOUBLE columns that use less than 8 bytes. If you use the LENGTH statement to store numeric data with less than 8 bytes and then load that data into CAS, the server stores the data with 8 bytes. If you have

several numeric columns that use less than 8 bytes, the in-memory table size can be much larger than what is required for a .sas7bdat file.

VARCHAR Data Type

Definition

VARCHAR(*length* | *)

Stores a varying-length character string. *length* is the user-defined maximum length of the character string, and * indicates that the maximum storage size is 536,870,911 characters.

Syntax

```
LENGTH (variable-naame) VARCHAR(length|*);
ARRAY array-name[N] VARCHAR(length|*);
ARRAY array-name[*] VARCHAR-variables;
```

variable-name
specifies one or more variables that are assigned the type VARCHAR.

length
specifies a numeric constant that is the user-defined maximum number of characters that can be stored in the VARCHAR variable. This value can be up to 536,870,911 characters in length. Uninitialized VARCHAR variables are given a length of 1 by default. This value is based on the defined range.

```
length xyz varchar(32);
```

Uninitialized VARCHAR variables are given a length of 1 by default. 1 is the minimum length of a VARCHAR variable.

Range 1 to 536,870,911 (UTF-8) characters (2^{31} bytes)

See "PROC CONTENTS Output for VARCHAR Variables" on page 47

*
specifies that SAS uses the maximum length allowed, which is 536,870,911 characters. When assigning a character constant to a VARCHAR variable, the character constant is limited to 32,767 bytes.

```
length xyz varchar(*);
```

Uninitialized or missing VARCHAR variables are given a length of 1 by default. This value is based on the defined range for VARCHAR variables, in which 1 is the minimum length a VARCHAR variable can be.

See "PROC CONTENTS Output for VARCHAR Variables" on page 47

array-name
specifies the name of the array. Defines the elements in an array as a list of VARCHAR variables.

When using a list of VARCHAR variables with the ARRAY statement, you can use the hyphen (–), colon / prefix, and double-dash lists:

```
array arr1[*] v1-v5;
array arr2[*] v:;
array arr3[*] v1--v5;
```

You cannot use VARCHAR character lists specified as _CHARACTER_.

N

describes the number and arrangement of elements in the array

*

specifies the maximum length allowed, 536,870,911 characters. When assigning a character constant to a VARCHAR variable, the character constant is limited to 32767 bytes.

```
array myArray{*} varchar(*) a1 a2 a3 ('a','b','c');
```

Requirements

requires a CAS engine libref on the output table

requires additional storage space. See VARCHAR Variable Storage. on page 49

Engine

CAS engine only

Note

If you have sites that support multiple languages, consider running your SAS session using UTF-8 encoding and using the VARCHAR data type to minimize character conversion issues.

Details

The VARCHAR type is a varying length character data type whose length represents the maximum number of characters you want to store in a column. VARCHAR variables have the following characteristics:

- their length is measured in terms of characters rather than bytes
- their length varies depending on the values present

These characteristics are in contrast to those of the CHAR data type, in which length is fixed and measured in bytes.

For example, a VARCHAR(100) can store up to 100 characters, but the actual storage used in any given row depends on the lengths of the individual values in the row and column.

For example, if a VARCHAR variable, **vc2**, is defined as a VARCHAR(100) variable, this means that it can store up to 100 characters. But because the value “abc” contains only 3 characters and each character uses one byte, only 3 bytes of memory are allocated for the value in that row. A fixed-length CHAR column, on the other hand, takes up the defined number of bytes regardless of the actual value. In the case of the example, the fixed-width CHAR with a defined length of 100 would use all 100 bytes even though the actual value “abc” needs only 3 bytes.

Example: Create a VARCHAR Using the LENGTH Statement

```
libname mycas cas;
data mycas.roman;
   length vc32 varchar(32);
      do i = 1 to 10;
         vc32 = put(i, ROMAN.);
         output;
      end;
run;
```

Example: Create a VARCHAR Variable Using the ARRAY Statement

```
data mycas.test;
   array test{*} varchar(*) a1 a2 a3 ('a','b','c');
   put test[1]; put test[2]; put test[3];
run;
```

When to Use a VARCHAR Data Type

The VARCHAR data type is useful because it can save space when the lengths of the column values vary. With fixed-width data types, any space that is not used by the value in the column is padded with trailing blanks, which wastes space. The entire space is blocked out in memory whether the value needs the space or not. With varying-length data types, such as VARCHAR, only the space that is needed is used (there are no trailing blanks).

- CHAR – use a fixed-width CHAR when the sizes of the column data entries are similar. Fixed-width columns are usually accessed faster.
- VARCHAR(n) – use when the sizes of the column data vary considerably but you are reasonably certain they will not exceed a certain width.
- VARCHAR(*) – use when the sizes of the column data vary considerably and the column width might exceed any limits you might place on it.

In most cases, you should take advantage of VARCHAR support. However, if values are consistently short, such as an ID column of airport codes, then a fixed-width CHAR variable uses less memory and runs faster. This is because VARCHAR values require 16 bytes plus the memory needed to store the VARCHAR value. So, if your values are always smaller than 16 bytes, you can save memory and processing time by using a CHAR type variable instead.

Range

SAS defines the length of a VARCHAR data type in terms of characters rather than bytes. The maximum length of a VARCHAR variable is 536,870,911 Unicode characters, or, 2^{29}-1 bytes. This means that up to 536,870,911 characters, or 2,147,483,644 bytes of data or can be stored in a VARCHAR variable. The maximum length in bytes is calculated by multiplying 536,870,911 by the maximum length that any one character in the UTF-8 character set can be, which is 4 bytes.

Setting Missing VARCHAR Variables

You can use the CALL MISSING routine to set a VARCHAR variable to missing:

```
if var1 = "abc" then call missing(var1);
```

You can use the MISSING function to test whether a VARCHAR variable is missing.

```
if missing(var2) then var2 = "missing";
```

VARCHAR Support for Implicit and Explicit Data Type Conversion

Type conversion happens when a SAS DATA step or procedure moves data from a CAS table into a SAS data set. The data must be converted from the data type supported by the CAS engine to a data type supported by the SAS V9 engine. VARCHAR type variables are not supported by the Base SAS V9 engine, so they are automatically converted from VARCHAR types to fixed-length CHAR types.

The DATA step supports the processing of VARCHAR data. However, only the CAS engine supports the VARCHAR data type. This means that the DATA step can read in and process VARCHAR data, but the data is converted to a CHAR when it is stored as a SAS data set.

When you convert between CAS data and SAS data, the supported data type conversions are defined by the engine.

If character strings declared as VARCHAR data types are converted to the CHAR data type, values that are too long for the CHAR data type are truncated.

Data types can be converted from one type to another either implicitly or explicitly.

In *implicit conversions* SAS automatically converts data from one type to another and the conversions are not visible to the user. An example is when you save a CAS table containing a VARCHAR as a SAS data set. The VARCHAR is implicitly converted to a CHAR in the output data set.

In *explicit conversions*, users deliberately convert one type to another using programming statements.

SAS language elements that can explicitly convert one data type to another are the PUT function and the INPUT function. The following converts the numeric value of a VARCHAR to a DOUBLE data type and writes the output to a CAS table.

```
data mycas.new;
   length vc varchar(40);
   vc = '5000';
   num = input(vc,8.);
run;
proc contents data=mycas.new; run;
```

Output 2.1 *PROC CONTENTS Output Showing Explicit Data Type Conversion*

Alphabetic List of Variables and Attributes

#	Variable	Type	Len		Max Bytes Used
			Bytes	Chars	
2	num	Num	8		
1	vc	Varchar	40	40	4

PROC CONTENTS Output for VARCHAR Variables

The CONTENTS procedure provides metadata about a CAS table including detailed information about the variables in the table.

When a table contains a VARCHAR variable, PROC CONTENTS displays extra length information about the variable that is not normally displayed when a VARCHAR is not present in the table:

PROC CONTENTS Output for VARCHAR Variables

Alphabetic List of Variables and Attributes

#	Variable	Type	Len		Max Bytes Used
			Bytes	Chars	
1	vc1	Varchar	.	.	1
2	vc2	Varchar	400	100	9
3	vc3	Varchar	40	10	3

PROC CONTENTS Output for CHAR Variables

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	ch1	Char	10
2	ch2	Char	100
3	ch3	Char	3

- **Len** displays the length specified by the user when the variable is declared. The defined length is given in 2 sub-columns, Chars and Bytes:
  - **Chars** displays the *defined* length in characters, or the length that the user specifies n the VARCHAR(*n* | *) statement when the variable is declared.
  - **Bytes** displays a length in bytes that SAS calculates based on the defined length and the SAS session encoding. SAS calculates the length by multiplying the defined length by the largest possible number of bytes required to store any one character in the character set encoding. The defined length is the value defined by the user in the VARCHAR(*n*) statement.

    For example, if your SAS session encoding is UTF-8, then the **Len Bytes** value is calculated as (*n* x 4), where *n* is the length defined in the VARCHAR(*n*) statement and 4 is the largest possible number of bytes

required to store any character in the UTF-8 character set. Similarly, if the local SAS session encoding is Latin1, then the **Len Bytes** value is calculated as (*n* x 1), where *n* is the length defined in the VARCHAR(*n*) statement and 1 is the largest number of bytes required to store any character in the Latin1 character set.

- **Max Bytes Used** displays the number of bytes that are used to store the longest string value in the column.

```
data mycas.new;
  length vc1 varchar(*)   /* defined length of vc1, shown in "Len
Chars" column */
         vc2 varchar(100) /* defined length of vc2, shown in "Len
Chars" column */
         vc3 varchar(10); /* defined length of vc3, shown in "Len
Chars" column */
  vc2 = "123456789";   /* actual bytes used for vc2, shown in Max
Bytes Used column */
  vc3 = "abc";         /* actual bytes used for vc3, shown in Max
Bytes Used column */
run;
proc contents data=mycas.new; run;
```

defined length | actual length

#	Variable	Type	Len		Max Bytes Used
			Bytes	Chars	
1	vc1	Varchar	.	.	1
2	vc2	Varchar	400	100	9
3	vc3	Varchar	40	10	3

In a UTF-8 session, the length in Bytes is 4 times the length in Chars.

If you specify VARCHAR(*), then SAS defines the length as the maximum allowable length.

VARCHAR Length with Implicit Type Conversion

VARCHAR variables are not supported by the SAS V9 engine. SAS automatically converts VARCHAR variables to CHAR variables if you try to store them in anything other than a CAS table.

When converting a variable from a VARCHAR to a CHAR, the length of the CHAR depends on how the VARCHAR is originally defined.

- VARCHAR(*) – If a table that contains a VARCHAR(*) definition is saved as a SAS data set, the VARCHAR is automatically converted to a CHAR that has a length equal to the Max Bytes Used for the original VARCHAR.

- VARCHAR(*n*) – If a table that contains a VARCHAR(*n*) definition is converted to a SAS data set, then the length of the variable depends on the local SAS session encoding. The length is calculated as follows: SAS multiplies the current length of the VARCHAR by the maximum value that a character's length can be in the local SAS session encoding.
  - If the local SAS session encoding uses single-byte characters, then the VARCHAR is converted to a CHAR with a length of (*n* x 1). *n* is the length of the original VARCHAR and 1 is the largest number of bytes required to store any character in the character set.
  - If the local SAS session encoding uses double-byte characters, then the VARCHAR is converted to a CHAR with length (*n* x 2). *n* is the length of the original VARCHAR and 2 is the largest number of bytes required to store any character in the character set.
  - If the local SAS session encoding uses UTF-8 encoding, then the VARCHAR is converted to a CHAR with length (*n* x 4). *n* is the length of the original VARCHAR and 4 is the largest number of bytes required to store any character in the character set.

Note: When assigning a character constant to a VARCHAR variable, the character constant is limited to 32767 bytes.

VARCHAR Variable Storage

When CHAR values are stored, they are right-padded with spaces to the specified length. VARCHAR values are not padded. When VARCHAR values are stored, they are stored with descriptor information in each row that takes space along with the data value. The descriptor is a 16-byte prefix that is stored with the data value in each row and that contains information about the length of the data value.

Restrictions for the VARCHAR Data Type in the CAS Engine

Not all SAS language elements support the VARCHAR data type, even with the CAS engine. There are also differences in how some SAS language elements behave with VARCHAR variables. These limitations and behavior differences are listed in the table below.

Table 2.3 *Restrictions and Notable Behaviors for the VARCHAR Data Type in the CAS Engine*

Feature	Description

ATTRIB	You cannot use the ATTRIB statement to create VARCHAR variables.
BY statement	The BY statement uses a fixed width for VARCHAR variables. Using a VARCHAR(*) type in the BY statement might cause unexpected results.
Formats	In DATA step processing, the width of VARCHAR formats is measured in characters. **Note:** Prior to 2025.02, the width of VARCHAR formats was measured in bytes.
Functions	When passing a character value to a function, numbers indicating a length or position that are passed to the function or returned by the function are in units of bytes. When passing a VARCHAR variable to functions, these numbers are in units of characters. Of note, this includes the INDEX and SUBSTR functions. See "Index CHAR and VARCHAR Character Strings" in *SAS Cloud Analytic Services: DATA Step Programming* for a related example.
KEY= on SET and MODIFY statements	VARCHAR variables are not supported by the KEY= option in either the SET or MODIFY statements.
PUT statement (to ODS output)	VARCHAR variables are not supported with the PUT statement when the DATA step writes output using ODS.
VARCHAR	The VARCHAR data type is supported by the CAS engine but not by the V9 engine. This means that to create or store a VARCHAR variable, you must use the CAS engine. The SAS DATA step (with the V9 engine) can read data containing VARCHAR variables but it converts and stores them as CHAR data types.
Variable Lists	Selecting a character variable range for character variable lists (for example, *a-character-f*) is not supported for VARCHAR variables because VARCHAR variables are not fixed-width character variables. For example, you cannot specify VARCHAR variables using the following shorthand forms: _CHARACTER_ *var1*–CHARACTER-*varN*
Concatenating variables	When concatenating character values, the result is a character value that is limited to 32767 bytes. In this example, the character result of the concatenation is

assigned to a VARCHAR variable and the result value is limited to 32767 bytes.

```
length vc varchar(*);
vc = "a string that is 32000 bytes long"
     || "another string that is 32000 bytes long";
```

To go beyond the 32767 byte limit, include a VARCHAR variable in the concatenation. The concatenation result is a VARCHAR that can go beyond 32767 bytes.

```
length vc1 vc2 varchar(*);
vc1 = "a string that is 32000 bytes long";
vc2 = vc1 || "another string that is 32000 bytes long";
```

Support for Implicit Declaration of Data Types

The CAS engine supports implicit data type declaration for DOUBLE (NUMERIC) and CHAR type variables.

Implicit declaration means that you do not have to explicitly declare a variable's type or length before using it. You can create a new variable and use it for the first time in an assignment statement without having to explicitly declare its type or length. When you create a variable in this way, SAS determines the type based on the values that you assign to the variable.

- Variables that are assigned a character string value are implicitly defined as a CHAR types with a default length of 8 bytes.
- Variables that are assigned an integer value are implicitly defined as DOUBLE types with a default length of 8 bytes.

  Note: This is different from the V9 engine, which supports a length range of 1 - 8 bytes for NUMERIC types.

- Implicit type declaration is not supported for VARCHAR variables. VARCHAR variables must be explicitly declared in either a LENGTH statement or an ARRAY statement.

In the following DATA step, the type and length for variables **x** and **y** are set implicitly:

```
libname mycas cas;
data mycas.datatypes;
   x=1;
   y='hello';
run;

proc contents data mycas.datatypes;
run;
```

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
1	x	Num	8
2	y	Char	5

For information about data types supported by the SAS V9 engine, see "Data Types" in *SAS Programmer's Guide: Essentials*.

Use of WHERE with the CAS LIBNAME Engine

WHERE processing enables you to conditionally select a subset of rows so that SAS processes only the rows that meet specified conditions. If you use the WHERE= data set option or WHERE statement with the CAS LIBNAME engine, the engine attempts to resolve the WHERE expression on the CAS server so that less data is transferred from the server to SAS. Performance is often improved with large tables when the WHERE expression qualifies only a relatively small subset.

There are WHERE expressions that cannot be processed by CAS, so the WHERE evaluation happens in SAS instead.

Note: You can set system option MSGLEVEL=i to view messages about CAS processing in the SAS log.

Working with Temporary Columns

The CAS LIBNAME engine enables users to create temporary columns. Temporary columns are useful when preparing and exploring data because you can rapidly revise the expressions that create the temporary columns. The alternative is to create in-memory tables with permanent columns. In this case you might be creating additional in-memory tables that consume resources. These tables might require additional programming statements to drop them if they do not lead to better analytical insights.

The TEMPNAMES= option and the SCRIPT= option are required arguments for working with temporary columns. The following SAS code defines three variables in a simple script and captures it in the NewCols file reference. The names of the variables that you can add to the mycas.cars table are then listed in the TEMPNAMES= data set option. Only the variables listed in the TEMPNAMES= option are added to the input table, which conserves resources in the server.

If you want to add a temporary character column to the input table, you must follow the name of the variable in the TEMPNAMES= option with a dollar sign ($) and the variable length in bytes.

```
cas casauto host="cloud.example.com" port=5570;
proc casutil;
  load data=sashelp.cars(where=(type ne 'Hybrid')) replace;
quit;

filename newcols temp;                                          /*1*/

data _null_;                                                    /*2*/
   file newcols;
   put "t1 = round(mpg_highway/mpg_city,0.5);";
   put "t2 = round(cylinders/enginesize,0.2);";
   put "t3 = msrp / invoice;"

run;

libname mycas cas sessref=casauto;

proc mdsummary data=mycas.cars(tempnames=(t1 t2 t3)    /*3*/
  script=newcols);
   output out=mycas.carssum;
   var mpg_city mpg_highway t1 t2 t3;
run;

proc print data=mycas.carssum;
  var _column_ -- _sum_;
run;
```

1 The example shows how to use the FILENAME statement with the TEMP option to create a temporary file. The expressions for each column are written to the temporary file in the DATA step that follows. You could specify a path for the file instead of using the TEMP option.

2 A DATA _NULL_ statement is used to write three expressions to the temporary file. As an alternative, you can enter the expressions in a text file and specify the path.

3 The MDSUMMARY procedure produces summary statistics for the Mpg_City and Mpg_Highway variables in addition to the t1, t2, and t3 variables listed in the TEMPNAMES= data set option. The OUTPUT statement creates an in-memory table named Mycas.Carssum. The table includes the variables from the TEMPNAMES= data set option.

The "ONDEMAND= Data Set Option" in *SAS Data Set Options: Reference* is optional for working with temporary columns. ONDEMAND= enables users to customize how their data is evaluated: one row at a time or collectively at the outset.

Variable Names and Data Set Names in CAS Engine

The rules for naming variables and tables when running the DATA step in SAS and in CAS are as follows:

- Names can be up to 32 bytes in length.
- Names can contain alphanumeric characters and the underscore (_) character.
- Names can contain special characters if they are specified as name literals (in quotation marks followed by n). Names cannot contain special characters / \ * ? " < > | : -.
- Name literals enable you to use many more characters in the name, including blanks, national characters, and numerals as the first character in the name. See "SAS Name Literals" in *SAS Programmer's Guide: Essentials*

Here is an example showing these rules:

```
data _null_ / sessref=casauto;
      greater_than_8_but_less_than_32 = 100;
      'blanks & special chars'n = 'hello';
      put greater_than_8_but_less_than_32
      put 'blanks & special chars'n;
run;
```

Note: These rules are identical to the rules for names in SAS when system option VALIDVARNAME=ANY and system option VALIDMEMNAME=EXTEND.

Troubleshooting CAS LIBNAME Engine Errors

The Call to Cloud Analytic Services Failed

```
ERROR: The call to Cloud Analytic Services failed.
ERROR: The action cannot be processed while an action is active.
```

Cause	Solution
A procedure or DATA step with more than one output uses one CAS LIBNAME engine libref for input and the same libref for output, such as output created using the OUT= option or a TABLES statement.	■ Use multiple DATA steps. For best results, output a single table and use the same CAS engine libref that was used for input. ■ Use multiple PROC steps. ■ Create output data with a different engine.

There are several scenarios that cause the error. One example of how the error occurs is when the CAS LIBNAME engine is used for more than one output table. See the following code sample that demonstrates this specific scenario.

```
libname mycas cas wts=1k;
data mycas.one mycas.two;
   set sashelp.cars;
run;
```

In the above code example, the WRITETRANSFERSIZE= option is set to an artificially small value to trigger the error. However, the solution is to use multiple DATA steps in the program rather than increase the transfer size.

Where to go from here

For information about CAS LIBNAME syntax, see the following documentation.

- “LIBNAME Statement: CAS Engine” on page 765
- *SAS Data Set Options: Reference*
- “CAS LIBNAME Options” on page 615

3

CASUTIL Procedure

Overview: CASUTIL Procedure ***57***
What Does the CASUTIL Procedure Do? 58
Terminology 58

Syntax: CASUTIL Procedure ***59***
PROC CASUTIL Statement 60
APPEND Statement 61
ALTERTABLE Statement 63
CONTENTS Statement 64
COPY Statement 65
DELETESOURCE Statement 67
DROPTABLE Statement 68
INDEX Statement 69
LIST Statement 72
LOAD Statement 73
PARTITION Statement 80
PROMOTE Statement 83
SAVE Statement 84
UPDATE Statement 86

Usage: CASUTIL Procedure ***87***
Enclose Values in Quotation Marks 87
Subdirectories and File Name Matching 88
Limitations and Restrictions 90

Results: CASUTIL Procedure ***91***
Procedure Output 91
ODS Table Names 93

Examples: CASUTIL Procedure ***94***
Example 1: Load a CSV File into CAS 94
Example 2: Append Rows to an In-Memory Table 95
Example 3: Promote a Table 98
Example 4: Saving and Loading Encrypted SASHDAT Files 99
Example 5: Copying and Saving a Subset of the Copied Table 102
Example 6: Rename a Table and Rename Columns 103

Overview: CASUTIL Procedure

What Does the CASUTIL Procedure Do?

The CASUTIL procedure works with tables in SAS Cloud Analytic Services, SAS data sets in SAS libraries, and external files. The procedure has three functional areas:

- data transfer
- table and file information
- drops tables and deletes files

In the area of data transfer, you can perform the following operations:

- load a data set from a SAS library into a memory on SAS Cloud Analytic Services.
- save in-memory tables in a caslib to the data source that is associated with the caslib.
- load files from the data source that is associated with a caslib into memory on SAS Cloud Analytic Services.

For file and table information, you can perform the following operations:

- view column names, data types, and other column information.
- list the in-memory tables in a caslib.
- list the files in a caslib's data source.

In the area of table and file management, the procedure enables you to drop in-memory tables. Dropping a table frees resources in the server but leaves the file in the data source that is associated with the caslib untouched. The procedure also enables you to delete files from the data source that is associated with the caslib.

The CASUTIL procedure executes without using the RUN statement. After you submit the PROC CASUTIL statement, you can submit additional procedure statements without submitting the PROC statement again. Use the QUIT statement to terminate the procedure.

Terminology

The following terms are used throughout the CASUTIL procedure documentation:

file
: is used to refer to the source data that is in a caslib's data source. For a caslib that uses a path-based data source, this is natural. For a caslib that uses a database as a data source, the tables in the database are referred to as files.

table
: is used to refer to in-memory data. After a file (using the preceding definition) is loaded into the server, it is referred to as a table.

Syntax: CASUTIL Procedure

PROC CASUTIL <*options*>;

APPEND SOURCE="*table-1*"<SRCCASLIB="*caslib*" >
<DATASOURCEOPTIONS=(*data-source-options*)>
TARGET="*table-2*" <TGTCASLIB="*caslib*">;

ALTERTABLE CASDATA="*table-name*" <INCASLIB="*caslib*"> <LABEL="*label*">
<RENAME | CASOUT="*table-name*">
<COLUMNS={*column-options-1* <,{*column-options-n*}>}>
<COLUMNORDER={"*variable-name*" <,"*variable-name-n*">}>
<DROP={"*variable-name*" <,"*variable-name-n*">}>
<KEEP={"*variable-name*" <,"*variable-name-n*">};

CONTENTS CASDATA="*table-name*" <INCASLIB="*caslib*"> <*options*>;

DELETESOURCE CASDATA="*file-name*" <INCASLIB="*caslib*">
<DATASOURCEOPTIONS={*data-source-options*}>
<DELETEOPTIONS={*delete-options*}> <QUIET>;

COPY CASDATA="*file-name*" <INCASLIB="*caslib*">
CASOUT="*table-name*"<OUTCASLIB="*caslib*">
<DATASOURCEOPTIONS=(*data-source-options*)>
<INDEXVARS=((*variable-name-1*),<(*variable-name-n*)>)><*options*>;

DROPTABLE CASDATA="*table-name*" <INCASLIB="*caslib*"> <QUIET>;

INDEX CASDATA="*table-name*"
<INCASLIB="*caslib*">
<VARS={ {*casinvardesc-1*}, {<*casinvardesc-n*>} }>
<WHERE="*where-expression <logical-operator where-expression*">>
<IMPORTOPTIONS=(FILETYPE=*file-type <file-type-options>*)>
<DATASOURCEOPTIONS=(*data-source-options*)>
<CASOUT="*table-name*">
<OUTCASLIB="*caslib*">
<LABEL="*label-name*">
<COMPRESS><PROMOTE><REPLACE><REPLICATION|COPIES=*integer*>

```
    <INDEXVARS=((variable-name-1),<(variable-name-n)>)>;
  LIST <FILES | TABLES> <options>;
  LOAD CASDATA="file-name" | DATA=SAS-data-set | FILE="SAS-file" <options>;
  PARTITION CASDATA="table-name"
    <INCASLIB="caslib">
    <VARS={ {casinvardesc-1}, {<casinvardesc-n>} }>
    <WHERE="where-expression <logical-operator where-expression">>
    <PARTITIONBY | GROUPBY=(group-by-variable-1 <group-by-variable-n>)>
    <IMPORTOPTIONS=(FILETYPE=file-type <file-type-options>)>
    <DATASOURCEOPTIONS=(data-source-options)>
    <CASOUT="table-name">
    <OUTCASLIB="caslib">
    <LABEL="label-name">
    <COMPRESS><PROMOTE><REPLACE><REPLICATION|COPIES=integer>
    <INDEXVARS=((variable-name-1),<(variable-name-n)>)>
  PROMOTE CASDATA="table-name" <INCASLIB="caslib">
    <CASOUT="table-name">
    <OUTCASLIB="caslib">
    <DROP | KEEP>;
  SAVE CASDATA="table-name " <INCASLIB="caslib">
    <CASOUT="file-name"> <OUTCASLIB="caslib"> <EXPORTOPTIONS={export-
    options}> <IMPORTDATASOURCEOPTIONS = {import-data-source-options}>
    <options>;
  UPDATE CASDATA="table-name"
    <INCASLIB="caslib">
    <SET={ {var="variable-name", value="string"} }>
    <WHERE="where-expression <logical-operator where-expression">;>
QUIT;
```

PROC CASUTIL Statement

Manages tables and files in SAS Cloud Analytic Services.

See: For examples of the CASUTIL procedure in real-life scenarios, see *SAS Cloud Analytic Services: User's Guide*.

Syntax

PROC CASUTIL <*options*>;

Optional Arguments

INCASLIB=*caslib*
specifies the input caslib for the procedure. This option does not change the active caslib for your session. If you do not specify this option here or in a statement, such as LOAD, then the active caslib is used.

Specifying the caslib to use is a best practice until you develop experience working with the active caslib with this procedure, the CAS LIBNAME engine, and other procedures.

OUTCASLIB=*caslib*
specifies the output caslib for the procedure. This option does not change the active caslib for your session. If you do not specify this option here or in a statement, such as LOAD, then the active caslib is used.

SESSREF=*session-reference*
specifies the session to use with the procedure. If you omit SESSREF=, then procedure uses the session that specified in the &_SESSREF_ macro variable.

Alias SESSION=

APPEND Statement

The append statement concatenates two tables.

Syntax

APPEND SOURCE="*table-1*"<SRCCASLIB="*caslib*" >
<DATASOURCEOPTIONS=(*data-source-options*)>
TARGET="*table-2*" <TGTCASLIB="*caslib*">;

Required Arguments

SOURCE
specifies the source table that is appended to the target table.

TARGET
specifies the target table.

Optional Arguments

SRCCASLIB="*caslib*"
specifies the caslib that contains the source table.

DATASOURCEOPTIONS=(*data-source-options*)
specifies overrides for the DATASOURCE= options for the input caslib. For more information, see "Platform Data Sources" on page 125 and "Data Connectors" on page 167.

TGTCASLIB="*caslib*"
specifies the caslib that contains the target table.

Example: Concatenate Two Tables Using the APPEND Statement

This example appends the table Buick_cars to the table FordChry_cars.

```
proc casutil;
   load
data=sashelp.cars(where=(make="Buick"))                    /*1*/
      casout="buick_cars"
      label="Some makes from the Sashelp.Cars sample data."
      replace;
   load data=sashelp.cars(where=(make in ("Ford",
"Chrysler")))     /*2*/
      casout="FordChry_cars"
      replace;
   list
tables;                                                    /*3*/
   append srccaslib='casuser' source='buick_cars'

datasourceoptions={dataTransferMode='serial'}               /*4*/
      target='FordChry_cars' tgtcaslib='casuser';
   list tables;                                                  /
*5*/
quit;
```

1 Create the Buick_cars table using the LOAD statement.

2 Create the FordChry_cars table using the LOAD statement.

3 List the tables. In the SAS log, notice that Buick_cars has 9 rows and Fordchry has 38 rows.

4 Append the source table Buick_cars to the target table FordChry_cars.

5 List the updated table. In the SAS log, noticed that Buick_cars has 9 rows and FordChry_cars now has 47 rows.

Example Code 3.1 *Listing of the Original Buick_cars and FordChry_cars Tables*

```
Table Name     Label                                              of Rows  of
Columns  Columns  encoding  Created
BUICK_CARS     Some makes from the Sashelp.Cars sample data.            9
15        0  utf-8     2021
FORDCHRY_CARS  2004 Car Data                                          38
15        0  utf-8     2021
```

Example Code 3.2 *Listing of the Appended Buick_cars and FordChry_cars Tables*

```
Number      Number  Indexed  NLS
Table Name     Label                                                 of Rows  of
Columns  Columns  encoding  Created
BUICK_CARS     Some makes from the Sashelp.Cars sample data.             9
15         0  utf-8     2021
FORDCHRY_CARS  2004 Car Data                                           47
15         0  utf-8     2021
```

ALTERTABLE Statement

The ALTERTABLE statement enables you to change your table. For example, you can rename the table, change labels and formats, drop columns, and so on.

Syntax

ALTERTABLE CASDATA="*table-name*" < INCASLIB="*caslib*" > < LABEL="*label*" >

<RENAME | CASOUT="*table-name*">

< COLUMNS={{*column-options-1*} <,{*column-options-n*}>} >

< COLUMNORDER={"*variable-name*" <,"*variable-name-n*">} >

< DROP={"*variable-name*" <,"*variable-name-n*">} >

< KEEP={"*variable-name*" <,"*variable-name-n*">} >

Required Argument

CASDATA="*table-name*"
specifies the name of the in-memory table to change.

Optional Arguments

INCASLIB="*caslib*"
specifies the caslib with the in-memory table to change. If specified, this option overrides the INCASLIB= value in the procedure statement or the active caslib.

LABEL="*label*"
specifies a new label for the table.

RENAME | CASOUT= "*table-name*"
specifies a new name for the table.

COLUMNS={{*column-options-1*} <,{*column-options-n* }>}
specifies how to alter columns. After you specify the column name, you can drop a column or assign a new format, label, or name. Following are the column options:

NAME="*variable-name*"

<DROP= TRUE | FALSE >

<FORMAT= *"format-name "*>

<LABEL= *"label "*>

<RENAME= *"new-variable-name "*>

COLUMNORDER={*"variable-name*" <,*"variable-name-n "*>}
specifies the order of the columns.

DROP={*"variable-name*" <,*"variable-name-n "*>}
specifies the variables to drop from the table.

KEEP={*"variable-name*" <,*"variable-name-n "*>}
specifies the variables to keep in the table.

CONTENTS Statement

The CONTENTS statement displays table metadata such as column names and data types for files or in-memory tables.

Examples: "Example 3: Promote a Table" on page 98
"Example 4: Saving and Loading Encrypted SASHDAT Files" on page 99

Syntax

CONTENTS CASDATA=*"table-name"* <INCASLIB=*"caslib"*> <*options*>;

Required Argument

CASDATA="*table-name*"
specifies the name of the file or table.

Optional Arguments

IMPORTOPTIONS=(FILETYPE="*file-type*" <*file-type-options*>)
specifies the file format and options. Specify this option only if you specify a file name in the CASDATA= option. If you want to display metadata for an in-memory table (the result of a LOAD statement), then do not specify this option.

FILETYPE="BASESAS" | "CSV" | "DTA" | "EXCEL" | "HDAT" | "LASR" | "XLS"
specifies the file format.

file-type-options
specifies options for importing the data. For more information, see "Platform File Types" on page 153 and "Data Connectors" on page 167.

INCASLIB=*caslib*
specifies the caslib that is associated with the file or table. If specified, this option overrides the INCASLIB= value in the procedure statement or the active caslib.

DATASOURCEOPTIONS=(*data-source-options*)
specifies overrides for the DATASOURCE= options for the caslib. For more information, see "Platform Data Sources" on page 125 and "Data Connectors" on page 167.

Alias OPTIONS=

VARS=((*casinvardesc-1*) < (*casinvardesc-2*) ...>)
specifies the variables to include. If you do not include this option, all variables are included.

The value for *casinvardesc* is described in "VARS=((*casinvardesc-1*) < (*casinvardesc-2*) ...>)" on page 78.

Alias VARLIST=

Example: Viewing Column Names from a CSV File

```
contents casdata="somefile.csv" importoptions=(filetype="csv");
```

COPY Statement

The COPY statement copies one table from a caslib's data source to another. The copy of the table is saved to the data source that is associated with the caslib.

Syntax

COPY CASDATA="*file-name*" <INCASLIB="*caslib*">
CASOUT="*table-name*"<OUTCASLIB="*caslib*">

<DATASOURCEOPTIONS=(*data-source-options*)>
<INDEXVARS=((*variable-name-1*),<(*variable-name-n*)>)>
<*options*> ;

Required Arguments

CASDATA="*table-name*"
specifies the name of the file to load from the server-side data source that is associated with the INCASLIB= option or the active caslib.

CASOUT="*table-name*"
specifies an alternative name for the file. A file is created in the data source that is associated with the caslib from the OUTCASLIB= option. By default, a .sashdat file suffix is added when you save to a path-based caslib. If you specify a .csv suffix, a CSV file is saved.

Note This argument does not follow or enforce SAS naming rules such as the name literal syntax.

Tip Some data sources support table names that exceed 32 bytes. Use this option to limit table names to 32 bytes so that you can access the table with the CAS LIBNAME engine when you load it again.

Optional Arguments

DATASOURCEOPTIONS=(*data-source-options*)
specifies overrides for the DATASOURCE= options for the input caslib. For more information, see "Platform Data Sources" on page 125 and "Data Connectors" on page 167.

Alias OPTIONS=

INCASLIB="*caslib*"
specifies the caslib with the in-memory table to copy. If specified, this option overrides the INCASLIB= value in the procedure statement or the active caslib.

INDEXVARS={"*variable-name-1*" , "<*variable-name-2*>"}
specifies the list of variables to create indexes for in the output data.

OUTCASLIB="*caslib*"
specifies an alternative caslib to use for the copied table. If specified, this option overrides the OUTCASLIB= value in the procedure statement or the active caslib.

COPY Statement Options

COMPRESS
specifies to compress the data in the copied file.

COPIES=*integer*
specifies the number of replicate copies of the rows to make for fault tolerance. This value is ignored unless you save to the SASHDAT file format.

Alias REPLICATION=

GROUPBY=(*group-by-variable-1* <*group-by-variable-2* ...>)
specifies the names of the variables to use for partitioning the SASHDAT file.

Alias PARTITIONBY=

IMPORTOPTIONS=(FILETYPE="*file-type*" <*file-type-options*>)
specifies the input file format and options. Specify this option only if you want to read a file from the input caslib's data source and copy it without loading it to memory.

FILETYPE="BASESAS" | "CSV" | "DTA" | "EXCEL" | "HDAT" | "LASR" | "XLS"
specifies the file format.

file-type-options
specifies options for importing the data. For more information, see "Platform File Types" on page 153 and "Data Connectors" on page 167.

ORDERBY=(*variable-1 <variable-2 ...>*)
specifies the variables to use for ordering observations within partitions. This parameter applies to partitioned tables.

PROMOTE
specifies to load the table with global scope. This makes the table available to all sessions that use the caslib, subject to access controls. The caslib must also have global scope.

REPLACE
specifies that a new file with a given name replaces an existing file with the same name.

VARS=((*casinvardesc-1*) < (*casinvardesc-2*) ...>)
specifies the variables to load into the table. If you do not specify this option, then all variables are loaded into the table. For an example, see "Example 5: Copying and Saving a Subset of the Copied Table ".

The value can be one or more of the following:

FORMAT="*string*"
specifies the format to apply to the variable.

FORMATTEDLENGTH=*integer*
specifies the format field length plus the format precision length.

LABEL="*string*"
specifies the descriptive label for the variable.

NAME="*string*"
specifies the name for the variable.

NFD=*integer*
specifies the format precision length.

NFL=*integer*
specifies the format field length.

Alias VARLIST=

WHERE="*where-expression-1 <logical-operator where-expression-2>*"
The specification for this option is described in WHERE= on page 78 for the LOAD statement.

DELETESOURCE Statement

The DELETESOURCE statement removes a file from the data source that is associated with a caslib. You do not need to drop an in-memory table with the same name before using this statement.

Syntax

DELETESOURCE CASDATA="*file-name*" <INCASLIB="*caslib*" > <DATASOURCEOPTIONS = {*data-source-options*}> <DELETEOPTIONS = {*delete-options*}> <QUIET>;

Required Argument

CASDATA="*file-name*"
specifies the name of the file to remove.

Optional Arguments

INCASLIB="*caslib*"
specifies the caslib that is associated with the file to remove. If specified, this option overrides the INCASLIB= value in the procedure statement or the active caslib.

DATASOURCEOPTIONS={*data-source-options*}
specifies data source options.

Aliases dataSource

options

See dataSourceOptions

DELETEOPTIONS={*delete-options*}
specifies the settings for deleting a file from a data source.

Note The value that you specify for fileType determines the other parameters that apply.

See deleteOptions

QUIET=TRUE | FALSE
suppresses error messages and avoids setting the SYSERR automatic macro variable when the specified table or file is not found.

When QUIET is set to `True`, attempting to delete a file or table that does not exist returns an `OK` status and severity. When set to `False`, attempting to delete a file or table that does not exist returns an error.

Default FALSE

DROPTABLE Statement

The DROPTABLE statement removes a table from memory on SAS Cloud Analytic Services.

Example: "Example 4: Saving and Loading Encrypted SASHDAT Files" on page 99

Syntax

DROPTABLE CASDATA="*table-name*" <INCASLIB="*caslib*"> <QUIET>;

Required Argument

CASDATA="*table-name*"
specifies the name of the table to remove from memory.

Optional Arguments

INCASLIB="*caslib*"
specifies the caslib that is associated with the table to remove. If specified, this option overrides the INCASLIB= value in the procedure statement or the active caslib.

QUIET
suppresses error messages and avoids setting the SYSERR automatic macro variable when the specified table is not found.

INDEX Statement

The INDEX statement enables you to create indexes on one or more table variables.

Syntax

INDEX CASDATA="*table-name*"
<INCASLIB="*caslib*">
<VARS={ {*casinvardesc-1*}, {<*casinvardesc-n*>} }>
<WHERE="*where-expression* <*logical-operator where-expression*">>
<IMPORTOPTIONS=(FILETYPE=*file-type* <*file-type-options*>)>
<DATASOURCEOPTIONS=(*data-source-options*)>
<CASOUT="*table-name*">
<OUTCASLIB="*caslib*">
<LABEL="*label-name*">
<COMPRESS><PROMOTE><REPLACE><REPLICATION|COPIES=*integer*>
<INDEXVARS=((*variable-name-1*),<(*variable-name-n*)>)>

Required Argument

CASDATA="*table-name*"
specifies the name of the in-memory table to change.

Optional Arguments

INCASLIB="*caslib*"
specifies the caslib with the in-memory table to change. If specified, this option overrides the INCASLIB= value in the procedure statement or the active caslib.

VARS={ {*casinvardesc-1*}, {<*casinvardesc-n*>} }
specifies the variables to use in the action. The *casinvardesc-n* value can be one or more of the following:

format="*string*"
specifies the format to apply to the variable.

formattedLength=*integer*
specifies the length of format field plus the format precision.

label="*string*"
specifies the descriptive label for the variable.

name="*variable-name*"
specifies the name for the variable.

nfd=*integer*
specifies the length of the format precision.

nfl=*integer*
specifies the length of the format field.

WHERE="*where-expression*"
specifies an expression for subsetting the input data.

where-expression
is an arithmetic or logical expression that consists of a sequence of operators, operands, and SAS functions. An operand is a variable, a SAS function, or a constant. An operator is a symbol that requests a comparison, logical operation, or arithmetic calculation. The expression must be enclosed in single or double quotation marks.

logical-operator
can be AND, AND NOT, OR, or OR NOT.

IMPORTOPTIONS=(FILETYPE="*file-type*" <*file-type-options*>)
specifies the file format and options. If you do not specify the FILETYPE= option, the server attempts to determine the file type based on the file name suffix, such as .sashdat, .csv, and so on. Files with a .txt suffix are imported as a CSV file.

FILETYPE="BASESAS" | "CSV" | "DTA" | "EXCEL" | "HDAT" | "LASR" | "XLS"
specifies the file format.

file-type-options
specifies options for importing the data. For more information, see "Platform File Types" on page 153 and "Data Connectors" on page 167.

DATASOURCEOPTIONS=(*data-source-options*)
specifies overrides for the DATASOURCE= options for the input caslib. For more information, see "Platform Data Sources" on page 125 and "Data Connectors" on page 167.

Alias OPTIONS=

CASOUT="*table-name*"
specifies the name to use for the in-memory table.

Interaction This argument is required when you use the LOAD CASDATA= or LOAD FILE= forms.

Note This argument does not follow or enforce SAS naming rules such as the name literal syntax.

Tip Some data sources support table names that exceed 32 bytes. Use this option to limit table names to 32 bytes so that you can access the table with the CAS LIBNAME engine.

OUTCASLIB="*caslib*"
specifies an alternative caslib to use for the in-memory table. If specified, this option overrides the OUTCASLIB= value in the procedure statement or the active caslib.

LABEL="*label-name*"
specifies a descriptive label for the table. The label can be up to 256 characters. If the label text contains single quotation marks, enclose the label in double quotation marks. To remove a label from a table, assign a blank space that is enclosed in quotation marks.

COMPRESS
specifies to compress the output table. This option is supported with the DATA= argument only.

PROMOTE
specifies to load the table with global scope. This makes the table available to all sessions that use the caslib, subject to access controls. The caslib must also have global scope.

REPLACE
specifies that an in-memory table with a given name replaces an existing in-memory table with the same name.

REPLICATION|COPIES=*integer*
specifies the number of replicate copies of the rows to make for fault tolerance. Larger values use more memory and can result in slower performance, but provide high availability for data in the event of a node failure.

Default 1

Interaction This option is ignored if the data source is a data connector or data connect accelerator. The SAS Data Connector to Oracle and SAS Data Connector to Hadoop are examples of these products.

INDEXVARS={"*variable-name-1*" , "<*variable-name-2*>"}
specifies the list of variables to create indexes for in the output data.

Details

If the INDEX statement is used with no optional arguments, then all previous set indexes are cleared.

LIST Statement

The LIST statement lists files from a caslib's data source or in-memory tables in a caslib.

Example: "Example 1: Load a CSV File into CAS" on page 94

Syntax

LIST <FILES | TABLES> <*options*>;

Required Argument

FILES | TABLES
specifies whether to list the files from a caslib's data source or in-memory tables.

FILES lists the files that are available in the caslib's data source.

TABLES lists the in-memory tables in a caslib.

Default TABLES

Optional Arguments

INCASLIB="*caslib*"
specifies the caslib that is associated with the tables or files to list. If specified, this option overrides the INCASLIB= value in the procedure statement or the active caslib.

NOSUBDIRS
specifies to exclude subdirectories from the results.

Applies to LIST FILES and path-based caslibs

DATASOURCEOPTIONS=(*data-source-options*)
specifies overrides for the DATASOURCE= options for the caslib. For more information, see "Platform Data Sources" on page 125 and "Data Connectors" on page 167.

Alias OPTIONS=

Applies to LIST FILES

SUBDIR="*path*"
specifies to list the files in the specified subdirectory.

Applies to LIST FILES and path-based caslibs

LOAD Statement

The LOAD statement reads data from a file in a caslib's data source, a libref, or a client-side file and loads it into memory on SAS Cloud Analytic Services.

Syntax

Form 1: **LOAD** CASDATA="*file-name*" <INCASLIB="*caslib*"> CASOUT="*table-name*"

```
<OUTCASLIB="caslib">
  <IMPORTOPTIONS=(FILETYPE="file-type" <file-type-options>)>
  <GROUPBY=(group-by-variable-1 <group-by-variable-2 ...>)>
  <ORDERBY=(variable-1
  <variable-2 ...>)>
  <LABEL="table-description">
  <DATASOURCEOPTIONS=(data-source-options)>
  <PROMOTE | REPLACE>
  <COPIES=integer>
  <VARS=((casinvardesc-1) <, (casinvardesc-2), ...>)>
  <WHERE="where-expression-1 <logical-operator where-expression-2>">
  ;
```

Form 2: **LOAD** DATA=*SAS-data-set*<(*data-set-options*)>

```
<CASOUT="table-name">
  <OUTCASLIB="caslib">
  <APPEND | COMPRESS>
  <GROUPBY=(group-by-variable-1 <group-by-variable-2 ...>) >
  <ORDERBY=(variable-1
  <variable-2 ...>)>
  <LABEL="table-description">
  <DATASOURCEOPTIONS=(data-source-options)>
  <PROMOTE | REPLACE>
  <REPEAT>
  <COPIES=integer>
  <NCHARMULTIPLIER=n>
  <TRANSCODE_FAIL=ERROR | WARN | SILENT >
  <TIMEOUTSECONDS=n>
  <WRITETRANSFERSIZE=integer | integerK | integerM | integerG>;
```

Form 3: **LOAD** FILE="*path-to-client-file*" CASOUT="*table-name*"

```
<OUTCASLIB="caslib">
  <COMPRESS>
  <IMPORTOPTIONS=(FILETYPE=file-type <file-type-options>)>
```

<LABEL="*table-description*">
<PROMOTE | REPLACE>
<COPIES=*integer*>
<TIMEOUTSECONDS=*n*>
<WRITETRANSFERSIZE=*integer* | *integerK* | *integerM* | *integerG*>;

Form 4: **LOAD** FILE=*fileref* CASOUT="*table-name*"

<OUTCASLIB="*caslib*">
<COMPRESS>
<IMPORTOPTIONS=(FILETYPE=*file-type* <*file-type-options*>)>
<LABEL="*table-description*">
<PROMOTE | REPLACE>
<COPIES=*integer*>
<TIMEOUTSECONDS=*n*>
<WRITETRANSFERSIZE=*integer* | *integerK* | *integerM* | *integerG*>;

Required Arguments

CASDATA="*file-name*"
specifies the name of the file to load from the server-side data source that is associated with the INCASLIB= option or the active caslib.

Requirement You must specify CASOUT=.

CASOUT="*table-name*"
specifies the name to use for the in-memory table.

Interaction This argument is required when you use the LOAD CASDATA= or LOAD FILE= forms.

Note This argument does not follow or enforce SAS naming rules such as the name literal syntax.

Tip Some data sources support table names that exceed 32 bytes. Use this option to limit table names to 32 bytes so that you can access the table with the CAS LIBNAME engine.

DATA=*SAS-data-set*
specifies the libref and data set name to use.

FILE="*path-to-client-file*"
specifies an external file that is accessible to the SAS client host. Use this option to upload a file to the server and import the data. Do not use this option to import a SAS data set, use the DATA= option.

Requirement You must specify CASOUT=.

FILE=*fileref*
specifies an external file that is accessible with a file reference. Use this option to upload a file to the server and import the data. Do not use this option to import a SAS data set, use the DATA= option.

Requirement You must specify CASOUT=.

Tip Use the IMPORTOPTIONS= and FILETYPE= arguments to specify the file type. Otherwise, the server cannot determine the file type.

Optional Arguments

APPEND
adds the rows from the SAS data set in the DATA= argument to the end of an in-memory table. This option is supported with the DATA= argument only.

Note: You cannot specify these arguments simultaneously:

- REPLACE and APPEND
- APPEND and COMPRESS

COMPRESS
specifies to compress the output table. This option is supported with the DATA= argument only.

Note: You cannot specify these arguments simultaneously:

- APPEND and COMPRESS

COPIES=*integer*
specifies the number of replicate copies of the rows to make for fault tolerance. Larger values use more memory and can result in slower performance, but provide high availability for data in the event of a node failure.

Alias REPLICATION=

Default 1

Interaction This option is ignored if the REPEAT option is also specified.

DATASOURCEOPTIONS=(*data-source-options*)
specifies overrides for the DATASOURCE= options for the input caslib. For more information, see “Platform Data Sources” on page 125 and “Data Connectors” on page 167.

Alias OPTIONS=

GROUPBY=(*group-by-variable-1* <*group-by-variable-2* ...>)
specifies the names of the variables to use for grouping results.

Alias PARTITIONBY=

IMPORTOPTIONS=(FILETYPE="*file-type*" <*file-type-options*>)
specifies the file format and options. If you do not specify the FILETYPE= option, the server attempts to determine the file type based on the file name suffix, such as .sashdat, .csv, and so on. Files with a .txt suffix are imported as a CSV file.

FILETYPE="BASESAS" | "CSV" | "DTA" | "EXCEL" | "HDAT" | "LASR" | "XLS"
specifies the file format. For more available file formats, see the importOptions parameter for the "Load table" in *SAS Viya Platform: System Programming Guide*.

file-type-options
specifies options for importing the data. For more information, see "Platform File Types" on page 153 and "Data Connectors" on page 167.

INCASLIB="*caslib*"
specifies the caslib that is associated with the file to load. If specified, this option overrides the INCASLIB= value in the procedure statement or the active caslib.

LABEL="*string*"
specifies a descriptive label for the table. The label can be up to 256 characters. If the label text contains single quotation marks, enclose the label in double quotation marks. To remove a label from a table, assign a blank space that is enclosed in quotation marks.

NCHARMULTIPLIER=*n*
specifies the multiplication factor used when transcoding.

When SAS writes data to the CAS server, SAS estimates the number of bytes that are needed to transcode the data to UTF-8 based on the character set of the SAS session. When DBCS characters and some SBCS characters are transcoded to UTF-8, they require additional bytes to represent a character. You can use the NCHARMULTIPLIER= option to specify a byte multiplier when you know the number of bytes that are needed to represent the data in UTF-8.

CAUTION
Data truncation can occur during transcoding. If a data set is not wide enough to store the results that are encoded as UTF-8 when data transcoding occurs, then data might be truncated. To avoid truncation, use the NCHARMULTIPLER= option or the CASNCHARMULTIPLIER= system option to widen columns so that data is not truncated.

Defaults For SBCS environments the default is 1

For DBCS environments, the default is 1.5.

Range 0–4

Tip A best practice is to test reading data before the option is used in a production environment.

See "CASNCHARMULTIPLIER= System Option" in *SAS System Options: Reference*.

ORDERBY=(*variable-1* <*variable-2* ...>)
specifies the variables to use for ordering observations within partitions. This parameter applies to partitioned tables.

OUTCASLIB="*caslib*"
specifies an alternative caslib to use for the in-memory table. If specified, this option overrides the OUTCASLIB= value in the procedure statement or the active caslib.

PROMOTE
specifies to load the table with global scope. This makes the table available to all sessions that use the caslib, subject to access controls. The caslib must also have global scope.

Note: You cannot specify these arguments simultaneously:

- REPLACE and PROMOTE

REPEAT
specifies to duplicate the rows for the table on every machine of a distributed server. Making duplicate copies of tables can be useful in cases like a dimension table that is used in a join. This option is supported with the DATA= argument only and cannot be combined with the GROUPBY= option.

REPLACE
specifies that an in-memory table with a given name replaces an existing in-memory table with the same name.

Note: You cannot specify these arguments simultaneously:

- REPLACE and PROMOTE
- REPLACE and APPEND

TIMEOUTSECONDS=*n*
specifies the client inactivity time-out interval in seconds. If SAS stops transferring data to the server, the server starts the inactivity timer for the specified duration and cancels the request if SAS does not resume data transfer.

Alias TIMEOUT=

Default 1800 seconds

Range 30–3600 seconds

TRANSCODE_FAIL=ERROR | WARN | SILENT
specifies how the engine handles transcoding errors.

ERROR
writes an error message to the SAS log. Processing stops.

WARN
writes a warning message to the SAS log. Processing continues.

SILENT
transcoding errors are ignored. Processing continues.

Default ERROR

VARS=((*casinvardesc-1*) < (*casinvardesc-2*) ...>)
specifies the variables to load into the table. If you do not specify this option, then all variables are loaded into the table. For an example, see "Example 5: Copying and Saving a Subset of the Copied Table ".

The value can be one or more of the following:

FORMAT="*string*"
specifies the format to apply to the variable.

FORMATTEDLENGTH=*integer*
specifies the format field length plus the format precision length.

LABEL="*string*"
specifies the descriptive label for the variable.

NAME="*string*"
specifies the name for the variable.

NFD=*integer*
specifies the format precision length.

NFL=*integer*
specifies the format field length.

Alias VARLIST=

WHERE="*where-expression-1 <logical-operator where-expression-2>*"
specifies conditions for selecting observations from the data.

where-expression
is an arithmetic or logical expression that consists of a sequence of operators, operands, and SAS functions. An operand is a variable, a SAS function, or a constant. An operator is a symbol that requests a comparison, logical operation, or arithmetic calculation. The expression must be enclosed in single or double quotation marks.

logical-operator
can be AND, AND NOT, OR, or OR NOT.

Restriction You can specify the WHERE= as a data set option when you use the LOAD CASDATA= form.

WRITETRANSFERSIZE=*integer* | *integerK* | *integerM* | *integerG*
specifies the maximum number of bytes the procedure writes when transferring data to the CAS server.

integer
specifies the total number of bytes to read.

integerK
specifies the total number of kilobytes to read.

integerM
specifies the total number of megabytes to read.

integerG
specifies the total number of gigabytes to read.

Alias WTS=

Details

Summary of Options

Table 3.1 *Summary of LOAD Statement Options*

LOAD Statement Option	LOAD Statement Form
APPEND	LOAD DATA=
COMPRESS	LOAD DATA=
COPIES=	All LOAD statement forms. For more information, see "Data Redundancy" on page 125.
DATASOURCEOPTIONS=	LOAD CASDATA= and LOAD DATA=
GROUPBY=	LOAD CASDATA= and LOAD DATA=
IMPORTOPTIONS=	LOAD CASDATA= and LOAD FILE=
INCASLIB=	LOAD CASDATA=
LABEL=	All LOAD statement forms
NCHARMULTIPLIER=	LOAD DATA=
ORDERBY=	LOAD CASDATA= and LOAD DATA= **Note:** ORDERBY= requires GROUPBY=
OUTCASLIB=	All LOAD statement forms
PROMOTE	All LOAD statement forms
REPEAT	LOAD DATA=
REPLACE	All LOAD statement forms **Note:** Keep in mind that global-scope tables cannot be replaced. Use the DROPTABLE statement before the LOAD statement.
TIMEOUTSECONDS=	LOAD DATA= and LOAD FILE=

LOAD Statement Option	LOAD Statement Form
TRANSCODE_FAIL=	LOAD DATA=
VARS=	LOAD CASDATA=
WHERE=	LOAD CASDATA= **Note:** You can specify the WHERE= as a data set option when you use the LOAD CASDATA= form.
WRITETRANSFERSIZE=	LOAD DATA= and LOAD FILE= **Note:** When used with LOAD CASDATA= the WRITETRANSFERSIZE= option is ignored.

PARTITION Statement

The PARTITION statement enables you to partition a table.

Syntax

PARTITION CASDATA="*table-name*"
<INCASLIB="*caslib*">
<VARS={ {*casinvardesc-1*}, {<*casinvardesc-n*>} }>
<WHERE="*where-expression* <*logical-operator where-expression*">>
<PARTITIONBY | GROUPBY=(*group-by-variable-1* <*group-by-variable-n*>)>
<IMPORTOPTIONS=(FILETYPE=*file-type* <*file-type-options*>)>
<DATASOURCEOPTIONS=(*data-source-options*)>
<CASOUT="*table-name*">
<OUTCASLIB="*caslib*">
<LABEL="*label-name*">
<COMPRESS><PROMOTE><REPLACE><REPLICATION|COPIES=*integer*>
<INDEXVARS=((*variable-name-1*),<(*variable-name-n*)>)>

Required Argument

CASDATA="*table-name*"
specifies the name of the in-memory table to change.

Optional Arguments

INCASLIB="*caslib*"
specifies the caslib with the in-memory table to change. If specified, this option overrides the INCASLIB= value in the procedure statement or the active caslib.

VARS={ {*casinvardesc-1*},{<*casinvardesc-n*>} }
specifies the variables to use in the action. The *casinvardesc-n* value can be one or more of the following:

format="*string*"
specifies the format to apply to the variable.

formattedLength=*integer*
specifies the length of format field plus the format precision.

label="*string*"
specifies the descriptive label for the variable.

name="*variable-name*"
specifies the name for the variable.

nfd=*integer*
specifies the length of the format precision.

nfl=*integer*
specifies the length of the format field.

WHERE="*where-expression*"
specifies an expression for subsetting the input data.

where-expression
is an arithmetic or logical expression that consists of a sequence of operators, operands, and SAS functions. An operand is a variable, a SAS function, or a constant. An operator is a symbol that requests a comparison, logical operation, or arithmetic calculation. The expression must be enclosed in single or double quotation marks.

logical-operator
can be AND, AND NOT, OR, or OR NOT.

PARTITIONBY | GROUPBY=(*group-by-variable-1* <*group-by-variable-n*>)
specifies the names of the variables to use for grouping results.

IMPORTOPTIONS=(FILETYPE="*file-type*" <*file-type-options*>)
specifies the file format and options. If you do not specify the FILETYPE= option, the server attempts to determine the file type based on the file name suffix, such as .sashdat, .csv, and so on. Files with a .txt suffix are imported as a CSV file.

FILETYPE="BASESAS" | "CSV" | "DTA" | "EXCEL" | "HDAT" | "LASR" | "XLS"
specifies the file format.

file-type-options
specifies options for importing the data. For more information, see "Platform File Types" on page 153 and "Data Connectors" on page 167.

DATASOURCEOPTIONS=(*data-source-options*)
specifies overrides for the DATASOURCE= options for the input caslib. For more information, see "Platform Data Sources" on page 125 and "Data Connectors" on page 167.

Alias OPTIONS=

CASOUT="*table-name*"
specifies the name to use for the in-memory table.

Interaction This argument is required when you use the LOAD CASDATA= or LOAD FILE= forms.

Note This argument does not follow or enforce SAS naming rules such as the name literal syntax.

Tip Some data sources support table names that exceed 32 bytes. Use this option to limit table names to 32 bytes so that you can access the table with the CAS LIBNAME engine.

OUTCASLIB="*caslib*"
specifies an alternative caslib to use for the in-memory table. If specified, this option overrides the OUTCASLIB= value in the procedure statement or the active caslib.

LABEL="*label-name*"
specifies a descriptive label for the table. The label can be up to 256 characters. If the label text contains single quotation marks, enclose the label in double quotation marks. To remove a label from a table, assign a blank space that is enclosed in quotation marks.

COMPRESS
specifies to compress the output table. This option is supported with the DATA= argument only.

PROMOTE
specifies to load the table with global scope. This makes the table available to all sessions that use the caslib, subject to access controls. The caslib must also have global scope.

REPLACE
specifies that an in-memory table with a given name replaces an existing in-memory table with the same name.

REPLICATION|COPIES=*integer*
specifies the number of replicate copies of the rows to make for fault tolerance. Larger values use more memory and can result in slower performance, but provide high availability for data in the event of a node failure.

Default 1

Interaction This option is ignored if the data source is a data connector or data connect accelerator. The SAS Data Connector to Oracle and SAS Data Connector to Hadoop are examples of these products.

INDEXVARS={*"variable-name-1"* , *"<variable-name-2>"*}
specifies the list of variables to create indexes for in the output data

Details

If the PARTITION statement is used with no optional arguments, then all previous set partitions are cleared.

PROMOTE Statement

The PROMOTE statement copies a session-scope table to global scope.

Note: The PROMOTE statement does not include a REPLACE option. The server does not support promoting a session-scope table and replacing a global-scope table in one operation. You must drop the global-scope table first.

Example: "Example 3: Promote a Table" on page 98

Syntax

PROMOTE CASDATA=*"table-name"*
INCASLIB=*"caslib"*<CASOUT=*"table-name"*> <><OUTCASLIB=*"caslib"*> <DROP | KEEP>
<QUIET>;

Required Argument

CASDATA=*"table-name"*
specifies the name of the in-memory table to promote.

Optional Arguments

CASOUT=*"table-name"*
specifies the name to use for the promoted table.

Note This argument does not follow or enforce SAS naming rules such as the name literal syntax.

Tip Some data sources support table names that exceed 32 bytes. Use this option to limit table names to 32 bytes so that you can access the table with the CAS LIBNAME engine.

DROP
specifies to drop the session-scope table after promoting it to global scope, which is the default behavior and a best practice.

INCASLIB="*caslib*"
specifies the caslib with the in-memory table to promote. If specified, this option overrides the INCASLIB= value in the procedure statement or the active caslib.

KEEP
specifies to keep the session-scope table after promoting it to global scope. This workflow is uncommon. The table precedence rules are that the session-local table is accessed before a global-scope table is accessed.

OUTCASLIB="*caslib*"
specifies an alternative caslib to use for the promoted table. If specified, this option overrides the OUTCASLIB= value in the procedure statement or the active caslib.

QUIET
does not return a message with status and severity. If the QUIET option is not specified, then a message returns with status and severity.

Alias silent

SAVE Statement

The SAVE statement creates a permanent copy of an in-memory table. The in-memory table is saved to the data source that is associated with the caslib.

Example: "Example 4: Saving and Loading Encrypted SASHDAT Files" on page 99

Syntax

SAVE CASDATA="*table-name*" <INCASLIB="*caslib*">
<CASOUT="*file-name*"> <OUTCASLIB="*caslib*"> <EXPORTOPTIONS={*export-options*}> <IMPORTDATASOURCEOPTIONS={*import-data-source-options*}> <*options*>;

Required Argument

CASDATA="*table-name*"
specifies the name of the in-memory table to save.

Optional Arguments

CASOUT="*file-name*"
specifies an alternative name for the file. A file is created in the data source that is associated with the caslib from the OUTCASLIB= option. By default, a .sashdat file suffix is added when you save to a path-based caslib. If you specify a .csv suffix, a CSV file is saved.

Note This argument does not follow or enforce SAS naming rules such as the name literal syntax.

Tip Some data sources support table names that exceed 32 bytes. Use this option to limit table names to 32 bytes so that you can access the table with the CAS LIBNAME engine when you load it again.

EXPORTOPTIONS={*export-options*}
specifies the settings for saving a table to a data source.

Alias export

Note The value that you specify for fileType determines the other parameters that apply.

See exportOptions

IMPORTDATASOURCEOPTIONS={*import-data-source-options*}
enables you to save an in-memory table to the specified data source.

See dataSourceOptions

INCASLIB="*caslib*"
specifies the caslib with the in-memory table to save. If specified, this option overrides the INCASLIB= value in the procedure statement or the active caslib.

OUTCASLIB="*caslib*"
specifies an alternative caslib to use for the saved table. If specified, this option overrides the OUTCASLIB= value in the procedure statement or the active caslib.

SAVE Statement Options

COMPRESS
specifies to compress the data in the saved file.

COPIES=*integer*
specifies the number of replicate copies of the rows to make for fault tolerance. This value is ignored unless you save to the SASHDAT file format.

Alias REPLICATION=

DATASOURCEOPTIONS=(*data-source-options*)
specifies overrides for the DATASOURCE= options for the input caslib. For more information, see "Platform Data Sources" on page 125 and "Data Connectors" on page 167.

Alias OPTIONS=

GROUPBY=(*group-by-variable-1* <*group-by-variable-2* ...>)
specifies the names of the variables to use for partitioning the SASHDAT file.

Alias PARTITIONBY=

IMPORTOPTIONS=(FILETYPE="*file-type*" <*file-type-options*>)
specifies the input file format and options. Specify this option only if you want to read a file from the input caslib's data source and save it without loading it to memory.

FILETYPE="BASESAS" | "CSV" | "DTA" | "EXCEL" | "HDAT" | "LASR" | "XLS"
specifies the file format.

file-type-options
specifies options for importing the data. For more information, see "Platform File Types" on page 153 and "Data Connectors" on page 167.

ORDERBY=(*variable-1* <*variable-2* ...>)
specifies the variables to use for ordering observations within partitions. This parameter applies to partitioned tables.

REPLACE
specifies that a new file with a given name replaces an existing file with the same name.

WHERE="*where-expression-1* <*logical-operator where-expression-2*>"
The specification for this option is described in WHERE= on page 78 for the LOAD statement.

UPDATE Statement

The UPDATE statement enables you to update rows in a table.

Syntax

UPDATE CASDATA="*table-name*"

<INCASLIB="*caslib*">

<SET={ {var="*variable-name*", value="*string*"} }>

<WHERE="*where-expression* <*logical-operator where-expression*">>

Required Argument

CASDATA="*table-name*"
specifies the name of the in-memory table to change.

Optional Arguments

INCASLIB="*caslib*"
specifies the caslib with the in-memory table to change. If specified, this option overrides the INCASLIB= value in the procedure statement or the active caslib.

SET={ {var="*variable-name*", value="*string*"} }
specifies the variable name and the value to set. Use the where parameter to limit the updates to specific rows.

var="*variable-name*"
specifies the column name to update.

value="*string*"
specifies the value to set the column to. Enclose the value in quotation marks even if the data type is numeric.

WHERE="*where-expression*"
specifies an expression for subsetting the input data.

where-expression
is an arithmetic or logical expression that consists of a sequence of operators, operands, and SAS functions. An operand is a variable, a SAS function, or a constant. An operator is a symbol that requests a comparison, logical operation, or arithmetic calculation. The expression must be enclosed in single or double quotation marks.

logical-operator
can be AND, AND NOT, OR, or OR NOT.

Usage: CASUTIL Procedure

Enclose Values in Quotation Marks

When you specify a value for one of the following items, you can enclose the value in quotation marks:

- INCASLIB="*caslib*"
- OUTCASLIB="*caslib*"
- CASDATA="*table-name*"
- CASOUT="*table-name*"

For caslibs that use a case-sensitive file system or database as a data source, you control the case that is used.

If you do not enclose the INCASLIB= and OUTCASLIB= values in quotation marks, then SAS naming rules apply.

Subdirectories and File Name Matching

Exclude Subdirectories from the File Information Results

If a path-based caslib has many subdirectories, the results of the LIST FILES statement can be long and can obscure the list of files that can be loaded as data. The NOSUBDIRS option is available to exclude the subdirectories from the results.

```
cas casauto setsessopt=(caslib='casuser');

proc casutil;
   list files nosubdirs;
quit;
```

List the Files in a Subdirectory

This topic is the opposite of the preceding topic. You might have files in a subdirectory of a caslib's data source root that you want to use as data. For example, if your CASUSER personal caslib is assigned to **/home/sasdemo** and you have data in **/user/sasdemo/mydata**, you cannot assign a caslib to the **/home/sasdemo/mydata** path.

To list the files in **/home/sasdemo/mydata**, you can use the SUBDIR= option in the LIST FILES statement.

```
cas casauto setsessopt=(caslib='casuser');

proc casutil;
   list files subdir='mydata';
quit;
```

The CASUTIL Procedure

CAS File Information

Name	Permission	Owner	Group	File Size	Last Modified
iris.csv	-rwxr-xr-x	[redacted]	[redacted]	3.1KB	08Aug2016:19:38:59
titanic3.xls	-rwxr-xr-x	[redacted]	[redacted]	277.5KB	09Aug2016:09:01:48
jta_free_wifi.csv	-rwxr-xr-x	[redacted]	[redacted]	19.1MB	09Aug2016:09:03:39

Using Wildcard Characters for File Name Matching

Just as the NOSUBDIRS option can help limit the results of the LIST FILES statement, you can use wildcard characters to limit results. The following example lists the files with a CSV suffix.

```
cas casauto setsessopt=(caslib='casuser');

proc casutil;
   list files subdir='mydata/%.csv';
quit;
```

> **TIP** When you specify a wildcard pattern, the results include the matches and any subdirectories (matching or otherwise). To exclude subdirectories, you can specify the NOSUBDIRS option.

- File name matching is case sensitive. The **`%.csv`** pattern does not match files with an uppercase or mixed case suffix.
- `%` matches any number of characters or numbers.
- _ matches a single character or number. You can include more than one underscore in a pattern.
- \ escapes a wildcard character so that it is treated as a literal character instead of a wildcard in a pattern.

The SAS Macro language also uses the '%' character to indicate macro names to resolve during program compilation. When you use the '%' character for pattern matching, include the value in single quotations marks. If you enclose a value in double quotation marks, the SAS Macro parser tries to resolve a macro name. For example, the code **`SUBDIR="mydata/%s.csv"`** results in an attempt to resolve the %S macro name, with the message `WARNING: Apparent invocation of macro S not resolved`. Instead, specify the code as **`SUBDIR='mydata/%s.csv'`**.

Loading Files from a Subdirectory

After you list the files in a caslib's subdirectory, you can load the file into the server with the LOAD CASDATA= statement.

```
cas casauto setsessopt=(caslib="casuser");

proc casutil;
   load casdata="mydata/iris.csv" casout="iris";
quit;
```

Saving Files to a Subdirectory

By default, when you use the SAVE statement, a file is created in the directory that is associated with your active caslib, or the OUTCASLIB= option. To save a file in a subdirectory, include the path in the CASOUT= option.

```
cas casauto setsessopt=(caslib="casuser");

proc casutil;
   load casdata="mydata/iris.csv" casout="iris";
   save casdata="iris"
        where="species eq 'Setosa'"
        casout="mydata/setosa.csv"
        replace;

   list files subdir="mydata";
quit;
```

The CASUTIL Procedure

CAS File Information

Name	Permission	Owner	Group	File Size	Last Modified
iris.csv	-rwxr-xr-x	[redacted]	[redacted]	3.1KB	08Aug2016:19:38:59
titanic3.xls	-rwxr-xr-x	[redacted]	[redacted]	277.5KB	09Aug2016:09:01:48
jta_free_wifi.csv	-rwxr-xr-x	[redacted]	[redacted]	19.1MB	09Aug2016:09:03:39
setosa.csv	-rwxr-xr-x	[redacted]	[redacted]	0.9KB	09Aug2016:10:00:23

Limitations and Restrictions

The FORMAT statement in a PROC CASUTIL step applies to the LOAD DATA statements only. The FORMAT statement must be submitted prior to the LOAD DATA statements. Otherwise, it has no effect.

Results: CASUTIL Procedure

Procedure Output

The CONTENTS statement provides detailed information for an in-memory table. The following program generated the results with a distributed server with seven worker nodes.

```
proc casutil;
   load data=sashelp.iris;
   contents casdata="iris";
quit;
```

Table Information for Caslib CASUSER(sasdemo)

Table Name	Label	Number of Rows	Number of Columns	Indexed Columns	NLS encoding	Created	Last Modified	Promoted Table	Repeated Table	View	Compressed
IRIS	Fisher's Iris Data (1936)	150	5	0	utf-8	2019-09-19T13:27:38-04:00	2019-09-19T13:27:38-04:00	No	No	No	No

Detail Information for iris in Caslib CASUSER(sasdemo).

Node	Number of Blocks	Active Blocks	Rows	Data size	Variable Data size	Blocks Mapped	Memory Mapped	Blocks Unmapped	Memory Unmapped	Blocks Allocated	Memory Allocated	Index Size	Compressed Size	Compression Ratio
ALL	4	2	150	7200	0	2	7552	2	7552	0	0	0	0	0

Column Information for IRIS in Caslib CASUSER(sasdemo)

Column	Label	Type	Length	Formatted Length	Format Width	Format Decimal
Species	Iris Species	char	10	10	0	0
SepalLength	Sepal Length (mm)	double	8	12	0	0
SepalWidth	Sepal Width (mm)	double	8	12	0	0
PetalLength	Petal Length (mm)	double	8	12	0	0
PetalWidth	Petal Width (mm)	double	8	12	0	0

The Promoted Table field indicates when a table has global scope. Yes indicates that the table is a global-scope table. No indicates that the table is a session-scope table. For information about scope and repeated tables, see "More about Tables" in *SAS Cloud Analytic Services: Fundamentals*. If you load a table with the LOAD CASDATA= form, then the Source Name field indicates the original file name and the Source Caslib field indicates the original caslib.

For the table details information, see these definitions:

Node
: This field always reports ALL. This procedure provides a summary of the table information for all machines in a distributed server.

Number of Blocks
: The server organizes rows in blocks. For distributed servers, this column shows the sum of the active blocks and any copies of blocks that provide redundancy. Single-machine servers do not use redundant data blocks,

Active Blocks
: The server reads rows from active blocks.

Fixed Data size
: This field shows the number of bytes that are used for numeric columns and fixed-width character columns.

Variable Data size
: This field shows the number of bytes that are used for varying-length character columns (VARCHAR) and varying-length binary columns (VARBINARY). When these data types are used in a table, each row includes a 16-byte descriptor for each varying-length column.

Blocks Mapped
: This field shows the number of blocks that are currently mapped into memory. For small tables, smaller than 16M, this value can be 0. A zero indicates that the data is in memory only. A nonzero value indicates that some data is either using blocks from a SASHDAT file or that data is temporarily stored in CAS_DISK_CACHE.

  Note: The 16M value is the default value for the MAXTABLEMEM= session option. For more information, see the MAXTABLEMEM= on page 679 session option.

Memory Mapped
: This field shows the number of bytes for the blocks that are mapped. The blocks are mapped from temporary files in CAS_DISK_CACHE or from a SASHDAT file.

  If the table is from a SASHDAT file that is encrypted or is subset with a WHERE filter, then blocks from the original SASHDAT file are not used. In these cases, temporary files in CAS_DISK_CACHE are used.

Blocks Unmapped
: This field shows the number of blocks that the server can map into memory. The blocks are mapped into memory when the server handles a request for data from the table. For distributed servers, the redundant blocks that enable fault tolerance are included in this value.

Memory Unmapped
: This field shows the number of bytes for the blocks that the server can map into memory.

Blocks Allocated
: This field shows the number of blocks that are allocated in memory and do not have an on-disk representation. The blocks can become cached under the following conditions:

  - when you promote a session-scope table to global-scope
  - when you append or update rows

Memory Allocated
: This field shows the number of bytes for the blocks.

Index Size
: This field shows the number of bytes that are used for column indexes.

Compressed Size
: This field shows the number of bytes that are used when the table is compressed. You can use compression by specifying the COMPRESS option when you use one of the forms of the LOAD statement.

Compression Ratio
: This field shows the ratio of the Fixed Data Size value when it is divided by the Compressed Size value. If the remainder from the division is greater than half of the compressed size, then the value is rounded up by 1.

There are two ways for the server to have on-disk blocks that correspond to the Blocks Mapped and Blocks Unmapped values:

- The server can create blocks in a directory that is used for caching. The directories are specified at deployment time by an administrator and corresponds to the CAS_DISK_CACHE environment variable.
- For distributed servers that use a DNFS caslib, the blocks correspond to the blocks of a SASHDAT file.

ODS Table Names

PROC CASUTIL assigns a name to each table that it creates.

Table 3.2 *ODS Tables Produced by the CASUTIL Procedure*

ODS Table	Description	Statement Used
CaslibInfo	Caslib information	LIST
ColumnInfo	Column information	CONTENTS
FileInfo	CAS file information	LIST with the FILES option
TableDetails	Detailed table information	CONTENTS
TableInfo	CAS table information	LIST with the TABLES option, CONTENTS

Examples: CASUTIL Procedure

Example 1: Load a CSV File into CAS

Program

```
caslib csvfiles task=add type=dnfs                          /* 1 */
   path="/data/csv/"
   desc="Spreadsheets and CSV source data.";

proc casutil;
   list files;

   load casdata="County_Population.csv"                     /* 2 */
      importoptions=(filetype="csv" getnames="true")
      casout="county_population";

   list tables;
quit;
```

1 The TYPE=DNFS option to the CASLIB statement specifies a distributed NFS caslib type. This type requires every machine that is used for the server to have network access to the specified path. The CASLIB statement also sets the active caslib. The caslib Csvfiles is the input and output caslib.

2 The IMPORTOPTIONS= option is used to specify the file type and options for reading the data into the server.

Results: LIST FILES Statement for the CSVFILES Caslib

The following display shows the results of the LIST FILES statement. It is a listing of the files that the server can access from the **/data/csv** directory.

The CASUTIL Procedure

File Information for root of caslib CSVFILES.

Name	Permission	Owner	Group	File Size	Last Modified
Spreadsheet10k.xlsx	-rwxrwxrwx	nfsnobody	nfsnobody	563.5KB	08Nov2012:07:21:12
mailorder.csv	-rwxrwxrwx	nfsnobody	nfsnobody	117.2KB	29Sep2015:11:40:14
NST_EST2012_ALLDATA.csv	-rwxrwxrwx	nfsnobody	nfsnobody	20.0KB	15Jan2013:08:59:28
sine.csv	-rwxrwxrwx	nfsnobody	nfsnobody	309.3KB	15Jan2013:13:18:14
SC-EST2011-6RACE-ALL.csv	-rwxrwxrwx	nfsnobody	nfsnobody	138.3KB	15Jan2013:09:02:53
sine10k.csv	-rwxrwxrwx	nfsnobody	nfsnobody	582.9KB	09May2013:07:16:16
UNdata_Export_20130115_092219649.csv	-rwxrwxrwx	nfsnobody	nfsnobody	595.4KB	15Jan2013:09:22:20
2012_nfl_pbp_data_reg_season1.csv	-rwxrwxrwx	nfsnobody	nfsnobody	6.5MB	06Feb2013:11:19:02
County_Population.csv	-rwxrwxrwx	nfsnobody	nfsnobody	1.3MB	14Jan2013:14:15:41

Results: LIST TABLES Statement for the CSVFILES Caslib

The following display shows the results of the LIST TABLES statement. It shows that the County_Population table is the only in-memory table in the caslib.

The CASUTIL Procedure

Caslib Information	
Library	CSVFILES
Source Type	DNFS
Description	"Spreadsheets and CSV source data."
Path	[redacted]
Session local	Yes
Active	Yes
Personal	No

The SAS System

The CASUTIL Procedure

Table Information for Caslib CSVFILES

Table Name	Number of Rows	Number of Columns	NLS encoding	Created	Last Modified	Promoted Table	Repeated Table	View	Source Name	Source Caslib	Compressed
COUNTY_POPULATION	3193	97	utf-8	08Apr2016:10:41:18	08Apr2016:10:41:18	No	No	No	County_Population.csv	CSVFILES	No

Example 2: Append Rows to an In-Memory Table

Program

```
proc casutil;
   load
data=sashelp.cars(where=(make="Buick"))                              /*
1 */
        incaslib="casuser"
```

```
        casout="some_cars"
        outcaslib="casuser"
        label="Some makes from the Sashelp.Cars sample data."
        promote;

   /* add rows for a few more makes */
   load data=sashelp.cars(where=(make in ("Ford", "Chrysler")))
        incaslib="casuser"
        casout="some_cars"
        outcaslib="casuser"

append;                                                          /*
2 */

   list tables;
quit;

libname mycas cas;

proc cardinality data=mycas.some_cars outcard=mycas.cars_cardinality;
   vars enginesize mpg_highway
mpg_city;                                    /* 3 */
run;

proc casutil;
   contents casdata="cars_cardinality"
incaslib="casuser";                  /* 4 */
run;

proc print data=mycas.cars_cardinality;
   var _varname_ _cardinality_ _nobs_ _nmiss_ _min_--_kurtosis_;
run;
```

1 The first LOAD DATA= statement subsets the Sashelp.Cars data set based on the Make variable. The CASOUT= option specifies the name for the output table, Some_cars. The PROMOTE option sets the output table as a global-scope table.

2 The second LOAD statement uses the APPEND option to append more rows from the Sashelp.Cars data set.

3 The CARDINALITY procedure is used to calculate summary statistics for three variables. The OUTCARD= option specifies an in-memory table to use for storing the summary data.

4 The CONTENTS statement is used to display the table information and column information for the Cars_cardinality table. Notice that the libref is not included with the CASDATA= option. The results of the statement include the column names. Some of the column names are specified in the PRINT procedure.

Results: LIST TABLES Statement for the CASUSER Caslib

The CASUTIL Procedure

Table Information for Caslib [redacted]

Table Name	Label	Number of Rows	Number of Columns	NLS encoding	Created	Last Modified	Promoted Table	Repeated Table	View	Compressed
SOME_CARS	Some makes from the Sashelp.Cars sample data.	47	15	utf-8	08Apr2016:13:27:37	08Apr2016:13:27:37	Yes	No	No	No

Results: CONTENTS Statement for the Cars_Cardinality Table

The CASUTIL Procedure

Table Information for Caslib [redacted]

Table Name	Number of Rows	Number of Columns	NLS encoding	Created	Last Modified	Promoted Table	Repeated Table	View	Compressed
CARS_CARDINALITY	3	26	utf-8	08Apr2016:13:29:19	08Apr2016:13:29:19	No	Yes	No	No

Detail Information for cars_cardinality in Caslib [redacted].

Node	Number of Blocks	Active Blocks	Rows	Fixed Data size	Variable Data size	Blocks Mapped	Memory Mapped	Blocks Unmapped	Memory Unmapped	Blocks Allocated	Memory Allocated
ALL	7	7	21	7560	0	0	0	7	8792	0	0

Column Information for CARS_CARDINALITY in Caslib [redacted]

Column	Label	Type	Length	Formatted Length	Format
VARNAME	Variable name	char	32	32	$
FMTWIDTH	Width of the variable formatted value	double	8	12	BEST
TYPE	Type of the raw values	char	1	1	$

Results: Select Columns from the Cars_Cardinality Table

Obs	_VARNAME_	_CARDINALITY_	_NOBS_	_NMISS_	_MIN_	_MAX_	_MEAN_	_STDDEV_	_SKEWNESS_	_KURTOSIS_
1	EngineSize	17	47	0	2	6.8	3.3957446809	1.0133795046	0.777871628	1.3478938265
2	MPG_Highway	15	47	0	13	36	26.638297872	4.3410354	-0.725567937	1.3662855198
3	MPG_City	12	47	0	10	27	19.382978723	3.4550101711	0.1929445364	0.6283786957

Example 3: Promote a Table

Program

```
caslib hps datasource=(srctype="path") path="/hps" global;

cas casauto sessopts=(caslib="casuser");

libname mycas cas;

proc casutil;
   load data=sashelp.iris casout="irisraw";
quit;

data mycas.irisout;                                    /* 1 */
   set mycas.irisraw;
   sepalratio = sepalwidth / sepallength;
   petalratio = petalwidth / petallength;
run;

/*
 * The purpose for outcaslib= is to show how to
 * work with more than one caslib.
 */
proc casutil outcaslib="hps";                          /* 2 */
   promote casdata="irisout";
quit;

proc casutil incaslib="hps";
   contents casdata="irisout";
quit;
```

1 The DATA step creates an output table that is named Irisout from an input table named Irisraw. Two columns are added to the table.

2 The OUTCASLIB= option is used to demonstrate how to work with more than one caslib. If you specify OUTCASLIB= when you promote or load a table, then you need to use INCASLIB= with the same name to access the table again.Notice that in the CASDATA= option in the PROMOTE statement on page 83 that follows, the table name is specified without the libref.

Results: CONTENTS Statement for the Irisout Table

The following graphic shows the results of the CONTENTS statement. The Table Information results show that the caslib is HPS and that the table is promoted to global scope. The Column Information results show the two columns that were added with the DATA step.

The CASUTIL Procedure

Table Information for Caslib HPS

Table Name	Number of Rows	Number of Columns	NLS encoding	Created	Last Modified	Promoted Table	Repeated Table	View	Compressed
IRISOUT	150	7	utf-8	08Apr2016:13:47:40	08Apr2016:13:47:40	Yes	No	No	No

Detail Information for irisout in Caslib HPS.

Node	Number of Blocks	Active Blocks	Rows	Fixed Data size	Variable Data size	Blocks Mapped	Memory Mapped	Blocks Unmapped	Memory Unmapped	Blocks Allocated	Memory Allocated
ALL	14	7	150	9600	0	0	0	14	19200	0	0

Column Information for IRISOUT in Caslib HPS

Column	Label	Type	Length	Formatted Length
Species	Iris Species	char	10	10
SepalLength	Sepal Length (mm)	double	8	12
SepalWidth	Sepal Width (mm)	double	8	12
PetalLength	Petal Length (mm)	double	8	12
PetalWidth	Petal Width (mm)	double	8	12
sepalratio		double	8	12
petalratio		double	8	12

Example 4: Saving and Loading Encrypted SASHDAT Files

Program

The CSV file that is used in this example was downloaded on 10FEB2016 from https://www.hokoukukan.go.jp/download/jta_free_wifi.csv. Your results for the count of WiFi access points by category could be different.

```
options validmemname=extend validvarname=any;                    /* 1
*/

options caslib="casuser";
libname mycas cas;

proc casutil incaslib="casuser" outcaslib="casuser";
   load casdata="jta_free_wifi.csv"
        importoptions=(filetype="csv" getnames="yes"
           encoding="sjis")                                      /* 2 */
        casout="jta_free_wifi";

   contents casdata="jta_free_wifi";                             /* 3
*/

   save casdata="jta_free_wifi"
```

```
        datasourceoptions=(encryptionPassword='changeit');

   droptable casdata="jta_free_wifi";                              /* 4
*/

   load casdata="jta_free_wifi.sashdat"                            /* 5
*/
        importoptions=(filetype="hdat" encryptionPassword='changeit')
        casout="jta_free_wifi2";
quit;

proc mdsummary data=mycas.jta_free_wifi2;                           /*
6  */
  var スポット ID;
  groupby カテゴリー / out=mycas.category;
run;

proc print data=mycas.category;                                     /* 7
*/
  var カテゴリー _nobs_;
run;
```

1 This example uses data with column names that do not follow SAS naming conventions. These options provide greater flexibility with table names and column names.

2 The CSV file uses the Shift JIS encoding. If a file does not use UTF-8 or 7-bit ASCII, then specify the encoding.

3 The CONTENTS statement on page 64 displays the table information, table details, and column information. It is included in the example as a reminder that you should confirm that the column names from the file are imported as you expect them to be imported. See "Results: CONTENTS Statement for the Jta_free_wifi Table" on page 101.

4 The DROPTABLE statement is not necessary in most programs. It is included in this example so that the subsequent LOAD CASDATA= statement succeeds without the REPLACE option.

5 The LOAD CASDATA= statement includes the encryption password. Notice also that you do not need to specify an ENCODING= option. The SAVE statement stored the data as UTF-8 when it created the SASHDAT file.

6 The MDSUMMARY procedure is included to show that after the table is loaded into memory, then you can use a CAS engine libref to access the in-memory table. The goal for this example is to identify the different categories of hotspots and the counts. Only one variable is summarized, the WiFi hotspot identifier, and the variable is grouped by values of the hotspot category. The summary is written to an in-memory table on the server that is named Category.

7 The PRINT procedure is used to read the summarized results from the in-memory table on the server. The VAR statement limits the display to the different hotspot categories and the count for each category. See Output 3.4 on page 101.

Results: CONTENTS Statement for the Jta_free_wifi Table

The CASUTIL Procedure

Table Information for Caslib CASUSER([redacted])

Table Name	Number of Rows	Number of Columns	NLS encoding	Created	Last Modified	Promoted Table	Repeated Table	View	Source Name	Source Caslib	Compressed
JTA_FREE_WIFI	42259	21	utf-8	08Apr2016:13:56:01	08Apr2016:13:56:01	No	No	No	jta_free_wifi.csv	CASUSER([redacted])	No

Detail Information for jta_free_wifi in Caslib CASUSER([redacted]).

Node	Number of Blocks	Active Blocks	Rows	Fixed Data size	Variable Data size	Blocks Mapped	Memory Mapped	Blocks Unmapped	Memory Unmapped	Blocks Allocated	Memory Allocated
ALL	84	42	42259	32730247	19883511	0	0	84	65471024	0	0

Column Information for JTA_FREE_WIFI in Caslib CASUSER([redacted])

Column	Type	Length	Formatted Length
スポットID	double	8	12
スポット名（日本語）	varchar	115	55
スポット名（英語）	varchar	255	255
スポットステータス	varchar	20	8

Results: WiFi Hotspot Categories and Counts

Output 3.1 *WiFi Hotspot Categories and Counts*

Obs	カテゴリー	_NObs_
1	その他	31989
2	バス	2
3	ホテル	1741
4	商業施設(百貨店、SC、アウトレットモール等)	46
5	港湾	5
6	移動中の休憩所(サービスエリア、道の駅等)	181
7	空港	6
8	美術館・博物館・寺社仏閣	11
9	観光スポット（景勝地等）	33
10	観光案内所	91
11	鉄道(駅構内)	175
12	飲食・小売店	7979

Example 5: Copying and Saving a Subset of the Copied Table

Program

Use the COPY statement to copy a table and save a subset of the copy to your caslib.

```
cas casauto sessopts=(caslib="casuser");

libname mycas cas;

proc casutil;
   load data=sashelp.cars casout="cars";
quit;

proc casutil;
   copy
casdata="cars"                                                    /
*1*/

casout="carsCopyWhereVars"
     /*2*/
      vars=((Name="Make") (Name="Origin") (Name="MSRP",
Label="Manufacturer's Suggested Retail Price")) /*3*/

where="MSRP>90000"
 /*4*/

replace;
 /*5*/
quit;
```

1 The CASDATA= argument specifies CARS as the name of the table to copy.

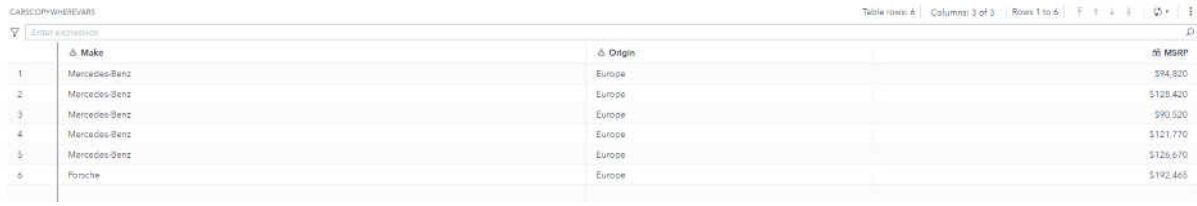
2 The CASOUT= argument specifies the alternative name for the copied file.

3 The VARS option specifies the variables that are loaded into the new table. If you do not specify this option, then all variables are loaded into the table.

4 The WHERE option specifies the condition MSRP greater than 90,000. This condition copies only those rows where the condition is met and the rows are copied into CarsCopyWhere table.

5 The REPLACE option specifies that CarsCopyWhere replaces an existing file with the same name. If you previously created a table with the name CarsCopyWhere and it was still available in your current session, the REPLACE option overwrites the existing CarsCopyWhere table with the current copy.

Output 3.2 *CarsCopyWhere Table*



	Make	Origin	MSRP
1	Mercedes-Benz	Europe	$94,820
2	Mercedes-Benz	Europe	$128,420
3	Mercedes-Benz	Europe	$90,520
4	Mercedes-Benz	Europe	$121,770
5	Mercedes-Benz	Europe	$126,670
6	Porsche	Europe	$192,465

Example 6: Rename a Table and Rename Columns

Program

Use the ALTERTABLE statement to rename the table and rename the columns in the table.

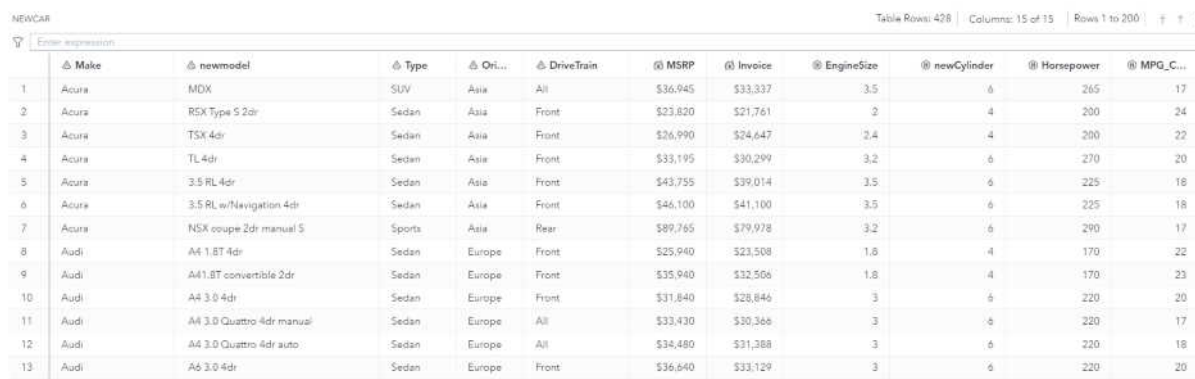
```
cas casauto sessopts=(caslib="casuser");

libname mycas cas;

proc casutil;
   load data=sashelp.cars casout="cars";

   altertable casdata="cars"
      rename = "newcar"
      columns =
        {
          {name= "Cylinders" rename="newCylinder"}
          {name="model" rename="newmodel"}
        };
quit;
```

Output 3.3 *NewCar Table*



	Make	newmodel	Type	Ori...	DriveTrain	MSRP	Invoice	EngineSize	newCylinder	Horsepower	MPG_C...
1	Acura	MDX	SUV	Asia	All	$36,945	$33,337	3.5	6	265	17
2	Acura	RSX Type S 2dr	Sedan	Asia	Front	$23,820	$21,761	2	4	200	24
3	Acura	TSX 4dr	Sedan	Asia	Front	$26,990	$24,647	2.4	4	200	22
4	Acura	TL 4dr	Sedan	Asia	Front	$33,195	$30,299	3.2	6	270	20
5	Acura	3.5 RL 4dr	Sedan	Asia	Front	$43,755	$39,014	3.5	6	225	18
6	Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	Front	$46,100	$41,100	3.5	6	225	18
7	Acura	NSX coupe 2dr manual S	Sports	Asia	Rear	$89,765	$79,978	3.2	6	290	17
8	Audi	A4 1.8T 4dr	Sedan	Europe	Front	$25,940	$23,508	1.8	4	170	22
9	Audi	A4 1.8T convertible 2dr	Sedan	Europe	Front	$35,940	$32,506	1.8	4	170	23
10	Audi	A4 3.0 4dr	Sedan	Europe	Front	$31,840	$28,846	3	6	220	20
11	Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	All	$33,430	$30,366	3	6	220	17
12	Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	All	$34,480	$31,388	3	6	220	18
13	Audi	A6 3.0 4dr	Sedan	Europe	Front	$36,640	$33,129	3	6	220	20

4

MDSUMMARY Procedure

Overview: MDSUMMARY Procedure ***105***
What Does the MDSUMMARY Procedure Do? 105

Syntax: MDSUMMARY Procedure ***106***
PROC MDSUMMARY Statement 106
VAR Statement 107
GROUPBY Statement 107
OUTPUT Statement 108

Usage: MDSUMMARY Procedure ***109***
PROC MDSUMMARY Output Data Sets 109

Results: MDSUMMARY Procedure ***109***
Output Tables 109

Examples: MDSUMMARY Procedure ***111***
Example 1: Compute Descriptive Statistics 111
Example 2: Computing Descriptive Statistics with Group-By Variables 113
Example 3: Using Multiple Group-By Variables 117
Example 4: Using Formats with Group-By Variables 119
Example 5: Graph Summary Statistics Results Obtained from the MDSUMMARY Procedure 122

Overview: MDSUMMARY Procedure

What Does the MDSUMMARY Procedure Do?

The MDSUMMARY procedure computes basic descriptive statistics for variables across all observations or within groups of observations in parallel for data tables

stored in SAS Cloud Analytic Services (CAS). The MDSUMMARY procedure uses CAS tables and capabilities, ensuring full use of parallel processing.

Syntax: MDSUMMARY Procedure

Restrictions: You cannot use a FORMAT statement in a PROC MDSUMMARY step. If you need to format the input table, apply the formats to the table when it is created. For examples, see "Example 4: Using Formats with Group-By Variables" on page 119 and "Add and Save a User-Defined Format" in *SAS Cloud Analytic Services: User-Defined Formats*.

You cannot use a BY statement in a PROC MDSUMMARY step. Use the MDSUMMARY procedure GROUPBY statement on page 107 instead.

PROC MDSUMMARY DATA=*table-name*< NTHREADS=*integer*>;

VAR <*variable-list*>;

OUTPUT OUT=*libref.table-name*;

GROUPBY *variable-list* </ OUT=*libref.table-name*>;

RUN;

PROC MDSUMMARY Statement

Calculates multidimensional summaries of numeric variables.

Syntax

PROC MDSUMMARY DATA=*libref.table-name*< NTHREADS=*integer*>;

Optional Arguments

DATA=*libref.table-name*
specifies the two-level input table name.

Alias TABLE=

Requirement The table name must be a two-level name where *Libref* is a CAS engine libref.

NTHREADS=*integer*
specifies the number of threads to use within each compute node.

VAR Statement

Specifies the analysis variables and their order in the output.

Syntax

VAR <*variable-list*>

Without Arguments

If no variables are listed, then the summary statistics are computed for all numeric variables.

Optional Argument

<variable(s)>
identifies one or more analysis variables and specifies their order in the results.

GROUPBY Statement

Creates BY groups in terms of the variable value combinations given the variables in the variable list.

Requirement: You must specify either the OUTPUT statement, or at least one GROUPBY statement with the OUT= option specified. You cannot specify both the OUTPUT= statement and a GROUPBY statement with the OUT= option specified.

Note: You cannot use a BY statement in an MDSUMMARY procedure step. Use a GROUPBY statement instead.

Tips: Multiple GROUPBY statements can be specified, each having its own output table.

If a variable value is missing, PROC MDSUMMARY includes the observations and rows in the analysis.

Syntax

GROUPBY <*variable(s)*> </ OUT=*libref.table-name*>;

Without Arguments

If no variables are listed, then the statistics are calculated across all input observations

Optional Arguments

OUT=*libref.table-name*
specifies the table name.

Requirement The table name must be a two-level name where *Libref* is a CAS engine libref.

variable(s)
specifies the analysis variables to group by.

OUTPUT Statement

Creates an output table that contains the results of PROC MDSUMMARY.

Restriction: You can specify one OUTPUT statement only.

Requirement: You must specify either the OUTPUT statement or at least one GROUPBY statement that specifies the OUT= option. You cannot specify the OUTPUT= statement with one or more GROUPBY statements that also specify the OUT= option.

Syntax

OUTPUT OUT=*libref.table-name*;

Required Argument

OUT=*libref.table-name*
specifies the two-level table name.

Requirement The table name must be a two-level name where *Libref* is a CAS engine libref.

Usage: MDSUMMARY Procedure

PROC MDSUMMARY Output Data Sets

You can create output tables by using one OUTPUT statement or multiple GROUPBY statements with the OUT= option specified. You must specify either the OUTPUT statement, or one or more GROUPBY statements with the OUT= option specified. You cannot specify both an OUTPUT statement and a GROUPBY statement with the OUT= option specified. To produce multiple output tables, use multiple GROUPBY OUT= statements.

PROC MDSUMMARY does not display output. You can use PROC PRINT to display the output table.

Results: MDSUMMARY Procedure

Output Tables

PROC MDSUMMARY produces one or more output tables for each BY group, defined by a set of variables listed in a GROUPBY statement.

A PROC MDSUMMARY table contains the following:

- One column for each basic statistic and one row for each combination of group-by level and analysis variable.
- If you are creating BY groups, then two columns for each group-by variable are also included in the output table. One column is for the group-by variable itself. The other column is of a character type and has the same name as the group-by variable but with _f appended. The column contains the formatted value of the group-by variable.
- A column named _Column_, containing the name of the analysis variable, is included in the output. The _Column_ column denotes the variable for which statistics have been computed.

Group-by processing collects observations for analysis according to the formatted values of the group-by variables, with each unique combination of formatted group-by variable values forming one group-by level. Groups are not collected or processed in any particular order.

The statistics produced by MDSUMMARY are not configurable but are fixed and include:

Table 4.1 *Table of Statistic Produced by the MDSUMMARY Procedure*

Column Name	Statistic	Description
CSS	CSS	Corrected sum of squares
_CV_	CV	Coefficient of variation
MAX	MAX	Maximum value
MEAN	MEAN	Arithmetic mean
MIN	MIN	Minimum value
NMISS	NMISS	Number of values that are missing
NOBS	NOBS	Total number of observations
PRT	PRT	p-Value for Student's t statistics
STD	STD	Standard deviation
STDERR	STDERR	Standard error of the mean
SUM	SUM	Sum
_T_	T	Student's t statistic
USS	USS	Uncorrected sum of squares
VAR	VAR	Variance

Examples: MDSUMMARY Procedure

Example 1: Compute Descriptive Statistics

Program

The following example shows you how to access your data with SAS Cloud Analytic Services (CAS). It shows you how to compute all statistics for each variable, and treat the entire input table as one group. The results are written to an output table.

```
/* options cashost="cloud.example.com" casport=5570; */
cas casauto;                          /* 1 */

libname mycas cas;                    /* 2 */

proc casutil;                         /* 3 */
   load data=sashelp.cars replace;
   contents casdata="cars";
quit;

proc mdsummary data=mycas.cars;       /* 4 */
   var MSRP MPG_City;
   output out=mycas.mdsumstat;
run;

proc print data=mycas.mdsumstat;      /* 5 */
   var _Column_ _NObs_ _Mean_ _Max_ _Min_ _Std_  ;
   title 'Summary of MSRP and City Miles Per Gallon';
run;

proc casutil;                         /* 6 */
   save casdata="mdsumstat" replace;
quit;
```

1 The CAS statement creates session Casauto.

2 The LIBNAME statement creates a CAS engine libref.

Note: To run PROC MDSUMMARY and PROC PRINT in CAS, you must specify the CAS engine LIBNAME statement and use the CAS engine libref with both the input and output table names.

3 The CASUTIL procedure loads the data. The LOAD DATA= statement reads the file into memory. The table is now available for analytics. The CONTENTS statement reads the on-disk file, Cars, and displays the table metadata. This enables you to learn if the file has column names in the first row and the data types.

4 The MDSUMMARY procedure produces summary statistics for MPG_CITY and MSRP. The OUTPUT statement creates an in-memory table named Mycas.MdsumStat. The table includes a row of summary statistics for MSRP and MPG_City.

5 The MDSUMMARY procedure does not print the output. Use the PRINT procedure to print the table Mycas.MdsumStat.

6 The CASUTIL procedure saves the table to the default caslib. The SAVE statement stores the data as UTF-8 when it creates the SASHDAT file.

Results

Output 4.1 *PROC CASUTIL Output: Column Information for the Cars Data Set*

Column Information for CARS in Caslib MYCASLIB							
Column	Label	Type	Length	Format Name	Formatted Length	Format Width	Format Decimal
Make		char	13		13	0	0
Model		char	40		40	0	0
Type		char	8		8	0	0
Origin		char	6		6	0	0
DriveTrain		char	5		5	0	0
MSRP		double	8	DOLLAR	8	8	0
Invoice		double	8	DOLLAR	8	8	0
EngineSize	Engine Size (L)	double	8		12	0	0
Cylinders		double	8		12	0	0
Horsepower		double	8		12	0	0
MPG_City	MPG (City)	double	8		12	0	0
MPG_Highway	MPG (Highway)	double	8		12	0	0
Weight	Weight (LBS)	double	8		12	0	0
Wheelbase	Wheelbase (IN)	double	8		12	0	0
Length	Length (IN)	double	8		12	0	0

Output 4.2 *PROC PRINT Output: Summary of MSRP and City Miles per Gallon*

Obs	_Column_	_NObs_	_Mean_	_Max_	_Min_	_Std_
1	MSRP	428	32774.85514	192465	10280	19431.716674
2	MPG_City	428	20.060747664	60	10	5.2382176386

Additional Information

- Tables that are saved from a caslib are saved in SASHDAT format by default.
- For documentation about the CASUTIL procedure syntax, see Chapter 3, "CASUTIL Procedure," on page 57.
- For more examples of using the CASUTIL procedure to access and save data, see Chapter 1, "Accessing Data with SAS Cloud Analytic Services," on page 1.

Example 2: Computing Descriptive Statistics with Group-By Variables

Program

The following example shows you how to access your data with SAS Cloud Analytic Services (CAS) and compute all statistics for each variable (treating the entire input table as one group). It also computes all statistics for every unique combination of the formatted values of variables Make and Type. The results are written to an output table.

```
/* options cashost="cloud.example.com" casport=5570; */
cas casauto;                                    /* 1 */

libname mycas cas;                              /* 2 */

proc casutil;                                   /* 3 */
   load data=sashelp.cars replace;
   contents casdata="cars";
quit;

proc mdsummary data=mycas.cars;                 /* 4 */
   var MPG_City;
   groupby / out=mycas.carsmpgcityall;          /* 5 */
   groupby make type / out=mycas.carsmaketype; /* 6 */
run;

proc print data=mycas.carsmpgcityall;           /* 7 */
```

```
   var _Column_ _NObs_ _Mean_ _Std_ _Min_ _Max_;
   title 'Overall City Mileage';
run;

proc print data=mycas.carsmaketype(obs=20);    /* 8 */
   var make Type  _Column_ _NObs_ _Mean_ _Std_ _Min_ _Max_;
   title 'City Mileage by Make and Type';
run;
```

1 The CAS statement creates session Casauto.

2 The LIBNAME statement creates a CAS engine libref.

Note: To run PROC MDSUMMARY and PROC PRINT in CAS, you must specify the CAS engine LIBNAME statement and use the CAS engine libref with both the input and output table names.

3 The CASUTIL procedure loads the data into the default caslib. The LOAD DATA= statement reads the file into memory. The table is now available for analytics. The CONTENTS statement reads the on-disk file, Cars, and displays the table metadata. This enables you to learn if the file has column names in the first row and the data types.

4 The MDSUMMARY procedure produces summary statistics for MPG_CITY.

5 The first GROUPBY statement with the OUT= option creates an in-memory table named Mycas.CarsMPGCityAll that calculates summaries based on all qualifying rows of the input table

6 The second GROUPBY statement with the OUT= option creates an in-memory table named Mycas.CarsMakeType. The in-memory table calculates summaries for rows grouped by Make and Type.

7 The MDSUMMARY procedure does not print output. The PRINT procedure prints table Mycas.MdsumStat.

8 The PRINT procedure prints the first 20 observations in table Mycas.CarsMakeType.

Results

Output 4.3 *PROC CASUTIL Output: Column Information for the Cars Data Set*

Column Information for CARS in Caslib CASUSERHDFS(sasdemo)							
Column	Label	Type	Length	Format Name	Formatted Length	Format Width	Format Decimal
Make		char	13		13	0	0
Model		char	40		40	0	0
Type		char	8		8	0	0
Origin		char	6		6	0	0
DriveTrain		char	5		5	0	0
MSRP		double	8	DOLLAR	8	8	0
Invoice		double	8	DOLLAR	8	8	0
EngineSize	Engine Size (L)	double	8		12	0	0
Cylinders		double	8		12	0	0
Horsepower		double	8		12	0	0
MPG_City	MPG (City)	double	8		12	0	0
MPG_Highway	MPG (Highway)	double	8		12	0	0
Weight	Weight (LBS)	double	8		12	0	0
Wheelbase	Wheelbase (IN)	double	8		12	0	0
Length	Length (IN)	double	8		12	0	0

Output 4.4 *PROC PRINT Output: Overall Summary of City Mileage*

Obs	_Column_	_NObs_	_Mean_	_Std_	_Min_	_Max_
1	MPG_City	428	20.060747664	5.2382176386	10	60

Output 4.5 *PROC PRINT Output: Summary of City Mileage by Make and Type*

Obs	Make	Type	_Column_	_NObs_	_Mean_	_Std_	_Min_	_Max_
1	Audi	Sedan	MPG_City	13	18.615384615	2.3642638579	14	23
2	Audi	Sports	MPG_City	4	19	2.7080128015	15	21
3	BMW	SUV	MPG_City	2	16	0	16	16
4	Chevrolet	SUV	MPG_City	4	15.75	2.3629078131	14	19
5	Chrysler	Sports	MPG_City	1	17	.	17	17
6	GMC	Truck	MPG_City	4	15.75	2.2173557826	13	18
7	Infiniti	Sedan	MPG_City	6	17.833333333	0.7527726527	17	19
8	Isuzu	SUV	MPG_City	2	16	1.4142135624	15	17
9	Mercedes-Benz	Sedan	MPG_City	16	18.0625	2.2351360883	13	22
10	Mercedes-Benz	Sports	MPG_City	5	16.2	3.1144823005	13	21
11	Nissan	Sedan	MPG_City	9	22	3.6742346142	18	28
12	Pontiac	Sports	MPG_City	1	16	.	16	16
13	Saab	Sedan	MPG_City	6	20.666666667	0.5163977795	20	21
14	Saturn	Wagon	MPG_City	1	24	.	24	24
15	Subaru	Sports	MPG_City	2	19	1.4142135624	18	20
16	BMW	Sedan	MPG_City	13	19.230769231	0.8320502943	18	20
17	BMW	Sports	MPG_City	4	18.25	2.6299556397	16	21
18	Buick	SUV	MPG_City	2	17	2.8284271247	15	19
19	Chevrolet	Sedan	MPG_City	15	22.266666667	4.0964560758	14	28
20	Lexus	SUV	MPG_City	3	15.333333333	2.5166114784	13	18

Additional Information

- Caslibs provide a way to access in-memory tables and an associated data source. They also provide a way to apply access controls to data. In this example, the personal caslib Casuser is being used, so no CASLIB statement is needed.
- Tables that are saved from a caslib are saved in SASHDAT format by default.
- For documentation about the CASUTIL procedure syntax, see Chapter 3, “CASUTIL Procedure,” on page 57.
- For more examples of using the CASUTIL procedure to access and save data, see Chapter 1, “Accessing Data with SAS Cloud Analytic Services,” on page 1.

Example 3: Using Multiple Group-By Variables

Program

The following example loads a data set into CAS and computes all statistics for every unique combination of the formatted values of variables Make, Model, Type, and MPG_CITY. The results are written to an output table

```
/* options cashost="cloud.example.com" casport=5570; */
cas casauto;                                /* 1 */

proc casutil;                               /* 2 */
   load data=sashelp.cars replace;
   contents casdata="cars";
quit;

libname mycas cas;                          /* 3 */

proc mdsummary data=mycas.cars ;            /* 4 */
   var MPG_City;
   groupby make model type / out=mycas.carsmiles;
run;

proc print data=mycas.carsmiles(obs=20);  /* 5 */
   var Make Model Type _Column_ _NObs_ _Min_ _Max_;
   title 'City Mileage for Make, Model, and Type';
run;
```

1 The CAS statement creates session Casauto.

2 The CASUTIL procedure loads the data into the default caslib, Casuser. The LOAD DATA= statement reads the file into memory. The table is now available for analytics. The CONTENTS statement reads the on-disk file, Cars, and displays the table metadata. This enables you to learn if the file has column names in the first row and the data types.

3 The LIBNAME statement for the CAS engine creates a CAS libref.

Note: To run PROC MDSUMMARY and PROC PRINT in CAS, you must specify the CAS engine LIBNAME statement and use the CAS engine libref with both the input and output table names.

4 The MDSUMMARY procedure produces summary statistics for MPG_CITY, grouped by Make, Model, and Type. The OUT= option creates an in-memory table named Mycas.CarsMiles. The table includes a row of summary statistics for each unique combination of MPG_City, Make, Model, and Type.

5 The PRINT procedure prints the first 20 observations in table Mycas.CarsMiles.

Results

Output 4.6 *PROC MDSUMMARY Output: City Mileage for Make, Model, and Type*

Obs	Make	Model	Type	_Column_	_NObs_	_Min_	_Max_
1	Audi	A4 3.0 Quattro 4dr manual	Sedan	MPG_City	1	17	17
2	Audi	A41.8T convertible 2dr	Sedan	MPG_City	1	23	23
3	Audi	A6 3.0 Quattro 4dr	Sedan	MPG_City	1	18	18
4	Audi	TT 1.8 convertible 2dr (coupe)	Sports	MPG_City	1	20	20
5	BMW	330xi 4dr	Sedan	MPG_City	1	20	20
6	BMW	525i 4dr	Sedan	MPG_City	1	19	19
7	BMW	Z4 convertible 2.5i 2dr	Sports	MPG_City	1	20	20
8	Buick	Rendezvous CX	SUV	MPG_City	1	19	19
9	Cadillac	SRX V8	SUV	MPG_City	1	16	16
10	Chevrolet	Cavalier 4dr	Sedan	MPG_City	1	26	26
11	Chevrolet	Malibu LT 4dr	Sedan	MPG_City	1	23	23
12	Chevrolet	Silverado SS	Truck	MPG_City	1	13	13
13	Chevrolet	Tahoe LT	SUV	MPG_City	1	14	14
14	Chrysler	Concorde LX 4dr	Sedan	MPG_City	1	21	21
15	Chrysler	Concorde LXi 4dr	Sedan	MPG_City	1	19	19
16	Chrysler	PT Cruiser Limited 4dr	Sedan	MPG_City	1	22	22
17	Chrysler	Sebring Touring 4dr	Sedan	MPG_City	1	21	21
18	Ford	Crown Victoria LX 4dr	Sedan	MPG_City	1	17	17
19	Ford	Escape XLS	SUV	MPG_City	1	18	18
20	GMC	Yukon XL 2500 SLT	SUV	MPG_City	1	13	13

Additional Information

- Caslibs provide a way to access in-memory tables and an associated data source. They also provide a way to apply access controls to data. In this example, the personal caslib Casuser is being used, so no CASLIB statement is needed. The in-memory table Mycas.CarsMiles is temporary, and is dropped when the session is ended. To add tables to a data source permanently, use the SAVE statement in PROC CASUTIL.
- For documentation about the CASUTIL procedure syntax, see Chapter 3, “CASUTIL Procedure,” on page 57.
- For more examples of using the CASUTIL procedure to access and save data, see Chapter 1, “Accessing Data with SAS Cloud Analytic Services,” on page 1.

Example 4: Using Formats with Group-By Variables

Program

The following example defines two value formats, one numeric and the other character, and uploads them to an existing CAS session and applies the formats to two variables.

```
/* options cashost="cloud.example.com" casport=5570; */
cas casauto;                              /* 1 */

libname mycas cas;                        /* 2 */

proc format casfmtlib='fmtlib';           /* 3 */
   value $flvrfmt
      'Chocolate'='Chocolate'
      'Vanilla'='Vanilla'
      'Rum','Spice'='Other Flavor';
   value agefmt (multilabel)
      15 - 29='below 30 years'
      30 - 50='between 30 and 50'
      51 - high='over 50 years';
run;

data mycas.cake;                          /* 4 */
input LastName $ 1-12 Age 13-14 PresentScore 16-17
TasteScore 19-20 Flavor $ 23-32 Layers 34 ;
format age agefmt. flavor $flvrfmt.;
datalines;
Orlando     27 93 80  Vanilla    1
Ramey       32 84 72  Rum        2
Goldston    46 68 75  Vanilla    1
Roe         38 79 73  Vanilla    2
Larsen      23 77 84  Chocolate  .
Davis       51 86 91  Spice      3
Strickland  19 82 79  Chocolate  1
Nguyen      57 77 84  Vanilla    .
Hildenbrand 33 81 83  Chocolate  1
Byron       62 72 87  Vanilla    2
Sanders     26 56 79  Chocolate  1
Jaeger      43 66 74             1
Davis       28 69 75  Chocolate  2
Conrad      69 85 94  Vanilla    1
Walters     55 67 72  Chocolate  2
Rossburger  28 78 81  Spice      2
Matthew     42 81 92  Chocolate  2
Becker      36 62 83  Spice      2
Anderson    27 87 85  Chocolate  1
Merritt     62 73 84  Chocolate  1
;
```

```
proc mdsummary data=mycas.cake;          /* 5 */
   var TasteScore;
   groupby flavor / out=mycas.flav;      /* 6 */
   groupby flavor age / out=mycas.flag; /* 7 */
run;

proc print data=mycas.flav;              /* 8 */
   var flavor _Column_ _NObs_ _Min_ _Max_ _Mean_;
title 'Taste Score for Cake Flavors and Participant''s Age';
title2 'GROUPBY Flavor';
run;

proc print data=mycas.flag;
   var flavor age _Column_ _NObs_ _Min_ _Max_ _Mean_;
title 'Taste Score for Cake Flavors and Participant''s Age';
title2 'GROUPBY Flavor and Age';
run;
```

1 The CAS statement creates session Casauto.

2 The LIBNAME statement for the CAS engine creates a CAS engine libref.

Note: To run PROC MDSUMMARY and PROC PRINT in CAS, you must specify the CAS engine LIBNAME statement and use the CAS engine libref with both the input and output table names.

3 The FORMAT procedure creates the formats Flvrfmt and Agefmt. The CASFMTLIB= option adds the format library to the CAS session. It associates the format library with the CAS tables.

4 The DATA step creates the input data set. This DATA step runs on the SAS Compute Server and not in CAS. However, the DATA step sends the results to CAS in the form of an in-memory CAS table. The CAS engine libref Mycas enables CAS processes to run on the data set.

5 The MDSUMMARY procedure computes summary statistics for cake tasting scores.

6 The first GROUPBY statement with the OUT= option creates an in-memory table named Mycas.Flav that is grouped by Flavor.

7 The second GROUPBY statement with the OUT= option creates an in-memory table named Mycas.Flag that is grouped by Flavor and Age.

8 The PRINT procedure prints the output data sets.

Results

Output 4.7 *PROC PRINT Output: Cake Flavors and Participant's Age Grouped by Flavor*

Obs	Flavor	_Column_	_NObs_	_Min_	_Max_	_Mean_
1		TasteScore	1	74	74	74
2	Chocolate	TasteScore	9	72	92	81.444444444
3	Other Flavor	TasteScore	4	72	91	81.75
4	Vanilla	TasteScore	6	73	94	82.166666667

Output 4.8 *PROC PRINT Output: Cake Flavors and Participant's Age Grouped by Flavor*

Obs	Flavor	Age	_Column_	_NObs_	_Min_	_Max_	_Mean_
1	Chocolate	between 30 and 50	TasteScore	2	83	92	87.5
2		between 30 and 50	TasteScore	1	74	74	74
3	Chocolate	below 30 years	TasteScore	5	75	85	80.4
4	Other Flavor	over 50 years	TasteScore	1	91	91	91
5	Vanilla	between 30 and 50	TasteScore	2	73	75	74
6	Vanilla	over 50 years	TasteScore	3	84	94	88.333333333
7	Chocolate	over 50 years	TasteScore	2	72	84	78
8	Other Flavor	between 30 and 50	TasteScore	2	72	83	77.5
9	Vanilla	below 30 years	TasteScore	1	80	80	80
10	Other Flavor	below 30 years	TasteScore	1	81	81	81

Additional Information

- Caslibs provide a way to access in-memory tables and an associated data source. They also provide a way to apply access controls to data. In this example, the personal caslib Casuser is being used, so no CASLIB statement is needed. The in-memory tables Mycas.Flav and Mycas.Flag are temporary, and are dropped when the session ends. To add tables to a data source permanently, use the SAVE statement in PROC CASUTIL.
- For documentation about the CASUTIL procedure syntax, see Chapter 3, "CASUTIL Procedure," on page 57.
- For more examples of using the CASUTIL procedure to access and save data, see Chapter 1, "Accessing Data with SAS Cloud Analytic Services," on page 1.

Example 5: Graph Summary Statistics Results Obtained from the MDSUMMARY Procedure

Program

The following example loads data into CAS, and creates a plot from the summarized results of the MDSUMMARY procedure.

```
/* options cashost="cloud.example.com" casport=5570; */
cas casauto;                          /* 1 */

libname mycas cas;                    /* 2 */

proc casutil;                         /* 3 */
   load data=sashelp.cars replace;
   contents casdata="cars";
quit;

proc mdsummary data=mycas.cars;       /* 4 */
   var mpg_highway;
   groupby origin type / out=mycas.mpghw_sum;
run;

ods graphics / width=4in;
title "Summarized Highway MPG";
proc sgpanel data=mycas.mpghw_sum;    /* 5 */
   where origin in ("Asia" "USA");
   panelby origin / uniscale=row;
   format _mean_ 2.;
   vbar type / response=_mean_;
   rowaxis label="Summary MPG Values";
run;
title;
ods graphics / reset=all;
```

1 The CAS statement creates session Casauto.

2 The LIBNAME statement for the CAS engine creates a CAS engine libref.

Note: To run PROC MDSUMMARY, PROC SGPANEL, and PROC PRINT in CAS, you must specify the CAS engine LIBNAME statement and use the CAS engine libref with both the input and output table names.

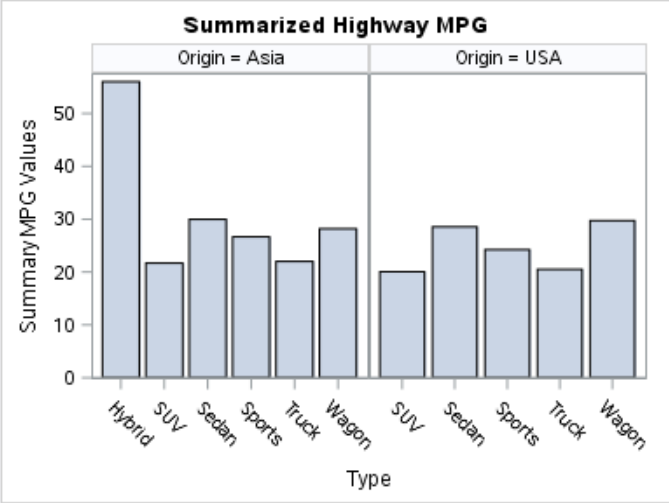
3 The CASUTIL procedure loads the data into the default caslib, Casuser. The LOAD DATA= statement reads the file into memory. The table is now available for analytics. The CONTENTS statement reads the on-disk file, Cars, and displays the table metadata. This enables you to learn if the file has column names in the first row and the data types.

4 The MDSUMMARY procedure produces summary statistics for highway miles-per-gallon. The OUT= option in the GROUPBY statement creates a table in CAS. The table includes a row of summary statistics for each unique combination of origin and type.

5 The SGPANEL procedure plots the summarized results from the MDSUMMARY procedure. The procedure creates a parameterized vertical bar chart that shows the mean statistic for highway miles-per-gallon. The procedure subsets the data, comparing only the cars made in Asia and the U.S.A. The graph is paneled by country of origin.

Results

Output 4.9 *PROC SGPANEL Output*



Additional Information

- For documentation about the CASUTIL procedure syntax, see Chapter 3, “CASUTIL Procedure,” on page 57.
- For more examples of using the CASUTIL procedure to access and save data, see Chapter 1, “Accessing Data with SAS Cloud Analytic Services,” on page 1.
- For information about the SGPANEL procedure, see “SGPANEL Procedure” in *SAS ODS Graphics: Procedures Guide*.

5

Platform Data Sources

Data Redundancy 125

Dictionary 126
Amazon S3 Data Source 126
Azure Data Lake Storage Data Source 135
DNFS Data Source 142
Google Cloud Storage Data Source 144
Path Data Source 146
SAS LASR Analytic Server Data Source 148

Data Redundancy

The following table shows how several factors interact with respect to data redundancy. Data redundancy applies to distributed servers only.

Table 5.1 *Data Redundancy by Data Access Method, Data Source, and File Type*

Data Access Method	Caslib Data Source	Redundancy		File Formats
		SASHDAT files	Other files	
DATA step and `PROC CASUTIL; LOAD DATA=`	The caslib data source isn't a factor for these data access methods.	Not applicable.	Based on COPIES= when the file is loaded.	SAS Data Sets
`PROC CASUTIL; LOAD CASDATA=`	DNFS	COPIES= is not needed,	Based on COPIES= when	SASHDAT and CSV files

Data Access Method	Caslib Data Source	Redundancy		File Formats
		SASHDAT files	Other files	
		surviving nodes reread the file.	the file is loaded.	
	PATH	Based on COPIES= when the file is loaded.	Based on COPIES= when the file is loaded.	SASHDAT, CSV, and PC Files
	Data Connectors	Not applicable.	Based on COPIES= when the file is loaded.	Based on the data connector.

Dictionary

Amazon S3 Data Source

Specifies an Amazon S3 location for loading and saving files that the SAS Cloud Analytic Services controller can access.

Valid in: CASLIB statement, LOAD statement for the CASUTIL procedure, SAVE statement for the CASUTIL procedure

Restriction: You cannot save files larger than 5 TB to the Amazon S3 environment. The largest object currently allowed in the Amazon S3 environment is 5 TB.

See: For system requirements, see "Support for Data Storage in Amazon S3 and MinIO" in *System Requirements for the SAS Viya Platform*.

For the supported file types, see "Path-Based Data Source Types and Options" on page 751.

Syntax

CASLIB *caslib-reference-name*

DATASOURCE=(SRCTYPE="S3" <AWSCONFIGPATH="*string*">
<AWSCONFIGPROFILE="*string*"> <AWSCREDENTIALSPATH="*string*">

<AWSCREDENTIALSPROFILE="*string*"> <AUTHDOMAIN="*authentication-domain*"> <AUTHSCOPE="*scope-URI*"> <ROLENAME="*IAM-role-name*" | ROLEARN="*IAM-role-ARN*'> <ACCESSKEYID="*string*"> <SECRETACCESSKEY="*string*" > <SESSIONTOKEN="*string*"> <KMSCONTEXT="*string*"> <KMSKEYID="*string*"> <SSEHEXKEY="*string*"> <SSETEXTKEY="*string*"> REGION="*region*" BUCKET="*name*" <OBJECTPATH="*path*"> <USESSL=TRUE | FALSE> <USEIAMROLE=TRUE | FALSE> <ACL="*canned-ACL-value*">);

PROC CASUTIL;

LOAD CASDATA="*file-name*" CASOUT="*table-name*"

<DATASOURCEOPTIONS=(ACCESSKEYID="*string*" SECRETACCESSKEY="*string*" <SESSIONTOKEN="*string*"> <ENCRYPTIONDOMAIN="*string*"> <ENCRYPTIONPASSWORD="*string*"> <USESSL=TRUE | FALSE> <USEIAMROLE=TRUE | FALSE>)> <*load-statement-options*>;

PROC CASUTIL;

SAVE CASDATA="*table-name*" CASOUT="*file-name*"

<DATASOURCEOPTIONS=(<ENCRYPTIONDOMAIN="*string*"> <ENCRYPTIONPASSWORD="*string*">)> <*save-statement-options*>;

Options for the S3 Data Source

ACCESSKEYID="*string*"

specifies the Amazon Web Services (AWS) access key ID that you use to access the Amazon S3 environment.

Requirement This option is required for the Amazon S3 data source in the CASLIB statement, unless you specify USEIAMROLE=.

ACL="*canned-ACL-value*"

specifies a canned ACL value that applies to a new object in the specified S3 location.

Here are the valid values:

authenticated-read	log-delivery-write
aws-exec-read	private
bucket-owner-full-control	public-read
bucket-owner-read	public-read-write

For more information about these values, see Amazon S3 Canned ACL documentation.

Default private

Requirement The location that you specify must be within an ACL-enabled S3 bucket.

AUTHDOMAIN="*authentication-domain*"

specifies a pre-defined authentication domain to use when you connect to your data source.

An administrator defines an authentication domain. The authentication domain returns the credentials that are needed to access your data source. For more information, see "Credentials Domain: How to (CLI)" in *SAS Viya Platform: External Credentials*.

AUTHSCOPE="*scope-URI*"
specifies the Microsoft Azure resource and permissions that are required for authorizing access. Enclose the scope URI value in single or double quotation marks.

The authentication scope is used when connecting to Amazon S3 using single sign-on capabilities for the SAS Viya platform. This requires an administrator to configure Microsoft Entra ID as an OpenID Connect provider. For more information, see "Set Up Federated Authentication for Azure and AWS" in *SAS Viya Platform: Authentication*.

Default none

Interaction When you specify AUTHSCOPE=, you must also specify ROLEARN=.

Note Support for this option was added in 2024.12.

Example authscope="storage.azure.com/user_impersonation"

AWSCONFIGPATH="*string*"
specifies the path and file name of an AWS CLI configuration file that contains connection options to access Amazon S3.

Default .aws/config

AWSCONFIGPROFILE="*string*"
specifies a named profile to use from an AWS CLI config files instead of the default profile.

AWSCREDENTIALSPATH="*string*"
specifies the path and file name of an AWS CLI credentials file.

Default .aws/credentials

AWSCREDENTIALSPROFILE="*string*"
specifies a named profile to use from an AWS CLI credentials file instead of the default profile.

BUCKET="*bucket-name*"
specifies the name of the S3 bucket that you are accessing.

Requirement This option is required for the S3 data source in the CASLIB statement.

Tip You can use PROC S3 to manipulate data in an S3 bucket.

ENCRYPTIONDOMAIN="*string*"
specifies the name for a collection of data that is stored with a common encryption password.

ENCRYPTIONPASSWORD="*string*"
specifies a password for encrypting or decrypting stored data.

KMSCONTEXT="*string*"
specifies the AWS Key Management Services encryption context for the cryptographic operation.

KMSKEYID="*string*"
specifies the AWS Key Management Services ID.

OBJECTPATH="*directory-path*"
specifies a subdirectory within an S3 bucket.

REGION="*region-value*"
specifies the region for the Amazon S3 environment that you are accessing. Here are the supported region values.

Note: You can provide either the SAS region value or the AWS region value.

Table 5.2 Predefined Region Values for S3

SAS Region Value	Description	Corresponding Amazon S3 Region
afcapetown	Africa (Cape Town)	af-south-1
aphongkong	Asia Pacific (Hong Kong)	ap-east-1
aphyderabad	Asia Pacific (Hyderabad)	ap-south-2
apindia	Asia Pacific (India)	ap-south-1
apjakarta	Asia Pacific (Jakarta)	ap-southeast-3
apmelbourne	Asia Pacific (Melbourne)	ap-southeast-4
aposaka	Asia Pacific (Osaka)	ap-northeast-3
apseoul	Asia Pacific (Seoul)	ap-northeast-2
apsingapore	Asia Pacific (Singapore)	ap-southeast-1
apsydney	Asia Pacific (Sydney)	ap-southeast-2
aptokyo	Asia Pacific (Tokyo)	ap-northeast-1
cacalgary	Canada (Calgary)	ca-west-1
cacentral	Canada (Central)	ca-central-1
cnbeijing	China (Beijing)	cn-north-1

SAS Region Value	Description	Corresponding Amazon S3 Region
cnningxia	China (Ningxia)	cn-northwest-1
eufrankfurt	EU (Frankfurt)	eu-central-1
euireland	EU (Ireland)	eu-west-1
eulondon	EU (London)	eu-west-2
eumilan	EU (Milan)	eu-south-1
euparis	EU (Paris)	eu-west-3
euspain	EU (Spain)	eu-south-2
eustockholm	EU (Stockholm)	eu-north-1
euzurich	EU (Zurich)	eu-central-2
fips	US Government Cloud (FIPS)	fips-us-gov-west-1
fipseast	US Government Cloud (FIPS East)	fips-us-gov-east-1
iltelaviv	Israel (Tel Aviv)	il-central-1
mebahrain	Middle East (Bahrain)	me-south-1
meuae	Middle East (UAE)	me-central-1
sa	South America (Sao Paulo)	sa-east-1
useast	US East (Virginia)	us-east-1
useastoh	US East (Ohio)	us-east-2
usgov	US Government Cloud	us-gov-west-1
usgoveast	US Government Cloud (East)	us-gov-east-1
usstd	US Standard	us-east-1
uswest	US West (Oregon)	us-west-2
uswestca	US West (California)	us-west-1

Requirement This option is required for the S3 data source in the CASLIB statement.

ROLEARN="*IAM-role-ARN*"
specifies an Amazon Resource Name (ARN) that identifies an IAM role. The associated IAM role enables you to get access credentials to S3 via the AssumeRole IAM operation.

The CASLIB connection to S3 uses an EC2 instance and an associated IAM role when you set USEIAMROLE=TRUE. If the role that you want to use is already associated with the profile for the EC2 instance where SAS processes are executing, then you do not need to specify ROLEARN=. The role ARN that you specify must already be defined on AWS.

Interactions To use the ROLEARN= option, you must also set USEIAMROLE=TRUE.

Specify either the ROLEARN= option or the ROLENAME= option. Do not specify both.

Note Specify this option only to use a role ARN that is not associated with the EC2 instance that you are using.

Example `rolearn="arn:aws:iam::123456789012:role/s3AccessRole"`

ROLENAME="*IAM-role-name*"
specifies an IAM role name that enables you to get access credentials to S3 via the AssumeRole IAM operation.

The CASLIB connection to S3 uses an EC2 instance and an associated IAM role when you set USEIAMROLE=TRUE. If the role that you want to use is already associated with the profile for the EC2 instance where SAS processes are executing, then you do not need to specify ROLENAME=. The role name that you specify must already be defined on AWS.

Interactions To use the ROLEARN= option, you must also set USEIAMROLE=TRUE.

Specify either the ROLENAME= option or the ROLEARN= option. Do not specify both.

Note Specify this option only to use a role name that is not associated with the EC2 instance that you are using.

Example `rolename="s3AccessRole"`

SECRETACCESSKEY="*string*"
specifies the AWS secret access key that you use to access the Amazon S3 environment.

Requirement This option is required for the Amazon S3 data source in the CASLIB statement, unless you specify USEIAMROLE=.

SESSIONTOKEN="*string*"
specifies the AWS session token that you use with ACCESSKEYID= and SECRETACCESSKEY= to access the Amazon S3 environment.

A session token is used typically with federated accounts. Do not specify this option if your credentials do not include a session token.

SRCTYPE="S3"
specifies that the data source is an Amazon S3 environment.

Requirement This option is required for the S3 data source in the CASLIB statement.

SSEHEXKEY="*string*"
specifies an encryption key in the form of a 64-digit hexadecimal value.

SSETEXTKEY="*string*"
specifies a plain-text version of the key to use with server-side encryption.

USEIAMROLE=TRUE | FALSE
specifies whether to use credentials that are associated with an IAM role. The IAM role is associated with an EC2 instance on an S3 platform.

Default FALSE

Requirement When USEIAMROLE=TRUE, the SUPERUSER or DATA role must be assigned to the adminRole for the assumeRole action.

Interaction If you specify USEIAMROLE=, then do not specify ACCESSKEYID=, SECRETACCESSKEY=, or SESSIONTOKEN=.

USESSL=TRUE | FALSE
specifies whether to use an SSL connection to an S3 bucket.

Default TRUE

Details

Support for Amazon S3 in the SAS Viya Platform

For system requirements, see "Support for Data Storage in Amazon S3 and MinIO" in *System Requirements for the SAS Viya Platform*.

The SAS Viya platform can access data that is stored in Amazon S3 by providing two main components:

- The "Amazon S3 Data Source" enables you to use the CASLIB statement to connect SAS Cloud Analytic Services (CAS) with your S3 instance. For the supported file types, see "Path-Based Data Source Types and Options" on page 751.
- The "FILENAME Statement: S3 Access Method" in *SAS Global Statements: Reference* enables you to connect SAS Compute Server with your data source.

The following resources might also be helpful:

- The "S3 Action Set" in *SAS Viya Platform: System Programming Guide* enables you to manage custom regions from CAS.
- The S3 procedure enables you to manage S3 objects and custom regions from SAS Compute Server.
- The Parquet LIBNAME statement enables you to connect SAS Compute Server with Parquet tables in Amazon S3 or in Google Cloud Storage.

Support for BIGINT Data

Starting in 2026.02, support was added for reading BIGINT (INT64) data into CAS. Computations that run in CAS maintain the original BIGINT precision. For more information, see "Numeric Data" on page 613.

Support for IMDS and IAM Roles

Support has been added for EC2 Instance Metadata Service version 2 (IMDSv2). IMDSv1 continues to be supported. When deployed to Amazon EKS, the SAS Viya platform uses IMDS to obtain an IAM role that enables you to access S3 resources.

Your SAS Viya offering first attempts to use IMDSv2 to obtain a session token. If using IMDSv2 fails, then IMDSv1 is used.

Specify the following options in your Amazon environment:

- When you set http-tokens to required, you must use IMDSv2. If http-tokens is set to optional, then either IMDSv1 or IMDSv2 can be used.
- Set the http_put_response_hop_limit to 2 when you use IMDSv2.

Loading Compressed CSV Files from S3

You can load compressed CSV files from the S3 data source. If your CSV file is in a GZIP file, it must be the only CSV file in that GZIP file. The GZIP file can contain other file types. That is, when archiveType='gz', multiFile= must be set to False, which is the default.

You can load multiple CSV files from a ZIP file. Therefore, if archiveType='zip', then multiFile= can be set to True or False.

For more information about importOptions=, see "Options for the CSV File Type" on page 154.

Support for XLS and XLSX Files from S3

The SAS Viya platform supports only loading (reading) XLS and XLSX files from an S3 environment. These files cannot currently be written back to S3.

Examples

Example 1: Add the S3 Data Source to the Mys3 Caslib

Define a caslib that connects to the S3 data source. Define a connection between CAS and the S3 data source by specifying the S3 source type (SRCTYPE=), S3 specific authentication values, the S3 bucket and location. The USESSL= option specifies to use an SSL connection to S3.

```
caslib mys3 datasource=(srctype="s3"
               accesskeyid="LASKDJFLKJE9EXAMPLE"
               secretaccesskey="kJflkusXRrtUMi/M8zbWofa/
bPxrGoaKDEXAMPLE"
               region="EU_Ireland"
               bucket="mybucket"
               objectpath="/asr/projB"
               usessl=true);
```

Example 2: Load a CSV File from a GZIP File in S3

You can load a CSV file that is stored in a GZIP file (*.gz). This example shows how to load a compressed CSV file.

Define a caslib that connects to the S3 data source. Define a connection between CAS and the S3 data source by specifying various security values and S3 values, such as region and bucket name. The SUBDIRS option specifies that subdirectories can be accessed by the casref `mys3`. For more information, see "CASLIB Statement" on page 745.

```
caslib mys3 dataSource=(srcType="s3"
               accesKeyId="JLEM4SA8CIJ2CG7KABOD"
               secretAccessKey="TeIOq748lRg6nSLKjsrk7lskdMiLwK/
gIOkDwbMd4C"
               bucket="mybucket"
               objectPath="/projdata_csv/gz/single"
               region="US_West_Oregon")
               subdirs;
```

Load data in a compressed CSV file from S3 into CAS. First, list the files in the S3 data source. Load the data from the `data_utf8.csv.gz` file, a GZIP compressed CSV file. The file resides in the `mys3` data source location. The CASOUT= option designates the name of the in-memory table, `load-utf8_csv_gz_out`. Specify the import options (IMPORTOPTIONS=) that describe the file to load—a GZIP compressed CSV file with column names in the first observation. Then, list the tables that have been loaded into memory in the `mys3` caslib. For more information, see Chapter 3, "CASUTIL Procedure," on page 57 and "Options for the CSV File Type" on page 154.

```
proc casutil;
   list files;
   load casdata="data_utf8.csv.gz" incaslib="mys3"
casout="load_utf8_csv_gz_out"
           replace
           importOptions=(fileType="csv",archiveType='gz',
                          getnames="true");
```

```
   list tables;
quit;
```

Example 3: Use an IAM Role with Your CASLIB Connection to S3

The following CASLIB statements connect to an EC2 instance on S3. The first example statement assumes that an IAM role, `myrole1`, is associated with the EC2 instance on S3. The second CASLIB statement shows how to specify an IAM role that is not already associated with the EC2 instance that you are using. The ROLENAME= and ROLEARN= options behave similarly. Use them only when you are using an IAM role that is not associated with the EC2 instance that you are accessing.

Define a caslib that connects to an EC2 instance on an S3 data source. The EC2 instance in this example is associated with an IAM role named `myrole1`. Because the IAM role is already associated with the EC2 instance that you are using, do not include the ROLENAME= option. Specify the USEIAMROLE= option to indicate that your SAS Viya offering should use the IAM role that is associated with your EC2 instance.

```
caslib mys3 datasource=(srctype="s3"
              region="EU_Ireland"
              bucket="mybucket"
              objectpath="/asr/projB"
              USEIAMROLE=true);
```

The second example statement shows how to specify an IAM role that is not already associated with an EC2 instance on S3.

Define a caslib that connects to an EC2 instance on an S3 data source. The EC2 instance for the statement below is not already associated with the `myrole2` IAM role, so you need to include the ROLENAME= option. The `myrole2` IAM role must already be defined in the S3 environment.

```
caslib mys3 datasource=(srctype="s3"
              region="EU_Ireland"
              bucket="mybucket"
              objectpath="/asr/projB"
              ROLENAME="myrole2");
```

Azure Data Lake Storage Data Source

Specifies an Azure Data Lake Storage Gen2 directory for loading and saving files with SAS Cloud Analytic Services.

Valid in: CASLIB statement, PROC CASUTIL: DELETESOURCE, LOAD, and SAVE statements

Requirements: The user ID that is used to authenticate must be assigned Contributor and Storage Blob Data Contributor roles in the ADLS account configuration.

Authentication is required. For details, see “Authentication” on page 138.

The ADLS path specifies the folder and subfolder. The path must begin with only a forward slash (/). It cannot begin with a period (.) or with only the folder name. This example uses the appropriate convention:

```
/folder/subfolder
```

See: For the supported file types, see "Path-Based Data Source Types and Options" on page 751.

Syntax

CASLIB *caslib-reference-name*

DATASOURCE=(SRCTYPE="ADLS"
ACCOUNTNAME="*string*"<APPLICATIONID="*string*">
<CLIENTSECRET="*string*"> FILESYSTEM="*file-system-name*"
<DNSSUFFIX="*network-storage-host-name*" > <TIMEOUT=*seconds*>)
<*caslib-statement-options*> <SUBDIRS>;

PROC CASUTIL;

DELETESOURCE CASDATA="*file-name*" <INCASLIB="*caslib*"> <*deletesource-statement-options*>;

PROC CASUTIL;

LOAD CASDATA="*file-name*" CASOUT="*table-name*"

<DATASOURCEOPTIONS=(*...options...*)> <*load-statement-options*>;

PROC CASUTIL;

SAVE CASDATA="*table-name*" CASOUT="*file-name*"

<DATASOURCEOPTIONS=(*...options...*)> <*save-statement-options*>;

Options for Azure Data Lake Storage Data Source

ACCOUNTNAME="*account-name*"
specifies the name of the account to use on the external storage system.

Valid in: CASLIB statement

PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Requirement: This option is required for the CASLIB statement.

APPLICATIONID="*string*"
specifies the application ID (a GUID string) for accessing the external storage system.

Valid in: CASLIB statement

PROC CASUTIL: DELETESOURCE, LOAD, and SAVE statements

Notes: The APPLICATIONID option is mandatory for the device code and service-principle-based authentication.

The APPLICATIONID option is not supported in single sign-on authentication.

A CASLIB statement that uses the APPLICATIONID option and does not use CLIENTSECRET option must use the device -code authentication.

A CASLIB statement that uses the APPLICATIONID and CLIENTSECRET options must use the service principal-based authentication.

Example `applicationId="b1fc955d5c-e0e2-45b3-a3cc-a1cf54120f"`

CLIENTSECRET="*client secret parameter*"
specifies a client's secret authentication parameter to access ADLS.

Valid in CASLIB statement

Alias secret

Notes A CASLIB statement that uses the APPLICATIONID option and does not use CLIENTSECRET option must use the device -code authentication.

A CASLIB statement that uses the APPLICATIONID and CLIENTSECRET options must use the service principal-based authentication.

DNSSUFFIX="*network-storage-host-name*"
specifies the network host name where the storage system resides. It is referred to as a suffix because the value is typically appended to the end of the storage account name.

Valid in CASLIB statement

PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Default dfs.core.windows.net

Example `dnsSuffix="dfs.core.windows.net"`

FILESYSTEM="*file-system-name*"
specifies the name of the file system within the external storage system that files are to be read from and written to.

Valid in CASLIB statement

PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Requirement The FILESYSTEM= option is required for the CASLIB statement.

SRCTYPE="ADLS"
specifies that the data source is Azure Data Lake Storage Gen2.

Requirement The SRCTYPE= options is required for the CASLIB statement.

TIMEOUT=*seconds*
specifies the maximum completion time for a storage task, expressed as a number of seconds. If the task does not complete within the specified time, the storage system might return an error. Use this option to establish a connection to the external storage system.

Valid in CASLIB statement

PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Default 60

Example `timeout=600`

Details

Support for ADLS in the SAS Viya Platform

For system requirements, see *System Requirements for the SAS Viya Platform*.

The SAS Viya platform can access data that is stored in ADLS by providing these main components:

- The "Azure Data Lake Storage Data Source" data connector enables you to use the CASLIB statement to connect SAS Cloud Analytic Services (CAS) with your data source. For the supported file types, see "Path-Based Data Source Types and Options" on page 751.
- The "FILENAME Statement: Azure Data Lake Storage Access Method" in *SAS Global Statements: Reference* and the "FILENAME Statement: Azure Blob Storage (ABS) Access Method" in *SAS Global Statements: Reference* enable you to connect SAS Compute Server with your data source.

The following resource might also be helpful: The ORC LIBNAME statement enables you to connect SAS Compute Server with ORC tables in ADLS.

Authentication

Overview of Authentication for ADLS

To access ADLS caslibs, you can use one of the following methods for authentication. Select each link for more information about the requirements.

Table 5.3 *Authentication Methods for ADLS*

Authentication Method	Example Code
Single sign-on	`caslib name datasource=(srctype="adls"`

Authentication Method	Example Code
	accountname="*my-account*" filesystem="*file-system-name*") subdirs ;
Service principal and client secret	cas casauto sessopts=(azuretenantid="*tenant-id*"); caslib *name* datasource=(srctype="adls" accountname="*my-account*" filesystem="*file-system-name*" applicationId="*string*" clientSecret="*my-secret*") subdirs ;
Device code flow	cas casauto sessopts=(azuretenantid="*tenant-id*"); caslib *name* datasource=(srctype="adls" accountname="*my-account*" filesystem="*file-system-name*" applicationId="*string*") subdirs ;

Requirements for Single Sign-On Authentication

The preferred method is to configure the SAS Viya platform for single sign-on with Microsoft Entra ID (formerly Azure Active Directory), using the OIDC authentication protocol. For details, see "Scenario: OIDC with Microsoft Entra ID" in *SAS Viya Platform: Authentication*.

Requirements for Service Principal and Client Secret Authentication

Starting with 2026.05, you can authenticate using a service principal with a client secret by specifying the client ID (APPLICATIONID=) and the client token (CLIENTSECRET=) in addition to other required options. This method of authentication is not supported for ORC files. Note that it might be necessary to delete the user's device code file, if they have previously used device-code authentication.

This information is not needed for single sign-on.

Table 5.4 *Requirements for Client Secret Authentication in Azure Data Lake Storage Gen2*

Requirement	How to Meet the Requirement
Tenant ID	If all users can use the same tenant ID, then an administrator can configure a server option to set the tenant ID. For more information, see the cas.AZURETENANTID= server option.

Requirement	How to Meet the Requirement
	Users can set a tenant ID individually or override the server setting by specifying the "AZURETENANTID= Session Option".
Client secret	Specify the CLIENTSECRET= option in the caslib.
Account, Application ID, and File system	Specify the ACCOUNTNAME=, APPLICATIONID=, and FILESYSTEM= options in the caslib. The LOAD CASDATA=, SAVE, and other statements for the CASUTIL procedure can override these values in the DATASOURCEOPTIONS= option.

Requirements for Device Code Flow Authentication

This information is not needed for single sign-on.

Table 5.5 *Requirements for Device Code Flow to Access Data in Azure Data Lake Storage Gen2*

Requirement	How to Meet the Requirement
Tenant ID	If all users can use the same tenant ID, then an administrator can configure a server option to set the tenant ID. For more information, see the cas.AZURETENANTID= server option. Users can set a tenant ID individually or override the server setting by specifying the "AZURETENANTID= Session Option".
Authentication	An ADLS caslib uses device code authentication with Azure. After users authenticate, the credentials are cached in a path that is accessible to the CAS controller. To cache the credential files for all users in a single directory, an administrator can configure the cas.AZUREAUTHCACHELOC= server option. If the server option is not configured, then the credential file is created in the home directory for the host account that runs the CAS session.
Account, Application ID, and File system	Specify the ACCOUNTNAME=, APPLICATIONID=, and FILESYSTEM= options in the caslib. The LOAD CASDATA=, SAVE, and other statements for the CASUTIL procedure can override these values in the DATASOURCEOPTIONS= option.

The first time that you access an ADLS caslib, this error message is displayed in the client log to authorize access.

```
ERROR: Authorization Pending user-interaction
ERROR: To sign in, use a web browser to open the page
       https://microsoft.com/devicelogin and
       enter the code ###### to authenticate.
```

After you log on to the Microsoft web page, a series of windows provide further instruction and result in the required authorization. Subsequent attempts to access data succeed. The authorization steps create a file that is accessible to the CAS controller for the host account that is used to run the CAS session. The file is named .sasadls_**userid**.json and is used to cache authentication information. It is created in the directory that is specified in the cas.AZUREAUTHCACHELOC= server option. If cas.AZUREAUTHCACHELOC= was not set when the server was started, the JSON file is created in the home directory for the host account that is used to run the CAS session.

This authentication and authorization step is not needed as long as the Azure authentication JSON file is not deleted and the user credentials do not change.

Examples

Example 1: Add an ADLS Caslib Using Single Sign-on and List the Available Files

For more information about authentication methods, see "Authentication" on page 138.

```
cas casauto;

caslib azds datasource=(srctype="adls"
                        accountname="myaccount"
                        filesystem="dir")
   subdirs;

proc casutil;
   list files;
quit;
```

Example 2: Load an ORC File from an ADLS Caslib

For more information about authentication methods, see "Authentication" on page 138.

```
/* After adding an ADLS caslib. */

proc casutil;
   load casdata="somefile.orc" casout="somefile";                /* 1 */
   load casdata="subdir/otherfile.orc" casout="otherfile"; /* 2 */
quit;
```

1 By default, files are read from the root directory of the caslib. This is equivalent to the filesystem option that is specified when the caslib is added.

2 If the caslib is added with the SUBDIRS option, then you can specify the subdirectory to access in the CASDATA= option.

Example 3: Save a Table as an ORC File in an ADLS Caslib

For more information about authentication methods, see "Authentication" on page 138.

```
/* After adding an ADLS caslib. */

proc casutil;
   load data=sashelp.class casout="class" outcaslib="azds";    /* 1 */
   save casdata="class" casout="class.orc";                    /* 2 */
quit;
```

1 You can load data into CAS in any way that you prefer. This short example shows the LOAD DATA= syntax that reads data from a SAS libref and loads it into the server.

2 To save data to Azure Data Lake Storage Get 2 as an ORC file, specify the .orc file name suffix or specify the EXPORTOPTIONS=(FILETYPE="ORC") options with SAVE statement.

DNFS Data Source

Specifies a server-side directory for loading and saving files that the SAS Cloud Analytic Services controller can access. The directory must be mounted by every machine that is used by the server.

Valid in: CASLIB statement, LOAD CASDATA= statement for the CASUTIL procedure, SAVE statement for the CASUTIL procedure

Note: Treat SASHDAT files in DNFS as read-only or force the server to copy the blocks to CAS_DISK_CACHE. If a SASHDAT file is replaced while other sessions have the table in memory, those sessions can experience errors and become unusable. For more information, see "Controlling Use of CAS_DISK_CACHE" in *SAS Viya Platform: System Programming Guide*.

See: For the supported file types, see "Path-Based Data Source Types and Options" on page 751.

Syntax

CASLIB *caslib-reference-name*

DATASOURCE=(SRCTYPE="DNFS" <ENCRYPTIONDOMAIN="*string*" > <ENCRYPTIONPASSWORD="*string*">) PATH="*/directory-path*" <*caslib-statement-options*>;

PROC CASUTIL;

LOAD CASDATA="*file-name*" CASOUT="*table-name*"

```
    DATASOURCEOPTIONS=(<ENCRYPTIONDOMAIN="string" >
      <ENCRYPTIONPASSWORD="string">) <load-statement-options>;

PROC CASUTIL;
    SAVE CASDATA="table-name" CASOUT="file-name"
    DATASOURCEOPTIONS=(<ENCRYPTIONDOMAIN="string" >
      <ENCRYPTIONPASSWORD="string">) <save-statement-options>;
```

Options for the DNFS Data Source

ENCRYPTIONDOMAIN="*string*"
specifies the name for a collection of data that is stored with a common encryption password.

ENCRYPTIONPASSWORD="*string*"
specifies a password for encrypting or decrypting stored data.

PATH="*/directory-path*"
specifies the fully qualified path to the directory to use as a data source. Notice that the PATH= argument is specified outside of the parenthesis for the DATASOURCE= argument.

SRCTYPE="DNFS"
specifies that the data source is a directory that is mounted by every machine that is used for SAS Cloud Analytic Services.

Requirement The SRCTYPE= option is required.

Details

Overview of the DNFS Data Source

DNFS is an abbreviation for distributed network file system. This data source type provides support for distributed data access to NFS directories. Several systems such as MapR-FS, EMC Isilon, and others provide high-availability, replicated, high-performance, stand-alone storage clusters with an NFS interface. These systems offer popular alternatives to Hadoop.

DNFS can also be used to access NFS-mounted directories from standard Linux file systems.

The design principle is that NFS-mounted directories are accessed concurrently by each controller node and worker node in a distributed server. This is why the directory path for a DNFS caslib must be mounted on every machine. DNFS performs parallel read and write for SASHDAT and CSV files that are stored in the directory path specified for the caslib.

Support for DNFS and Related Data Sources in the SAS Viya Platform

For system requirements, see *System Requirements for the SAS Viya Platform*.

The SAS Viya platform can access DNFS-based and path-based data sources by providing these main components:

- The CASLIB statement enables you to access your files from SAS Cloud Analytic Services (CAS). Here is the syntax for two useful CASLIB statements:
  - "DNFS Data Source" on page 142
  - "Path Data Source" on page 146
- LIBNAME statements and FILENAME statements enable SAS Compute Server to access many file types. Here are some commonly used path-based LIBNAME statements:
  - "LIBNAME Statement: ORC and Parquet Engines" in *SAS Viya LIBNAME Engines for ORC and Parquet: Reference*
  - "LIBNAME Statement: SPD Engine" in *SAS Scalable Performance Data Engine: Reference*
  - "LIBNAME Statement: V9 Engine" in *SAS V9 LIBNAME Engine: Reference*

Example: Add the /net/fileserver Directory as a DNFS Data Source for SAS Cloud Analytic Services

```
caslib dnfsds datasource=(srctype="dnfs") path="/net/fileserver/";
```

See Also

The SAVE example for the Path data source on page 147 can be performed with a DNFS data source.

Google Cloud Storage Data Source

Specifies a Google Cloud Storage directory for loading and saving files with SAS Cloud Analytic Services.

Valid in: CASLIB Statement, PROC CASUTIL: DELETESOURCE, LOAD, and SAVE Statements

See: For the supported file types, see "Path-Based Data Source Types and Options" on page 751.

Syntax

CASLIB *caslib-reference-name*

DATASOURCE=(SRCTYPE="GCS" AUTHDOMAIN="*authentication-domain*" BUCKET="*string*" GCSAUTHFILE="*string*");

PROC CASUTIL;

DELETESOURCE CASDATA="*file-name*" <INCASLIB="*caslib*"> <*deletesource-statement-options*>;

PROC CASUTIL;

LOAD CASDATA="*file-name*" CASOUT="*table-name*"

<DATASOURCEOPTIONS=(*...options...*)> <*load-statement-options*>;

PROC CASUTIL;

SAVE CASDATA="*table-name*" CASOUT="*file-name*"

<DATASOURCEOPTIONS=(*...options...*)> <*save-statement-options*>;

Options for Google Cloud Storage Data Source

AUTHDOMAIN="*authentication-domain*"
specifies a pre-defined authentication domain to use when you connect to your data source.

An administrator defines an authentication domain. The authentication domain returns the credentials that are needed to access your data source. For more information, see "Credentials Domain: How to (CLI)" in *SAS Viya Platform: External Credentials*.

Valid in CASLIB statement

BUCKET="*string*"
specifies the URI for your Google Cloud Storage bucket.

Valid in CASLIB Statement

PROC CASUTIL: DELETESOURCE, LOAD, and SAVE Statements

GCSAUTHFILE="*string*"
specifies the path for your Google Cloud Storage authentication file for your Google service account credential file.

Valid in CASLIB Statement

PROC CASUTIL: DELETESOURCE, LOAD, and SAVE Statements

Details

Support for Google Cloud Storage in the SAS Viya Platform

For system requirements, see *System Requirements for the SAS Viya Platform*.

The SAS Viya platform can access data that is stored in Google Cloud Storage by providing two main components:

- The "Google Cloud Storage Data Source" data connector enables you to use the CASLIB statement to connect SAS Cloud Analytic Services (CAS) with your data source. For the supported file types, see "Path-Based Data Source Types and Options" on page 751.
- The "FILENAME Statement: Google Cloud Storage (GCS) Access Method" in *SAS Global Statements: Reference* enables you to connect SAS Compute Server with your data source.

The following resource might also be helpful: The Parquet LIBNAME statement enables you to connect SAS Compute Server with Parquet tables in Google Cloud Storage or in Amazon S3.

Example: Add a Google Cloud Storage Caslib

```
cas auto;

caslib gcs
   datasource=(
     srctype="gcs"
     bucket="mybucket"
     gcsauthfile="myGCSauthfile"
     );
```

Path Data Source

Specifies a server-side directory for loading and saving files that the SAS Cloud Analytic Services controller can access.

Valid in: CASLIB statement, LOAD CASDATA= statement for the CASUTIL procedure, SAVE statement for the CASUTIL procedure

See: For the supported file types, see "Path-Based Data Source Types and Options" on page 751.

Syntax

CASLIB *caslib-reference-name*

DATASOURCE=(SRCTYPE="PATH" <ENCRYPTIONDOMAIN="*string*" > <ENCRYPTIONPASSWORD="*string*">) PATH="*/directory-path*" <*caslib-statement-options*>;

PROC CASUTIL;

LOAD CASDATA="*file-name*" CASOUT="*table-name*"

DATASOURCEOPTIONS=(<ENCRYPTIONDOMAIN="*string*" > <ENCRYPTIONPASSWORD="*string*">) <*options*> <*load-statement-options*>;

PROC CASUTIL;

SAVE CASDATA="*table-name*"

DATASOURCEOPTIONS=(<ENCRYPTIONDOMAIN="*string*" >
<ENCRYPTIONPASSWORD="*string*">) <*options*> <*save-statement-options*>;

Data Source Options

ENCRYPTIONDOMAIN="*string*"
specifies the name for a collection of data that is stored with a common encryption password.

ENCRYPTIONPASSWORD="*string*"
specifies a password for encrypting or decrypting stored data.

PATH="*/directory-path*"
specifies the fully qualified path to the directory to use as a data source. Notice that the PATH= argument is specified outside of the parenthesis for the DATASOURCE= argument.

SRCTYPE="PATH"
specifies that the data source is a directory that is accessible to the SAS Cloud Analytic Services controller.

Requirement The SRCTYPE= option is required.

Examples

Example 1: Add the data01 Directory as a Path Data Source for SAS Cloud Analytic Services

```
caslib pathds datasource=(srctype="path") path="/net/fileserver/
data01";
```

Example 2: Save a SASHDAT File into a Path Data Source Using PROC CASUTIL

```
proc casutil;
   load data=sashelp.class casout="class";

   save casdata="class" casout="male.sashdat" where="sex eq
'M'"        /* 1 */
      datasourceoptions=(encryptionPassword="secretm");

   save casdata="class" casout="female.sashdat" where="sex eq 'F'"
      datasourceoptions=(enryptionPassword="secretf");

   list files;

   load casdata="female.sashdat"
casout="female"                        /* 2 */
```

```
      datasourceoptions=(encryptionPassword="secretf");
quit;

libname mycas
cas;                                                    /* 3 */

proc print data=mycas.female;
run;
```

1 The SAVE statement is used to write the male.sashdat file to directory associated with the active caslib. The ENCRYPTIONPASSWORD= option is used to set the password SecretM.

2 The LOAD CASDATA= statement reads the female.sashdat file from the directory associated with the active caslib and loads it into memory. The in-memory table is named Female. The ENCRYPTIONPASSWORD= option is used to decrypt the data.

3 The LIBNAME statement assigns a CAS LIBNAME engine libref, Mycas. Because the CASLIB= option is not specified, the active caslib, PathDs, is used. The PRINT procedure can access the in-memory table that is named Female.

SAS LASR Analytic Server Data Source

Specifies the connection options for loading data from the SAS LASR Analytic Server into SAS Cloud Analytic Services.

Valid in: CASLIB statement, LOAD CASDATA= statement of the CASUTIL procedure

Requirement: A network connection must exist between the LASR server and Viya.

See: For more information about the signer service, see “SAS LASR Authorization Service” in the “Security” topic of *SAS Visual Analytics: Administration Guide*.

Syntax

CASLIB *caslib-reference-name*

DATASOURCE=(SRCTYPE="LASR" SERVER="*host-name*" PORT=*integer* <*data-source-options*>) <*caslib-statement-options*>;

PROC CASUTIL;

LOAD CASDATA="*file-name*" CASOUT="*table-name*"

<DATASOURCEOPTIONS=(*data-source-options*)>

IMPORTOPTIONS=(FILETYPE="LASR" *file-type-options*) <*load-statement-options*>;

Options for the LASR Analytic Server Data Source

METALIB="*metadata-libref*"
specifies the libref name for the SAS LASR Analytic Server engine library.

PASSWORD="*string*"
specifies the password for the identity in the USERNAME= option.

Requirement This option is required when using the signer service.

PORT=*integer*
specifies the network port that the SAS LASR Analytic Server listens on.

Default none

Requirement This option is required.

SERVER="*host-name*"
specifies the host name or IP address of the SAS LASR Analytic Server.

SIGNER="*authorization-web-service-uri*"
specifies the URI for the SAS LASR Authorization web service, which is specified as SIGNER="https://server.example.com/SASLASRAuthorization".

Requirement This option is required when using the signer service.

SRCTYPE="LASR"
specifies that the data source is a SAS LASR Analytic Server.

Requirement This option is required.

TAG=*server-tag*
specifies a tag that is used to qualify the names of SAS LASR Analytic Server tables that are accessed using this caslib.

Requirement This option is required when using the signer service.

USERNAME="*user-ID*"
specifies an identity that is authorized to access data in the SAS LASR Analytic Server.

Requirement This option is required when using the signer service.

Options for the LASR Analytic Server File Type

COMPPGM="*string*"
specifies an expression for each variable that you included in the COMPVARS option. End the expression for each variable with a semicolon.

COMPVARS=("*computed-variable-1*" <, "*computed-variable-2*", ...>)
specifies the names of the computed variables to create. Specify an expression for each option in the COMPPGM option.

FILETYPE="LASR"
specifies the file type.

Requirement This option is required.

PARALLELMODE="FALLBACK" | "FORCE" | "NONE"
specifies how the table is transferred from SAS LASR Analytic Server to SAS Cloud Analytic Services when both servers are distributed servers. If either server is running as a single-machine server, parallel data transfer is not possible, which is equivalent to NONE.

Default FALLBACK

FALLBACK
specifies that the worker nodes try to establish communication with each other. If the worker nodes cannot connect, the operation falls back to a serial data transfer between the SAS LASR Analytic Server root node and the SAS Cloud Analytic Services controller node. Serial data transfer is slower than parallel data transfer.

FORCE
specifies that the worker nodes try to establish communication with each other. If the worker nodes cannot connect to perform a parallel data transfer, the load request fails.

NONE
specifies to perform a serial data transfer between the SAS LASR Analytic Server root node and the SAS Cloud Analytic Services controller node.

PRESERVEORDER=TRUE | FALSE
when set to TRUE, rows are inserted into the new table in the same order as they are received from the SAS LASR Analytic Server. Creating the table is less efficient when you use this option.

Default FALSE

VARCHARS=TRUE | FALSE
when set to TRUE, variable-length strings are used for character variables.

Default FALSE

VARS=("*string-1*" <, "*string-2*", ...>)
specifies the variables to use in the action.

WHERE="*where-expression*"
specifies an expression to subset the input data.

Details

Options Summary

Table 5.6 *Summary of Options for SAS LASR Analytic Server*

Options	Valid Data Source Options in the CASLIB Statement	Valid Import Options in the CASUTIL Procedure
COMPPGM=		•
COMPVARS=		•
FILETYPE=		required
METALIB=	•	
PARALLELMODE=		•
PASSWORD=	required	
PORT=	required	
PRESERVEORDER=		•
SRCTYPE=	required	
SERVER=	required	
SIGNER=	required	
TAG=	required	
USERNAME=	required	
VARCHARS=		•
VARS=		•
WHERE=		•

Examples

Example 1: Add a SAS LASR Analytic Server as a Data Source for SAS Cloud Analytic Services

This example shows how to load a table from a SAS LASR Analytic Server and use the VARS= options. The user must be able to authenticate to the LASR server. The Epa_cars sample data is available from the SAS Visual Analytics 7.3 documentation page at http://support.sas.com/documentation/onlinedoc/va/index.html.

```
caslib valasr datasource=(
   srctype="LASR"
   port="6721"
   server="grid001.example.com"
   username="myusr1"
   password="mypwd1"
   signer="server.example.com/SASLASRAuthorization"
   tag="vapublic"
   /* additional data source options */);
```

Example 2: Load SAS LASR Analytic Server Data into SAS Cloud Analytic Services Using PROC CASUTIL

```
proc casutil;
   list files;                                                  /* 1 */

   load casdata="epa_cars" casout="epa_cars"
        importoptions=(
           filetype="lasr"
           varchars="true"
           vars=("model_year" "vehicle_manufacturer_name")   /* 2 */
        );

   contents casdata="epa_cars";
quit;
```

1 The LIST FILES statement shows the tables that are in SAS LASR Analytic Server.

2 The VARS= option is used to subset the columns that are read from SAS LASR Analytic Server and loaded into SAS Cloud Analytic Services.

6

Platform File Types

Dictionary 153
Delimited Files (CSV) File Type 153
Parquet Tables 157
SASHDAT Files 164

Dictionary

Delimited Files (CSV) File Type

Specifies the file type options for loading data from delimited files.

Valid in: LOAD CASDATA= statement for the CASUTIL procedure

Example: Load a Latin1 encoded CSV file into SAS Cloud Analytic Services.

```
proc casutil;
   load casdata="iris.csv" casout="iris"
        importoptions=(filetype="csv" encoding="latin1");
run;
```

Syntax

PROC CASUTIL;

LOAD CASDATA="*file-name*" CASOUT="*table-name*"

IMPORTOPTIONS=(FILETYPE="CSV" *file-type-options*) <*load-statement-options*>;

Options for the CSV File Type

ARCHIVETYPE="*string*"
specifies the compressed CSV archive type. Valid archive types are "gz" and "zip".

Restrictions GZ archives support only 1 CSV per file. GZ archives that contain multiple files are not supported.

Archive CSV files are not supported for the ADLS or Google Cloud Storage platform data sources.

DELIMITER="*string*"
specifies the character to use as the field delimiter.

Default ","

ENCODING="*string*"
specifies the text encoding of the file. If the file is not encoded in UTF-8 or 7-bit ASCII, then specify the encoding.

Default utf-8

FILETYPE="CSV"
specifies the file type.

Requirement The FILETYPE= option is required.

GETNAMES=TRUE | FALSE
when set to True, the values in the first line of the file are used as variable names.

Default TRUE

GUESSROWS=*integer*
specifies the number of rows to scan in order to determine data types for variables.

Default 20

LOCALE="*string*"
specifies the locale for interpreting data in the file.

MULTIFILE="*string*"
when set to TRUE, the server attemptes to load all CSV files from the specified path into a single in-memory table. The files must have the same number of columns and the same colmn data types.

Default FALSE

Restriction The Google Cloud Storage platform data source does not support loading multiple CSV files.

STRIPBLANKS=TRUE | FALSE
removes leading and trailing blanks from character variables.

Default FALSE

VARCHARS=TRUE | FALSE
when set to True, variable-length strings are used for character variables.

Default TRUE

VARS=((*casvardesc-1*) < , (*casvardesc-2*, ...) >)
specifies the names, types, formats, and other metadata for variables.

FORMAT="*string*"
specifies the format to apply to the variable.

FORMATTEDLENGTH=*integer*
specifies the format field length plus the format precision length.

LABEL="*string*"
specifies the descriptive label for the variable.

LENGTH=*integer*
specifies the unformatted length of the variable. This option applies to fixed-length character variables (type="CHAR") only.

Default 8

NAME="*string*"
specifies the name for the variable.

NFD=*integer*
specifies the format precision length.

NFL=*integer*
specifies the format field length.

TYPE="CHAR" | "DOUBLE" | "VARCHAR"
specifies the data type for the variable.

Details

These platform data sources can read delimited files from caslibs.

- ADLS
- DNFS
- GCS
- PATH
- Amazon S3

Example

By default, SAS Cloud Analytic Services expects to find column names in the first line of the file. If you have a file that does not include names, you must specify GETNAMES=FALSE. You might also prefer to specify column names when you load the data.

Input Data 6.1 *Sample File Contents*

```
Masculin;André;14,00;69,00;112,50
Masculin;Benoît;14,00;63,50;102,50
Masculin;Kévin;12,00;57,30;83,00
```

Example Code 6.1 *Load a CSV File and Specify Column Names*

```
options validvarname=any;                                          /* 1
*/

cas casauto sessopts=(caslib="casuser");                           /* 2
*/

proc casutil;
  load casdata="class_fr.csv" casout="class_fr"
       importoptions=(                                             /* 3
*/
          filetype="csv"
          encoding="utf8"
          delimiter=";"
          getnames=false
          locale="Fr_fr"
          vars=("sexe", "nom", "âge", "la taille", "poids")
       );

  contents casdata="class_fr";
quit;

libname mycas cas caslib="casuser";                                /* 4 */

proc print data=mycas.class_fr(obs=3);
run;
```

1 The VALIDVARNAME= system option is set to ANY so that column names can include national characters.

2 The CAS session, named Casauto, is set to use the Casuser caslib as the active caslib.

3 The IMPORTOPTIONS= option is used to describe how the CASUTIL procedure should read the Class_fr.csv file, including the encoding, locale, and column names to use for the table. The locale option is effective only when reading a CSV file from a caslib with a data source of PATH or DNFS. In this example, the locale enables CAS to use a comma as the decimal separator.

4 The Mycas libref is assigned to use the CAS engine. The CASLIB= option is used to bind the libref to the Casuser caslib.

Output 6.1 *Column Information from the CONTENTS Statement*

Column Information for CLASS_FR in Caslib CASUSER(sasdemo)					
Column	Type	Length	Formatted Length	Format Width	Format Decimal
sexe	varchar	8	8	0	0
nom	varchar	7	7	0	0
âge	double	8	12	0	0
la taille	double	8	12	0	0
poids	double	8	12	0	0

Output 6.2 *Three Rows from the Class_Fr Table*

Obs	sexe	nom	âge	la taille	poids
1	Masculin	André	14	69.0	112.5
2	Masculin	Benoît	14	63.5	102.5
3	Masculin	Kévin	12	57.3	83.0

Parquet Tables

Enables you to transfer data between Parquet tables and CAS.

Valid in: CASLIB statement, PROC CASUTIL: DELETESOURCE, LOAD, and SAVE statements

Requirements: For CAS to read these Parquet tables, they must have a .parquet extension:

- a single file that contains a Parquet table
- a directory of individual Parquet files that together comprise a unified Parquet table
- a single Amazon S3 object that contains a Parquet table
- a group of individual Amazon S3 objects that have the same naming prefix, where each object contains part of the unified Parquet table

For CAS to read Parquet tables that do not have a .parquet extension, you must use IMPORTOPTIONS=.

To load a parquet table into CAS, all part files must have the same schema.

When connecting to Azure Data Lake Storage Gen2, authentication is required. For information about authentication methods, see "Authentication" on page 138.

Note: Parquet tables consisting of a single file (mytable.parquet) loaded to CAS via PATH caslibs and are not distributed among multiple worker nodes, even in MPP-CAS configurations. They are processed and retained on a Single worker node. Multi-

part Parquet tables (mytable/part1.parquet, mytable/part2.parquet...) are not affected.

Syntax

CASLIB *caslib-reference-name*

DATASOURCE=(SRCTYPE="ADLS" | "DNFS" | "GCS" | "PATH" | "S3")
PATH="*/directory-path*" <*caslib-statement-options*>;

PROC CASUTIL;

DELETESOURCE CASDATA="*file-name*" <INCASLIB="*caslib*"> <*deletesource-statement-options*>;

PROC CASUTIL;

LOAD CASDATA="*file-name*" CASOUT="*table-name*" <*load-statement-options*>;

IMPORTOPTIONS=(FILETYPE="PARQUET") <*load-statement-options*>;

PROC CASUTIL;

SAVE CASDATA="*table-name*" CASOUT="*file-name*" <*save-statement-options*>;

Options for Parquet Tables

FILETYPE="PARQUET"
specifies the data source.

Requirement The FILETYPE= option is required.

SRCTYPE="ADLS" | "DNFS" | "GCS" | "PATH" | "S3"
specifies that the data source is a platform data source

- ADLS: a Microsoft Azure Data Lake Storage environment
- DNFS: a directory that is mounted by every machine that is used for SAS Cloud Analytic Services
- GCS: a Google Cloud Storage environment
- PATH: a directory that is accessible to the SAS Cloud Analytic Services controller
- S3: an Amazon S3 environment

Requirement The SRCTYPE= option is required for the CASLIB statement.

VARCHARS=TRUE | FALSE
variable-length strings are used for character variables, when set to True.

Default TRUE

Details

Platform Data Sources for Reading Parquet Tables

Parquet files can be read from caslibs with these platform data sources.

- ADLS
- DNFS
- GCS
- PATH
- S3

Advantages of Using Parquet

Here are some key reasons why you might want to use Parquet.

- Portability. Parquet is portable with other applications and analytics software components, particularly Apache Spark, Open Source, and the cloud ecosystem.
- Reduced data storage size. Parquet tables are often two to ten times smaller than the same data that is stored in other formats.
- Improved runtime performance. Frequently, only a subset of table columns are required for analysis. In such cases, referencing .parquet files directly with CAS actions can often significantly reduce random access memory (RAM) usage and CAS action elapsed runtime. This is true because CAS reads only the Parquet columns that are specified for analysis using optimized memory-buffering techniques.

Supported Parquet Data Types

The table below shows the data types that you can load from Parquet into CAS. It also shows the resulting data type for the data after it has been loaded into CAS, including the mapping between Parquet primitive and logical types with those in CAS. (Parquet logical data types extend the various Parquet primitive data types.) The length of the data format in CAS is based on the length of the source data.

CAUTION
Parquet LISTS and MAPS are not supported, and you should not save or read Parquet files with list or MAP arrays. Using a LIST or MAP logical type can result in loss of data.

Table 6.1 *Supported Parquet Data Types*

Parquet Primitive Data Type	Parquet Logical Type	CAS Data Type
	Character	
BINARY (BYTE_ARRAY)	NONE	VARBINARY

Parquet Primitive Data Type	Parquet Logical Type	CAS Data Type
	STRING or UTF8	VARCHAR
FIXED_LENGTH_BYTE_ARRAY	NONE	BINARY
	DECIMAL*(p,s)*	DOUBLE
	Numeric	
BOOLEAN	NONE	INT32
DOUBLE	NONE	DOUBLE
FLOAT	NONE	DOUBLE
INT32	NONE	INT32
	DECIMAL	DOUBLE
	INT(bitWidth=8|16|32,isSigned=true|false)	INT32
INT64	NONE	INT64
	DECIMAL	DOUBLE
	INT(bitWidth=64,isSigned=true|false)	INT64
	Date and Time	
INT32	DATE	DOUBLE (formatted as DATE)
INT64	TIMESTAMP(isAdjustedToUTC=true|false, unit=MILLIS)	DOUBLE (formatted as DATETIME)
	TIMESTAMP(isAdjustedToUTC=true|false, unit=MICROS)	
	TIME(isAdjustedToUTC=true|false, unit=MILLIS)	
	TIME(isAdjustedToUTC=true|false, unit=MICROS)	

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type.

CAS processes Parquet integer types directly when it can, or they are converted to a SAS DOUBLE during processing. Date, time, and datetime types are converted to SAS DOUBLE types as the Parquet table is read into CAS, before data is printed or saved. This conversion happens regardless of whether the data is loaded into a CAS in-memory table or read directly by a CAS analytics action.

A SAS DOUBLE value maintains approximately 15 digits of precision. When you read values that contain more than 15 decimal digits of precision into a DATA step, the data is converted to a DOUBLE value. When you read these values into a procedure that is not included with the SAS Viya platform, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Access a Parquet Table Using DNFS

```
caslib CAS_DNFS desc='DNFS Parquet Caslib'
   datasource=(srctype='dnfs')
               name="DNFS_PARQ"
               path="/mydir/mycustomer/"                    1
               session=false
               subDirectories=true;

proc casutil;
   load casdata="dnfsfile.parquet" casout="dnfsParqInMem";  2
quit;

proc casutil;
   save casdata="dnfsParqInMem" casout="dnfsParqOut.parquet"; 3
quit;
```

1 Set the PATH= data source caslib definition.

2 Using the above CAS_DNFS (default) caslib definition with PROC CASUTIL to load dnfsfile.parquet results in an in-memory CAS table that is ready for processing by subsequent CAS actions.

3 Save the CAS in-memory table to a .parquet table in the /mydir/mycustomer/ path directory that was defined in the caslib.

Example 2: Access a Parquet Table Using PATH

You can use a PATH caslib to access Parquet tables that are in a local or shared network storage location.

```
caslib CAS_PATH desc='Parquet caslib'
   datasource=(srctype='path')                                    1
               path="/mytest/customer";

proc casutil;
   load casdata="cars.parquet" casout="carsInMemory";             2
run;

proc casutil;
   save casdata="carsInMemory" casout="carsFile.parquet";         3
run;

proc casutil;
    contents casdata="carsFile.parquet";                          4
run;

proc casutil;
    contents casdata="carsInMemory";                              5
run;
```

1 Set the PATH= data source.

2 Load a Parquet table from the path into CAS using the above CAS_PATH (default) caslib definition.Running this PROC CASUTIL results in an in-memory CAS table named carsInMemory that is ready for processing by CAS actions.

3 The SAVE statement in PROC CASUTIL creates a permanent copy of a CAS in-memory table to a .parquet table at the path location that was defined in the CAS_PATH caslib.

4 The CONTENTS statement in PROC CASUTIL displays such table metadata as column names and data types for files.

5 An alternative CONTENTS statement in PROC CASUTIL displays the contents for CAS in-memory tables.

Example 3: Access a Parquet Table Using Azure Data Lake Storage (ADLS) with Single Sign-on

```
cas casauto;

caslib CAS_ADLS
   datasource=(srctype="adls"                                     1
               accountname="myaccount"
               filesystem="dir");
```

```
proc casutil;
   list files;                                                    2
quit;

proc casutil;
   load casdata="somefile.parquet" casout="somefileInMem";        3
quit;

proc casutil;
   load casdata="subdir/otherfile.parquet" casout="otherfileInMem";
quit;

proc casutil;
   save casdata="somefileInMem" casout="file3.parquet";
quit;

proc casutil;
   load data=sashelp.class casout="class" outcaslib="CAS_ADLS";   4
   save casdata="class" casout="classNew.parquet";
   droptable casdata="class";
quit;
```

1 Set the ADLS data source caslib definition.

2 List the tables in the data source.

3 Load the files from a Parquet table into CAS and save the in-memory file to a .parquet table using the default CAS_ADLS caslib.

4 Load a table from ADLS, save it as a Parquet table in an ADLS caslib, and drop the CAS table.

Example 4: Add a Google Cloud Storage Caslib

```
caslib CAS_GCS
   datasource=(srctype="gcs"                                      1
               bucket="mygcsbucket"
               gcsAuthFile="/dir/filename",);

proc casutil;
load data=sashelp.class casout="class" outcaslib="CAS_GCS";       2
save casdata="class" casout="classNew.parquet";                   3
droptable casdata="class";
quit;
```

1 Set the Google Cloud Storage data source caslib definition.

2 Load the table sashelp.class into CAS_GCS. Call the new table class.

3 Save the new table class to a .parquet table in the /dir/filename Google Cloud Storage authentication file that was defined in the caslib, and drop the CAS table.

SASHDAT Files

Specifies the file type options for loading data from SASHDAT files.

Applies to: CASUTIL procedure

Example: Load an encrypted SASHDAT file into SAS Cloud Analytic Services.

```
proc casutil;
   load casdata="cars.sashdat"
        importoptions=(filetype="hdat" encryptionPassword="pasquotank");
run;
```

Syntax

PROC CASUTIL;

LOAD CASDATA="*file-name*" CASOUT="*table-name*"

IMPORTOPTIONS=(FILETYPE="HDAT" *file-type-options*) <*load-statement-options*>;

Options for the HDAT File Type

ENCRYPTIONPASSWORD="*string*"
specifies a password for encrypting or decrypting stored data. You can use this option to override an ENCRYPTIONPASSWORD= value that is set with the CASLIB statement.

FILETYPE="HDAT"
specifies the file type.

Requirement The FILETYPE= option is required.

Details

These platform data sources can read SASHDAT files from caslibs.

- DNFS
- PATH
- S3

Instead of specifying the ENCRYPTIONPASSWORD= option on a file-by-file basis, you can use it as a data source option when you add a caslib. For an example, see Encrypt Tables in a Caslib on page 762.

See Also

The SAVE example for the Path data source on page 147 shows how to work with SASHDAT files.

7

Data Connectors

Working with SAS Data Connectors **168**
Accessing and Transferring Data 168
Case Sensitivity with Object Names 168
Security Considerations 168
Data Type Considerations 168
Passing WHERE Clauses That Contain Date or Datetime Values to the DBMS . . . 168

Quick Reference for Data Connector Syntax **170**

Where to Specify Data Connector Options **175**

Using Wildcards for Pattern Matching **176**
Overview of Wildcards 176
Using Wildcards with SAS Macros 176

Using Data Transfer Modes with Data Connectors for DBMS Data Sources **177**
Overview 177
How to Implement Data Transfer Modes 177
Where to Specify Data Transfer Modes 177
About Loading Data When Using Multinode Data Transfer 177

Dictionary **179**
Amazon Redshift Data Connector 179
DB2 Data Connector 188
DuckDB Data Connector 193
Google BigQuery Data Connector 199
Greenplum Data Connector 207
Hadoop Data Connector 212
Impala Data Connector 221
Informix Data Connector 227
JDBC Data Connector 232
Microsoft SQL Server Data Connector 237
MongoDB Data Connector 249
MySQL Data Connector 255
Netezza Data Connector 261
ODBC Data Connector 265
Oracle Data Connector 270
ORC Data Connector 276
PC Files Data Connector 281
PostgreSQL Data Connector 284

Salesforce Data Connector 292
SAP HANA Data Connector 296
SAP IQ Data Connector 301
SAS Data Sets Data Connector 306
SAS Federation Server Data Connector 310
SAS SpeedyStore Data Connector 315
SingleStore Standard Data Connector 333
Snowflake Data Connector 341
Spark Data Connector 351
SPD Engine Files Data Connector 359
Teradata Data Connector 364
Vertica Data Connector 372
Yellowbrick Data Connector 377

Working with SAS Data Connectors

Accessing and Transferring Data

To access and transfer data, an administrator must configure data connectors for SAS Cloud Analytic Services (CAS). Data connectors contain connection information and data-source specifics to connect with data sources, such as Oracle or SAS data sets. For instructions on how to configure data connectors, see the *SAS Viya Platform: Deployment Guide*.

There are two methods to transfer data between your data source and CAS. You can transfer data serially or you can transfer data in parallel. To transfer data in parallel, you must be using SAS Data Accelerator (known as SAS In-Database Technologies prior to 2026.08), an administrator must have deployed SAS Embedded Process, and you must be accessing a data source that can utilize parallel data transfer. If parallel data transfer is not available for your data source, most data sources can use multinode data transfer, which is almost as efficient as transferring data in parallel. For more information, see "Using Data Transfer Modes with Data Connectors for DBMS Data Sources" on page 177.

Case Sensitivity with Object Names

How a data source supports case sensitivity is reflected in how its corresponding data connector supports case sensitivity.

In general, for all data connectors that support case sensitivity, the data connector quotes everything it passes down. Therefore, if a data source supports case sensitivity, everything is case sensitive, and its data connector has no option to turn off case sensitivity.

For a data source such as Microsoft SQL Server that does not support case sensitivity, case also does not matter for the Microsoft SQL Server data connector. Although Teradata is case insensitive with respect to object name searches, the original case in which the object was saved is retained for display.

Security Considerations

You can associate data connectors that require logins with a domain on the CAS server. When users connect to the data source through a data source name (DSN), the domain name is used to retrieve user credentials that are associated with that data connector. The credentials are then passed to the third-party data source. User credentials are stored on your system and are accessible by the CAS server.

Data connectors can also contain optional information to control CAS data source behavior. Data connectors form the foundation for connectivity to a third-party database or data source. You can assign privileges that control user access to the data. However, relational databases provide authorization that limits the operations that can be performed on the data. CAS respects authorizations that are defined and enforced by a third-party database or data source. Authorizations that are defined on a third-party database overrule permissions and privileges that are set in CAS.

Data Type Considerations

If you save the contents of a CAS table to a table in an external data source, the resulting database table might contain different data types than the CAS table. Any formats that have been applied in the SAS Viya platform that cannot be mapped to an equivalent format in your data source are not preserved. Verify that your data is what you expect after writing data from the SAS Viya platform to a DBMS or other data source. For specifics about data type conversions between the SAS Viya platform and your data source, refer to the data type information for your data connector.

Passing WHERE Clauses That Contain Date or Datetime Values to the DBMS

When you include date or datetime comparisons in WHERE clauses, you want to ensure that these comparisons are passed to the data source. In this way, processing occurs in the date source. Otherwise, data must be transferred to SAS before the WHERE clause is processed. This can substantially increase processing time.

Select one of these methods to ensure that WHERE clause processing with dates or datetime values occurs in the data source. These examples use a datetime value, D, in a WHERE clause:

- Use FedSQL syntax for the datetime value, such as in this sample:

```
WHERE= "age = 11 AND d < TIMESTAMP'2040-11-06 00:00:00'"
```

- Use data source syntax in the date or datetime value with DATASOURCEOPTIONS=, such as in the LOAD statement for PROC CASUTIL. Here is an example that uses data source syntax to call the Teradata TO_DATE function:

```
DATASOURCEOPTIONS = {where="age=12 and
                                d < TO_DATE('06nov2040:00:00:00', 'ddmonyyyy:hh24:mi:ss',
                                'NLS_DATE_LANGUAGE=American')"}
```

Note: Attempting to use data source syntax outside of DATASOURCEOPTIONS= results in an error.

You want to avoid using SAS syntax in a WHERE clause, because this can result in transferring data into SAS before the WHERE clause is processed in SAS. This example completes, but first loads your data into SAS:

```
WHERE="age=13 AND d < '06NOV2024:00:00:00'DT"
```

TIP Specify `MSGLEVEL=i` in the OPTIONS statement to see where your query was processed.

See Also

- "LOAD Statement" on page 73
- *SAS Viya Platform: FedSQL Programming for SAS Cloud Analytic Services*

Quick Reference for Data Connector Syntax

This table shows the syntax, supported file types (if applicable), and an example for each data source. For path-based data source syntax, see the dataSource= parameter for the CASLIB statement on page 745.

Table 7.1 Data Source Types and Options

Data Connector	srcType= Type	Option Syntax	Example
Amazon Redshift	**redshift**	"Amazon Redshift Data Connector" on page 179	caslib RScaslib desc='Amazon Redshift Caslib' dataSource=(srctype='redshift', server='RSserver' username='user1', password='myPwd', database='rsdatabase');
Cloud Data Exchange	**clouddex** **cde**	"Add a Caslib to Data Agent" in *Cloud Data Exchange for the SAS Viya Platform: Administrator's Guide*	caslib cde_caslib global datasource=(srcType="clouddex", username="myuser1", password="myPwd", port="1337", server="data-agent-host.url", catalog="GRIDLIB", schema="model" conopts="dsn=dsn_tera");
DB2	**db2**	"DB2 Data Connector" on page 188	caslib mycaslib desc='DB2 Caslib' datasource=(srctype='db2' username='myusr1' password='myPwd' database="sample");
Google BigQuery	**bigquery**	"Google BigQuery Data Connector"	caslib gbqcaslib desc='Google BigQuery Caslib' dataSource=(srctype='bigquery' credfile= '/u/authfiles/BigQuery/xxx-yyy-1a2b3333c4444.json' schema='mySchema' project='project1' mode="performance" threadedLoad=true); **Note:** If you are loading small tables, you might see better performance if you disable the threadedLoad= option.
Greenplum	**greenplum**	"Greenplum Data Connector"	caslib greenplumcaslib desc='Greenplum Caslib' dataSource=(srctype='greenplum' server='GPserver' username='user1' password='myPwd' database="GPdatabase-name" schema='myschema');
Hadoop	**hadoop**	"Hadoop Data Connector" on page 212	caslib hvcaslib desc="Hadoop Caslib" datasource=(srctype="hadoop", dataTransferMode="serial", hadoopjarpath="/data/cdh58/sdm/lib", hadoopconfigdir="/data/cdh58/sdm/conf", username="myuser1", server="hive01.example.com", schema="default");

Data Connector	srcType= Type	Option Syntax	Example
Impala	**impala**	"Impala Data Connector"	caslib imcaslib desc="Impala Caslib" datasource=(srctype="impala", username="myuser1", server="impala01.example.com");
Informix	**informix**	"Informix Data Connector"	caslib Infcaslib desc="InformixToCaslib" datasource=(srctype="informix" host='informix-host' port=14463 server="DBMS-server-on-host" database="database-name" username="user1" password="password1");
JDBC	**jdbc**	"JDBC Data Connector" on page 232	caslib mycaslib desc="JDBC Caslib" datasource=(srctype='jdbc', url="jdbc:myDriver://<connection-options>", username='myuser1', password='myPwd', class="JDBC-driver-class", classpath="<path-to-JDBC-driver-class>");
Microsoft SQL Server	**sqlserver**	"Microsoft SQL Server Data Connector" on page 237	caslib mycaslib desc='Microsoft SQL Server Caslib' dataSource=(srctype='sqlserver' username='myuser1' password='myPwd' sqlserver_dsn="SQLserverDSN-name");
MongoDB	**mongodb**	"MongoDB Data Connector" on page 249	caslib mycaslib desc='MongoDB Caslib' dataSource=(srctype='mongodb' server='mymachine.com' username='myuser1' password='myPwd' db="database-name" schema_collection="mySchema");
MySQL	**mysql**	"MySQL Data Connector" on page 255	caslib mycaslib desc='MySQL Caslib' datasource=(srctype='mysql', host='mySQLserver', username='myuser1', password='myPwd', database="<database-name>", schema='myschema');
Netezza	**netezza**	"Netezza Data Connector"	caslib netezzacaslib desc='Netezza Caslib' dataSource=(srctype='netezza' server='NZserver' username='user1' password='myPwd1' database="NZdatabase-name" schema='myschema');

Data Connector	srcType= Type	Option Syntax	Example
ODBC	**odbc**	"ODBC Data Connector" on page 265	`caslib mycaslib desc="ODBC caslib" datasource=(srctype="odbc" username="myuser1" password="myPwd" database="dbodbc");`
Oracle	**oracle**	"Oracle Data Connector" on page 270	`caslib mycaslib desc="Oracle Caslib" datasource=(srctype="oracle", username="myuser1", password="myPwd", path="//machine.lnx.com:1521/exadat");`
ORC	path	"ORC Data Connector" on page 276	`caslib mycaslib desc='ORC Caslib' datasource=(srctype='path') path="/mytest/customer";`
PC Files	**path**	"PC Files Data Connector" on page 281	`caslib mycaslib desc='PCFiles Caslib' datasource=(srctype='path') path="/mytest/customer";`
PostgreSQL	**postgres**	"PostgreSQL Data Connector" on page 284	`caslib mycaslib desc='PostgreSQL Caslib' datasource=(srctype='postgres' server='PGserver' username='myuser1' password='myPwd' database="PGdatabase-name");`
Salesforce	**sforce**	"Salesforce Data Connector" on page 292	`caslib mycaslib desc='Salesforce Caslib' dataSource=(srctype='sforce' username='myuser1' password='myPwd');`
SAP HANA	**hana**	"SAP HANA Data Connector" on page 296	`caslib hacaslib sessref=mysess datasource=(srctype='hana', server='hana server', username='myuser1', password='myPwd', tableType='column' schema='hana schema' instance='00');`
SAP IQ	**sapiq**	"SAP IQ Data Connector"	`caslib sapiqcaslib desc="SAPIQtoCaslib" datasource=(srctype="sapiq" server="DBMS-server-on-host" host="SAPIQ-host" database="database-name" username="myuser1" password="myPwd");`
SAS Data Sets	**path**	"SAS Data Sets Data Connector" on page 306	`caslib mycaslib desc='SASDataSets Caslib' datasource=(srctype='path') path="/mytest/customer";`

Data Connector	srcType= Type	Option Syntax	Example
SingleStore[1]	**singlestore**	"SAS SpeedyStore Data Connector"	caslib singlestorelib datasource=(srctype='singlestore' database='my_database' epDatabase='metadata_database' host='svc-sas-singlestore-cluster-dml' pass='my_password' port='3306' user='my_username');
SingleStore standard	**singlestore_standard**	"SingleStore Standard Data Connector"	caslib mycaslib desc='SingleStore Caslib' dataSource=(srctype='singlestore_standard' host='server-name' username='user1' password='myPwd' database="database-name");
Snowflake	**snowflake**	"Snowflake Data Connector"	caslib snowcaslib desc='Snowflake Caslib' dataSource=(srctype='snowflake' server='SFserver' username='myuser1' password='myPwd' schema='mySchema' database="sfdatabase-name");
Spark	**spark**	"Spark Data Connector" on page 351	caslib sparklib desc="Spark Caslib" datasource=(srctype="spark", datatransfermode='serial', hadoopjarpath= "data/cdh58/sdm/lib/spark:/data/cdh58/sdm/lib", hadoopconfigdir="/data/cdh58/sdm/conf", username="myuser1", server="hive01.example.com", schema="default");
SPD Engine Files	**spde**	"SPD Engine Files Data Connector" on page 359	caslib mycaslib desc='SPD Engine Files Caslib' dataSource=(srctype='spde', hadoopjarpath="/data/cdh58/sdm/lib", hadoopconfigdir="/data/cdh58/sdm/conf", username="myuser1", mdfpath="/user/MyMDFPath", hdfs=true,host="default");
Teradata	**teradata**	"Teradata Data Connector" on page 364	caslib TDcaslib desc='Teradata Caslib' datasource=(srctype='teradata' username='myuser1' password='myPwd' server="teradataServer");
Vertica	**vertica**	"Vertica Data Connector" on page 372	caslib VTcaslib desc='Vertica Caslib' datasource=(srctype='vertica' server='VTserver' username='myuser1' password='myPwd' database="VTdatabase-name" schema='myschema');

Data Connector	srcType= Type	Option Syntax	Example
Yellowbrick	**yellowbrick**	"Yellowbrick Data Connector" on page 377	caslib ybrickcaslib desc='Yellowbrick Caslib' dataSource=(srctype='yellowbrick' server='YBserver' username='user1' password='myPwd' database="YBdatabase-name" schema='myschema');

1 Use the SingleStore data connector when you are using SAS with SingleStore and when your data source is integrated with SAS with SingleStore. Otherwise, use the SingleStore standard data connector.

Where to Specify Data Connector Options

The options that you specify for your data connector are specific to your data source. In general, you specify options for data connectors in your SAS code as described in this section.

Note: You can specify data connector options for all data connectors except SAS Data Connector to PC Files or SAS Data Connector to SAS Data Sets. For these you would instead use importOptions= to read data and exportOptions= to save data.

When you add a caslib, specify data connector options within the dataSource= option. Use the dataSource= option in the CASLIB statement. If the server is named Teraserver, the database is named Teradatabase, and a schema is named MySchema, here is how the caslib can be added.

```
/* CASLIB statememt */
caslib simple
  sessref=mysess
  dataSource=(srcType="teradata",
              dataTransferMode="parallel",
              server="teraserver",
              database="teradatabase",
              schema="mySchema");
```

For PROC CASUTIL, specify data connector options for dataSourceOptions= in a CONTENTS, LIST, LOAD, or SAVE statement.

```
proc casutil;
   list files incaslib="tdlib" dataSourceOptions=(schema="tdSchema");
   contents casdata="cars" vars((name="make"), (name="model"))
      dataSourceOptions=(schema="tdSchema2");
   load casdata="cars" incaslib="tdlib" casout="cars_CAS"
      vars((name="make"), (name="model"))
```

```
      dataSourceOptions=(statusinterval=5);
   save casdata="cars" incaslib="tdlib" casout="cars_CAS"
      dataSourceOptions=(schema="tdSchema");
quit;
```

For more information about using SAS code that is related to data connectors, see these sections.

- CASLIB Statement
- CASUTIL Procedure

You can also specify data connector options using CAS actions. For more information, see "Table Action Set" in *SAS Viya Platform: System Programming Guide*.

Using Wildcards for Pattern Matching

Overview of Wildcards

There are instances when you might use wildcards to facilitate pattern matching. For example, you might use wildcards to see a listing of all tables that begin with the string "aBc". To do this, you might specify this pattern to match:

```
path='aBc%'
```

You can use wildcards in the path= option for the fileInfo action. Case sensitivity is based on the behavior of your data source. You can use multiple wildcard characters within a pattern.

Here are the available wildcard characters:

%	matches any number of characters.
_	matches a single character. You can include more than one '_' in a pattern.
\	escapes a wildcard character so that it is treated as a literal character instead of a wildcard in a pattern.

If you have file names that contain underscores, escape the '_' character with the '\'. For example, this pattern searches for all files that begin with 'aBc_':

```
path='aBc\_%'
```

Using Wildcards with SAS Macros

The SAS Macro language also uses the '%' character to indicate variables to resolve during program compilation. When you use the '%' character for pattern matching,

include the value in single quotations marks. If you enclose a value in double quotation marks, the SAS Macro parser tries to resolve a macro variable name. For example, the code `path="testmisc/prdsa%e.csv"` results in an attempt to resolve a macro variable, E, and typically results in an error. Instead, specify the code as `path='testmisc/prdsa%e.csv'`.

Using Data Transfer Modes with Data Connectors for DBMS Data Sources

Overview

In CAS, data connectors provide access to data in a database management system (DBMS) or distributed system using one of these modes of data transfer: serial or parallel.

In *serial* data transfer mode, the data connector makes a single connection to the data source using a single data stream to the DBMS. This allows Read and Write access to data.

Most data connectors that use serial data transfer mode can take advantage of *multinode* data transfer. Multinode data transfer uses multiple CAS nodes to connect to a data source, which helps speed performance. A CAS controller node controls data transfer to and from worker nodes through **n** concurrent connections with the data source. This controller node directs each CAS worker node on how to query the source data to obtain the needed data. Each worker node therefore connects directly to the data source independently. As a result, multiple data streams can move data simultaneously.

Note: Multinode Read is not allowed with the FedSQL execDirect action. The load proceeds with numReadNodes=1.

In *parallel* data transfer mode, each CAS node connects directly to a data source node for parallel Read and Write. To use this mode, you must have access to SAS Data Accelerator (known as SAS In-Database Technologies prior to 2026.08) and an administrator must have deployed SAS Embedded Process. Data access is fastest with this method, as SAS Embedded Process can connect to individual slices of data on a data source node.

For details about CAS nodes, see these documents.

- *SAS Viya Platform: SAS Cloud Analytic Services*
- *SAS Cloud Analytic Services: Fundamentals*

How to Implement Data Transfer Modes

By default, most SAS data connectors use serial data transfer. To use multinode data transfer, specify a value other than 1 for the numReadNodes= and numWriteNodes= options.

To specify how many nodes to use for serial data transfer using multiple worker nodes, you specify a specific value.

- Set numReadNodes= and numWriteNodes= to a value other than 1 to enable multinode data transfer.
- Multinode can use only the number of nodes that are available. So if you specify five nodes but only four are available, then four nodes are used.
- If you specify numReadNodes=0 and numWriteNodes=0, then all available nodes are used for multinode data transfer.

Data connectors for some data sources use the dataTransferMode= option to specify serial or parallel data transfer. When you specify dataTransferMode="PARALLEL", data transfer is always attempted in parallel. When you specify dataTransferMode="AUTO", SAS attempts data transfer in parallel first and then in serial mode if parallel transfer fails. With serial data transfer, if the values for numReadNodes= and numWriteNodes= are different from 1, then data transfer uses multiple nodes to perform the transfer.

Where to Specify Data Transfer Modes

Two different SAS programming options have the same name: dataTransferMode=.

- You can specify the dataTransferMode= file import option in such SAS procedures as PROC CAS and PROC CASUTIL. Use this option when you are working with SAS tables (SAS7BDAT files) as specified by the PATH type of CASLIB. For SAS tables, the default value is dataTransferMode=AUTO. For any other file types, dataTransferMode= is not an available option, so the result is effectively the same as the SERIAL value.
- You can specify the dataTransferMode= caslib attribute with the srcType= attribute to specify how to transfer data between your data source and the SAS Viya platform. When you set dataTransferMode="SERIAL", a data connector is used to connect to a supported remote data source to transfer data in serial mode. When you set dataTransferMode="PARALLEL", the SAS Embedded Process must be active in a remote data source, and then data is transferred in parallel mode. For most data sources, dataTransferMode= is not an available option. For caslibs with a source type of Hadoop, SPD Engine Files, or Teradata, you can specify the dataTransferMode= option.

About Loading Data When Using Multinode Data Transfer

For multinode data loads, SAS checks the target data source table for a numeric variable. It takes the first numeric variable that it finds and uses those values to divide the table into slices. Division is accomplished by using the MOD function and the number of nodes that you specify. Therefore, the higher the cardinality of the numeric variable, the easier the data can be divided into slices.

Here is the order of data types that SAS uses to divide data into slices for multinode Read.

- INT (includes BIGINT, INTEGER, SMALLINT, TINYINT)
- DECIMAL
- NUMERIC
- DATE or TIMESTAMP
- TIME
- DOUBLE (see Note)

Note: Data of type DOUBLE is not used to partition data for the Hadoop data connector.

SAS chooses only NUMERIC or DECIMAL values that meet specific criteria for slicing data. These values must be formatted so that 0 < width - decimal digits < 19. This ensures that the value can be used as a BIGINT value for the slicing process. For example, a DECIMAL column that is formatted as `21.3` can be used because 21 - 3 = 18, but a DECIMAL column that is formatted as `22.3` could not be used because 22 - 3 = 19. Similarly, a NUMERIC column that is formatted as `18.1` can be used, but a NUMERIC column with format `24.3` could not be used.

Dictionary

Amazon Redshift Data Connector

Enables you to transfer data between Amazon Redshift and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

See: Amazon Redshift LIBNAME statement connection

Examples: Establish a connection between your Amazon Redshift database and SAS Cloud Analytic Services.

```
caslib redshiftcaslib desc='Amazon Redshift Caslib'
     dataSource=(srctype='redshift'
                 server='RSserver'
                 username='user1'
                 password='myPwd'
                 database="rsdatabase-name");
```

Override the user and password values.

```
proc casutil;
   load casdata="rsexamp" casout="myRSdata" incaslib="redshiftcaslib"
      outcaslib="casuser" dataSourceOptions=(
         username='user5'
         password='myPwd');
quit;
```

Details

Data Connector Options for Amazon Redshift

Table 7.2 *Data Connector Options for Amazon Redshift*

Data Connector Options	Default Value
authenticationDomain=	none
authenticationScope=	none
authMethod=	Native
awsAutoCreate=	NO
awsConfig=	**~/.aws/config**
awsDurTime=	900 (15 minutes)
awsIAMassumeRoleARN=	none
blEncKey=	none
blFormat=	"DELIMITED"
blOptions=	none
bucket=	none

Data Connector Options	Default Value
bulkLoad=	FALSE
bulkUnload=	FALSE
catalog=	Active caslib
charMultiplier=	1.0
client_encoding=	UTF-8
clusterId=	none
compress=	FALSE
conOpts=	none
credentialsFile=	none
credentialsProfile=	none
database=	none
dbCreateTableOpts=	none
defaultDir=	*temporary-file-directory* that is specified by the UTILLOC= system option
deleteDataFiles=	TRUE
delimiter=	bell character (ASCII 0x07)
driver_vendor=	"SAS"
escape=	FALSE
iamAssumeRoleARN=	none
iamAssumeRoleName=	none
insertBuff=	Calculated automatically based on row size
key=	none
numReadBuffers=	4
numReadNodes=	0 (use all nodes)

Data Connector Options	Default Value
numWriteNodes=	0 (use all nodes)
password=	none
profile=	default
readBuff=	Calculated automatically based on row size
redshift_dsn=	none
region=	none
s3Config=	none
schema=	none
secret=	none
server=	none
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "redshift" for Amazon Redshift data.
token=	none
useNarrowCharacterTypes=	FALSE
userName=	none
useSSL=	TRUE

Single Sign-On to Amazon Redshift

The Amazon Redshift data connector supports Single Sign-On (SSO) for connections to Amazon Redshift. SSO uses Microsoft Entra ID as an Identity Provider (IdP) in conjunction with AWS IAM authentication. By using SSO, you can connect to Amazon Redshift without storing passwords in your CASLIB statements.

Your administrator must have configured Microsoft Entra ID as an IdP. You or your administrator also need to set up IAM authentication in your Amazon Redshift data source. See your Amazon Redshift documentation for information about IAM authentication methods.

IAM authentication requires a list of valid Certificate Authorities. This list is stored in the amazon-trust-ca-bundle.crt file that comes with your deployment. If the list needs to be modified, an administrator should edit that file.

These options are required for SSO:

- authenticationScope=. This option specifies the Microsoft Azure resource and permissions for authorizing access.
- authMethod=. The default value is Native. Set authMethod=IAM for SSO to Amazon Redshift.
- awsIAMassumeRoleARN=. This option specifies an Amazon Resource Name that identifies an IAM role.
- clusterId=. Use this option to specify the cluster ID for your Amazon Redshift data.
- database=.
- port=.
- region=.
- server=.
- userName=.

These options are not required, but you can specify them for SSO:

- awsAutoCreate= (default is NO)
- awsDurTime= (default is 15 minutes)

Here's a sample CASLIB statement that uses SSO authentication:

```
caslib rscaslib dataSource=(srctype='redshift',
                username='myuserid', server='myserver.redshift.amazonaws.com',
                database='myuserdb', region='us-east-1',
                authMethod='IAM', clusterId='dbrs1',
                authenticationScope='api://a09827234-93eo-092fjas1-92038498273/awsassumerole',
                awsIAMassumeRoleARN='arn:aws:iam::129347192347098273:role/myrolename');
```

See Also

- "Scenario: OIDC with Microsoft Entra ID" in *SAS Viya Platform: Authentication*
- "Incorporate Additional CA Certificates" in *SAS Viya Platform Encryption: Data in Motion*

Bulk Loading and Unloading Amazon Redshift Data

Overview of Bulk Loading and Unloading Options

Bulk loading and unloading is the fastest way to move data between your Amazon Redshift database and the CAS server. To use the bulk-load facility to transfer data to Amazon Redshift, specify bulkLoad=TRUE. To use the bulk-unload facility to transfer data into CAS, specify bulkUnload=TRUE. Bulk loading and unloading in

Amazon Redshift uses the Amazon Simple Storage Services (S3) tool to transfer large amounts of data.

These data connector options are also related to bulk loading or unloading:

- awsConfig=
- blEncKey=
- blOptions=
- bucket=
- compress=
- credentialsFile=
- credentialsProfile=
- defaultDir=
- deleteDataFiles=
- delimiter=
- escape=
- key=
- profile=
- region=
- s3Config=
- secret=
- token=
- useSSL=

Managing Encryption for Bulk Loading and Unloading in S3 Environments

All Amazon S3 buckets have encryption enabled by default for data that is stored in S3. All new objects that are uploaded are automatically encrypted by using server-side encryption with S3-managed keys (SSE-S3). This encryption uses the AES-256 algorithm and is managed by AWS, which generates and rotates encryption keys. Data is encrypted and decrypted transparently for users.

For MinIO, which is an S3 compatible object storage platform, the default behavior for encryption depends on the configuration for your environment. MinIO supports server-side encryption using AES-256-GCM.

If you do not want AWS to manage your encryption keys, then you can specify your own key values that are used to access encrypted data in an Amazon S3 bucket or MinIO location.

1 Assign a user-friendly name to an encryption key by using the ENCKEY statement for PROC S3.

Note: Only IAM:KMS keys are supported for SAS/ACCESS Interface to Amazon Redshift.

2 Enable bulk loading with the bulkLoad= data connector option, or enable bulk unloading with the bulkUnload= data connector option.

3 Specify the blEncKey= data connector option to indicate the name that is associated with the encryption key for your data. The value for blEncKey= is the user-friendly name that you assigned to an encryption key by using PROC S3.

4 Specify any additional bulk-load options. For more information, see "Overview of Bulk Loading and Unloading Options".

Supported Amazon Redshift Data Types

The following table shows the data types that can be loaded from Amazon Redshift into CAS. This table also shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.3 *Supported Amazon Redshift Data Types*

Amazon Redshift Data Type	CAS Data Type
Character	
CHAR(*n*)	VARCHAR
VARCHAR(*n*)	VARCHAR
Numeric	
BIGINT	INT64
BOOLEAN	DOUBLE
DECIMAL(*p*, *s*)	DOUBLE
DOUBLE PRECISION	DOUBLE
INTEGER	INT32
NUMERIC	DOUBLE
REAL	DOUBLE
SERIAL	DOUBLE
SMALLINT	INT32

Amazon Redshift Data Type	CAS Data Type
Date and Time	
DATE	DOUBLE (formatted as DATE*w.*)
TIMESTAMP(*p*) (without time zone)	DOUBLE (formatted as DATETIME*w.d*)

About CHAR and VARCHAR Data Types

With Amazon Redshift, CHAR and VARCHAR data is defined in terms of bytes instead of characters. A CHAR variable can contain only single-byte characters. This means that a CHAR(10) variable can contain a string with a maximum length of 10 bytes. A VARCHAR variable can contain multiple-byte characters, up to a maximum of four bytes per character. For example, a VARCHAR(12) variable can contain 12 single-byte characters, six two-byte characters, four three-byte characters, or three four-byte characters.

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is

supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Add an Amazon Redshift Database as a Data Source for SAS Cloud Analytic Services

Use the CASLIB statement to establish a connection between your Amazon Redshift source data and a caslib, RScaslib. All of the options supplied in this example are required in the CASLIB statement, except the PASSWORD= option.

In this example, the Amazon Redshift data is stored at the location designated by the SERVER= and DATABASE= options.

```
caslib RScaslib desc='Amazon Redshift Caslib'
     dataSource=(srctype='redshift',
                 server='RSserver'
                 username='user1',
                 password='myPwd',
                 database='rsdatabase');
```

Example 2: Load Amazon Redshift Data into SAS Cloud Analytic Services Using PROC CASUTIL

```
proc casutil;
   list files incaslib="RScaslib";                    1
   load casdata="myRSdata" incaslib="RScaslib" outcaslib="casuser"
        casout="class_from_RScaslib";                 2
   list files incaslib="casuser";                     3
   contents casdata="%upcase(class_from_RScaslib)" incaslib="casuser";
                                                      4
quit;
```

1 List the tables in RScaslib before loading your data.

2 Load the table myRSdata from RScaslib into caslib Casuser. Call the new table class_from_RScaslib.

Note: You must specify Amazon Redshift table names using the capitalization that is used in the database.

3 List the tables in Casuser to see the newly created table, class_from_RScaslib, that you loaded.

4 List information about the newly loaded table, including column names, data types, and so on.

DB2 Data Connector

Enables you to transfer data between DB2 and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

Note: All users with SAS/ACCESS Interface to DB2 can use SAS Data Connector to DB2.

See: DB2 LIBNAME statement

Examples: Establish a connection between your DB2 database and SAS Cloud Analytic Services.

```
caslib db2caslib desc='DB2 Caslib'
   datasource=(srctype='db2'
               username='myusr1'
               password='myPwd'
               database="sample");
```

Override USER= and PASSWORD= values.

```
proc casutil;
   load casdata="%upcase(db2examp)" incaslib="db2caslib"
   dataSourceOptions=(username='user5' password='pwd5');
```

Details

Data Connector Options for DB2

Table 7.4 *Data Connector Options for DB2*

Data Connector Options	Default Value
authenticationDomain=	none
blAllowReadAccess=	FALSE
blCopyLocation=	none
blCpuParallelism=	none
blDataBufferSize=	none
blDiskParallelism=	none
blExceptionTable=	none

Data Connector Options	Default Value
blIndexingMode=	AUTOSELECT
blLoadReplace=	FALSE
blLog=	none
blobThreshold=	0
blOptions=	none
blPortMax=	none
blPortMin=	none
blRecoverable=	FALSE
blRemoteFile=	none
blWarningCount=	2147483646
bulkLoad=	FALSE
catalog=	Active caslib
charMultiplier=	1.0
client_encoding=	none
conOpts=	none
database=	none
dbCreateTableOpts=	none
insertBuff=	Calculated automatically based on row size
insertSQL=	TRUE
invalidDate=	WARN when loading multiple records ERROR when loading records individually
numReadBuffers=	4
numReadNodes=	1
numWriteNodes=	1

Data Connector Options	Default Value
password=	none
readBuff=	Calculated automatically based on row size
schema=	none
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "db2" for DB2 data.
useNarrowCharacterTypes=	FALSE
userName=	none

Bulk Loading Data to DB2

You can use bulk loading functionality to move data efficiently to your DB2 data source. To enable bulk loading, set bulkLoad=TRUE. In addition, these options are also involved with bulk loading to DB2:

- blAllowReadAccess=
- blCopyLocation=
- blCpuParallelism=
- blDataBufferSize=
- blDiskParallelism=
- blExceptionTable=
- blIndexingMode=
- blLoadReplace=
- blLog=
- blOptions=
- blPortMax=
- blPortMin=
- blRecoverable=
- blRemoteFile=
- blWarningCount=
- bulkLoad=

Supported DB2 Data Types

The DB2 data connector supports the following data types.

Table 7.5 *Supported DB2 Data Types*

DB2 Data Type	CAS Data Type
Character	
CHAR(*n*)	CHAR
BLOB	VARCHAR
CLOB	VARCHAR
GRAPHIC	CHAR
GRAPHIC(*n*)	VARCHAR
LONG VARCHAR	VARCHAR
VARGRAPHIC	VARCHAR
LONG VARGRAPHIC	VARCHAR
VARCHAR(*n*)	VARCHAR
Numeric	
BIGINT	INT64
DECIMAL	DOUBLE
DOUBLE	DOUBLE
REAL	DOUBLE
INTEGER	INT32
SMALLINT	INT32
Date and Time	
DATE	DOUBLE (formatted as DATE*w.*)
TIME	DOUBLE

DB2 Data Type	CAS Data Type
	(formatted as TIME*w.d*)
TIMESTAMP	DOUBLE (formatted as DATETIME*w.d*)

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Add a DB2 Database as a Data Source for SAS Cloud Analytic Services

Use the CASLIB statement to establish a connection between your DB2 source data and a caslib, db2lib. All of the options supplied in this example are required in the CASLIB statement.

In this example, DB2 data is stored in the location that the DB= option designates.

```
caslib db2lib
   datasource=(srctype='db2',
               username='myuser1',
               password='myPwd',
               db="myDB2data");
```

Example 2: Load DB2 Data into SAS Cloud Analytic Services Using PROC CASUTIL

```
proc casutil;
   list tables incaslib="casuser";                    1
   load casdata="MYDB2DATA" incaslib="db2caslib" outcaslib="casuser"
        casout="DB2data_from_db2caslib";              2
   list tables incaslib="casuser";                    3
   contents casdata="%upcase(class_from_db2caslib)" incaslib="casuser";
                                                      4
quit;
```

1 List the tables in Casuser before loading your data.

2 Load the table myDB2data from DB2 into caslib Casuser. Call the new table DB2data_from_db2caslib.

Note: You must list DB2 table names with capitalization that matches that in the DB2 database.

3 List the tables in Casuser again to see the newly created table, DB2data_from_db2caslib, that you loaded.

4 List information about the newly loaded table, including column names, data types, and so on.

DuckDB Data Connector

Enables you to transfer data between DuckDB and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

Note: All users with SAS/ACCESS Interface to DuckDB can use SAS Data Connector to DuckDB.

See: DuckDB LIBNAME statement

Example: Establish a connection between your DuckDB database and SAS Cloud Analytic Services.

```
caslib DuckDBcaslib desc='DuckDB Caslib'
   datasource=(srctype='DuckDB'
               database="/mymount/sample.db");
```

Details

Data Connector Options for DuckDB

Table 7.6 *Data Connector Options for DuckDB*

Data Connector Options	Default Value
allowMovedPaths=	TRUE
azureAccountName=	none
azureBlobEndpoint=	none
azureClientId=	none
azureClientSecret=	none
azureDnsSuffix=	none
azureResource=	none
azureSas=	none
azureTenantId=	none
catalog=	Active caslib
charMultiplier=	1.0
client_encoding=	Encoding for the current SAS session
conOpts=	none
copyOptions=	none
csvOptions=	none
database=	none
dbCreateTableOpts=	none
dbType=	none
directoriesAsData=	FALSE
duckdbThreads=	none

Data Connector Options	Default Value
excelOptions=	none
extensionDirectory=	Location that is specified by the UTILLOC system option
filePath=	none
fileType=	DUCKDB
gcsAccessKey=	none
gcsAuthDomain=	none
gcsCredFile=	none
gcsSecret=	none
icebergCatalog=	none
icebergClientId=	none
icebergClientSecret=	none
icebergEndpoint=	none
icebergEndpointType=	none
icebergOauthUri=	none
icebergSecretOptions=	none
icebergToken=	none
ignoreFedSQLObjects=	FALSE
insertBuff=	Calculated automatically based on row size
insertSQL=	TRUE
jsonOptions=	none
max_array_len=	2000
max_binary_len=	2000
max_char_len=	2000

Data Connector Options	Default Value
memoryLimit=	none
numReadBuffers=	4
numReadNodes=	1
numWriteNodes=	1
readBuff=	Calculated automatically based on row size
readOnly=	FALSE
s3AccessKey=	none
s3Region=	none
s3Secret=	none
s3Token=	none
scanStringColumns=	TRUE
schema=	none
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "DUCKDB" for DuckDB data.
tempDirectory=	Location that is specified by the UTILLOC system option
useNarrowCharacterTypes=	FALSE

Supported DuckDB Data Types

The DuckDB data connector supports the following data types.

Table 7.7 *Supported DuckDB Data Types*

DuckDB Data Type	CAS Data Type	Default CAS Format
BIGINT	INT64	20.

DuckDB Data Type	CAS Data Type	Default CAS Format
BLOB	VARCHAR	$HEX*w.*
BOOLEAN	DOUBLE	1.
DATE	DOUBLE	DATE9.
DECIMAL(*prec,scale*)[1]	DOUBLE	*w.d*
DOUBLE	DOUBLE	*w.d*
FLOAT	DOUBLE	*w.d*
HUGEINT	INT64	20.
INTEGER	INT32	11.
INTERVAL	CHAR	$*w.*
JSON	CHAR	$*w.*
SMALLINT	INT32	6.
TIME	DOUBLE	TIME15.6
TIMESTAMP	DOUBLE	DATETIME25.6
TIMESTAMP WITH TIME ZONE	DOUBLE	DATETIME25.6
TINYINT	INT32	4.
UBIGINT	INT64	20.
UHUGEINT	INT64	20.
UINTEGER	INT32	11.
USMALLINT	INT32	6.
UTINYINT	INT32	4.
UUID	CHAR	$*w.*
VARCHAR	VARCHAR	$*w.*

1 References to *prec* and *scale* in DuckDB numeric data types are equivalent to *w* and *d* in SAS formats.

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Connect CAS with DuckDB Data in an S3 Bucket

Use the CASLIB statement to establish a connection between your DuckDB data in an S3 bucket and caslib DuckDBcaslib.

```
caslib DuckDBcaslib desc='DuckDB Caslib'
   datasource=(srctype="duckdb",
               fileType="PARQUET", filePath="s3://mybucket/folder1",
               s3AccessKey="key-value",
               s3Secret="secret-value",
               s3Token="token-value");
```

Example 2: Connect CAS with DuckDB Data on GCS

Use a CASLIB statement to connect CAS with your DuckDB data on Google Cloud Storage.

```
caslib DuckDBcaslib desc='DuckDB Caslib'
```

```
   datasource=(srctype="duckdb",
               fileType="PARQUET", filePath="gs://mybucket/folder1",
               gcsAccessKey="key-value",
               gcsSecret="secret-value",
               gcsCredFile="/mypath/filename.json");
```

Example 3: Connect between CAS and DuckDB Data on Azure

Use a CASLIB statement to connect CAS with your DuckDB data on Microsoft Azure.

```
caslib DuckDBcaslib desc='DuckDB Caslib'
   datasource=(srctype="duckdb",
               fileType="PARQUET", filePath="az://mylocation/",
               azureAccountName='account',
               azureSas="shared-access-signature",
               azureBlobEndpoint='https://
account.blob.core.windows.net',
               azureClientId='client-ID',
               azureTenantId="tenant-ID");
```

Google BigQuery Data Connector

Enables you to transfer data between Google BigQuery and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

See: Google BigQuery LIBNAME statement

Example: Establish a connection between your Google BigQuery database and SAS Cloud Analytic Services. Note: If you are loading small tables, you might see better performance if you disable the threadedLoad= option.

```
caslib gbqcaslib desc='Google BigQuery Caslib'
     dataSource=(srctype='bigquery'
                 credfile='/u/authfiles/BigQuery/xxx-yyy-1a2b3333c4444.json'
                 schema='myschema'
                 project='project1'
                 mode="PERFORMANCE"
                 threadedLoad=true);
```

Details

Data Connector Options for Google BigQuery

Table 7.8 *Data Connector Options for Google BigQuery*

Data Connector Options	Default Value
allowLargeResults=	OFF
authenticationDomain= on page 398	none
bucket=	none
bulkLoad=	TRUE
bulkUnload=	FALSE
client_id=	none
client_secret=	none
credFile=	none
defaultDir=	*temporary-file-directory* that is specified by the UTILLOC= system option
deleteDataFiles=	TRUE
delimeter=	bell character (ASCII 0x07)
ignoreFedSQLObjects=	FALSE
largeResultsDataSet=	_sasbq_temp_tables
largeResultsExpirationTime=	86400000 (24 hours)
largeResultsProject=	none
mode=	STANDARD
numReadBuffers=	4
numReadNodes=	1
numWriteNodes=	1

Data Connector Options	Default Value
project=	none
proxy=	none
readBuff=	Calculated automatically based on row size
readMode=	STANDARD
refresh_token=	none
scanStringColumns=	FALSE
schema=	none
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "bigquery" for Google BigQuery data.
table_qualifier=	none
threadedLoad=	FALSE
use_information_schema=	TRUE

Options That Improve Performance

There are several options that impact the performance for data transfers between Google BigQuery and CAS. Here are the primary performance-related options:

- bulkLoad=TRUE (default) enables bulk loading to Google BigQuery from CAS
- bulkUnload=TRUE enables bulk loading from an external table into CAS
- mode="PERFORMANCE" automatically sets the following options:
  - numReadNodes=0
  - numWriteNodes=0
  - project="" (empty string)
  - readMode=STORAGE
  - scanStringColumns=TRUE

  You can override any of these values by specifying an alternative value for that option. For example, to work only with data for ProjectA, you would specify mode="PERFORMANCE" and project="ProjectA" in your CASLIB statement.

- numReadNodes=0 specifies that you want to use all available nodes to Read external data into CAS
- numWriteNodes=0 specifies that you want to use all available nodes to Write data from CAS to an external table
- scanStringColumns=TRUE specifies that you want to perform an initial scan of the data to determine the longest length for column data. The data connector then transfers the data to variables of an appropriate size in CAS in a subsequent scan.
- threadedLoad=TRUE specifies that you want to use multiple threads to load your data into CAS. Because of the overhead of partitioning data, you see the best performance enhancements for very large tables. Smaller tables can be negatively impacted by enabling this option. For this reason, it is recommended that you run tests on data on your system to determine what size tables typically benefit by using this option.

Supported Google BigQuery Data Types

The following table shows the data types that can be loaded from Google BigQuery into CAS. This table also shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.9 *Supported Google BigQuery Data Types*

Google BigQuery Data Type	CAS Data Type
	Character
JSON	VARCHAR
STRING	VARCHAR The default length is based on MAX_CHAR_LEN.
	Numeric
BIGNUMERIC	VARCHAR when precision is > 38 digits DOUBLE when precision is <= 38 digits
BOOLEAN	DOUBLE
FLOAT64	DOUBLE
INT64	INT64
NUMERIC	DOUBLE
	Date and Time

Google BigQuery Data Type	CAS Data Type
DATE	DOUBLE (formatted as DATE*w.*)
DATETIME	DOUBLE (formatted as DATETIME*w.d*)
TIME	DOUBLE (formatted as TIME*w.*)
TIMESTAMP	DOUBLE (formatted as DATETIME*w.d*)
	Other Data Types
ARRAY	VARCHAR The default length is based on MAX_CHAR_LEN.
GEOGRAPHY	VARCHAR The default length is based on MAX_CHAR_LEN.
RECORD[1]	Varies, based on the data type for the data in the RECORD object.

1 Also referred to as a STRUCT data type.

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Working with the JSON Data Type

You can load JSON data into CAS as VARCHAR data. When you work with JSON data in Google BigQuery, remember these restrictions:

- You cannot write a blank string or invalid JSON to a JSON column in Google BigQuery.
- You cannot extract values from JSON data. You need to pass a query directly to Google BigQuery to work with specific JSON values.

- Remember that WHERE clause processing is not passed to a data source. Instead, all table data is brought into CAS to process a WHERE clause. This can have a negative effect on performance for a query. Therefore, it is not recommended for you to subset JSON data with a WHERE clause.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Add a Google BigQuery Caslib and Load Data into SAS Cloud Analytic Services

Use the CASLIB statement to establish a connection between your Google BigQuery source data and a caslib, BQcaslib. All of the options supplied in this example are required in the CASLIB statement, except the SCHEMA= option.

In this example, the Google BigQuery data is stored at the location designated in the credentials file.

```
caslib BQcaslib desc='Google BigQuery Caslib'
     dataSource=(srctype='bigquery',
                 credfile='/u/myuser/authfiles/BigQuery/xxx-
yyy-1a2b3333c4444.json'
                 project="projectA"
                 schema="mySchema");                    1

proc casutil;
   list files incaslib="BQcaslib";                      2
   load casdata="myBQdata" outcaslib="BQcaslib" incaslib="casuser"
        casout="class_from_BQcaslib";                   3
   list files incaslib="casuser";                       4
```

```
   contents casdata="%upcase(class_from_BQcaslib)" incaslib="casuser";
                                                                    5
quit;
```

1 Add a caslib, BQcaslib, that connects your Google BigQuery database with CAS.

2 List the tables in BQcaslib before loading your data.

3 Load the table myBQdata from Casuser into BQcaslib. Call the new CAS table class_from_BQcaslib.

Note: You must specify Google BigQuery table names using the capitalization that is used in the database.

4 List the tables in Casuser to see the newly created CAS table, class_from_BQcaslib, that you loaded.

5 List information about the newly loaded table, including column names, data types, and so on.

Example 2: Load RECORD Data into CAS

This example shows how to load a Google BigQuery table called Sample that contains records. Although VARLIST= is not a data connector option, it is helpful to show how to refer to record data.

```
caslib gbqlib desc='Google BigQuery Caslib'
     dataSource=(srctype='bigquery'
                 credfile='/u/myuser/authfiles/BigQuery/xxx-
yyy-1a2b3333c4444.json'
                 project='project1');       /* 1 */

proc casutil;
   contents casdata="sample" incaslib="gbqlib";              /* 2 */
   load casdata="mySample" incaslib="gbqlib" outcaslib="casuser"
      casout="sample"
      varlist={'Sample.StringType'};        /* 3 */
quit;
```

1 Add a caslib, Gbqlib, that connects your Google BigQuery database with CAS.

2 List the contents of the Sample data set.

3 Load the StringType record from the Sample table in Gbqlib into caslib Casuser. Call the new CAS table MySample.

Example 3: Improve Load Performance for Large Tables

When you have large amounts of data, you can reduce the time it takes to load the data into CAS when you specify mode="PERFORMANCE" and threadedLoad=TRUE. When you set mode="PERFORMANCE", this sets several performance-related options to values. If you specify only threadedLoad=TRUE (and not mode="PERFORMANCE"), ensure that you have also set readMode="STORAGE". For more information, see mode= and threadedLoad= data connector options.

Note: Setting threadedLoad=TRUE with smaller tables might negatively impact performance when loading those tables. It is recommended that you test this option to have a sense of how large a table must be to result in better load performance with your system. For more information, see "Options That Improve Performance" on page 201.

This example shows a caslib that includes options that are designed to improve performance when you load large amounts of data.

```
caslib BQcaslib desc='Google BigQuery Caslib'
     dataSource=(srctype='bigquery',
                 credfile='/u/myuser/authfiles/BigQuery/xxx-
yyy-1a2b3333c4444.json'
                 project="projectA"
                 schema="mySchema"
                 mode="PERFORMANCE"
                 threadedLoad=true
                 );                                          1

proc casutil;
   list files incaslib="BQcaslib";                           2
   load casdata="myBQdata" outcaslib="BQcaslib" incaslib="casuser"
        casout="class_from_BQcaslib";                        3
   list files incaslib="casuser";                            4
   contents casdata="%upcase(class_from_BQcaslib)" incaslib="casuser";
                                                             5
quit;
```

1 Add a caslib, BQcaslib, that connects your Google BigQuery database with CAS. To improve load performance, set mode="PERFORMANCE" and threadedLoad=TRUE.

   Note: It is recommended that you test to ensure that threadedLoad=TRUE improves load performance for your data.

2 List the tables in BQcaslib before loading your data.

3 Load the table myBQdata from Casuser into BQcaslib. Call the new CAS table class_from_BQcaslib.

   Note: You must specify Google BigQuery table names using the capitalization that is used in the database.

4 List the tables in Casuser to see the newly created CAS table, class_from_BQcaslib, that you loaded.

5 List information about the newly loaded table, including column names, data types, and so on.

Greenplum Data Connector

Enables you to transfer data between Greenplum and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

See: Greenplum LIBNAME statement

Examples: Establish a connection between your Greenplum database and SAS Cloud Analytic Services.

```
caslib greenplumcaslib desc='Greenplum Caslib'
     dataSource=(srctype='greenplum'
                 server='GPserver'
                 username='user1'
                 password='myPwd'
                 database="GPdatabase-name"
                 schema='myschema');
```

Override the user and password values.

```
proc casutil;
   load casdata="gpexamp" casout="myGPdata" incaslib="greenplumcaslib"
      outcaslib="casuser"
      dataSourceOptions=(
         username='user5'
         password='myPwd');
quit;
```

Details

Data Connector Options for Greenplum

Data connector options are used in the context of different statements that connect your data in Greenplum with CAS. For each option described below, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.10 *Data Connector Options for Greenplum*

Data Connector Options	Default Value
authenticationDomain=	none
branded_driver=	none
catalog=	Active caslib

Data Connector Options	Default Value
charMultiplier=	1.0
conOpts=	none
database=	none
dbCreateTableOpts=	none
dbType=	none
greenplum_dsn=	none
insertBuff=	Calculated automatically based on row size
numReadBuffers=	4
numReadNodes=	0 (use all nodes)
numWriteNodes=	0 (use all nodes)
password=	none
port=	5432
readBuff=	Calculated automatically based on row size
schema=	none
server=	none
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "greenplum" for Greenplum data.
useNarrowCharacterTypes=	FALSE
userName=	none

Case Sensitivity with Greenplum

All quoted values are passed to the Greenplum data source exactly as you type them. This means that you must specify all values, such as table names or ID values, using the same capitalization that is used in the Greenplum database.

SAS/ACCESS passes all unquoted table names or column names to Greenplum in lower case.

Supported Greenplum Data Types

The following table shows the data types that can be loaded from Greenplum into CAS. This table also shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.11 *Supported Greenplum Data Types*

Greenplum Data Type	CAS Data Type
Character	
CHARACTER[(*p*)][1]	CHAR
CHARACTER VARYING[(*p*)][1]	VARCHAR
TEXT	VARCHAR
Numeric	
BIGINT	INT64
BIGSERIAL	INT64
DECIMAL[(*p,s*)][1]	DOUBLE
DOUBLE PRECISION	DOUBLE
INTEGER	INT32
REAL	DOUBLE
SERIAL	INT32
SMALLINT	INT32
Date and Time	
DATE	DOUBLE (formatted as DATE*w.*)
INTERVAL[(*p*)][1]	VARCHAR
TIME[(*p*)] [1]	DOUBLE (formatted as TIME*w.d*)

Greenplum Data Type	CAS Data Type
without time zone	
TIME[(*p*)][1] with time zone	VARCHAR
TIMESTAMP[(*p*)][1] without time zone	DOUBLE (formatted as DATETIME*w.d*)
TIMESTAMP[(*p*)][1] with time zone	VARCHAR
Network Address	
CIDR	VARCHAR
INET	VARCHAR
MACADDR	VARCHAR
Other	
JSON, JSONB	VARCHAR
MONEY	VARCHAR
UUID	CHAR

1 The *p* and *s* in Greenplum data types is equivalent to *w* and *d*, respectively, in SAS formats.

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Add a Greenplum Database as a Data Source for SAS Cloud Analytic Services

Use the CASLIB statement to establish a connection between your Greenplum source data and a caslib, Greenplumcaslib. All of the options supplied in this example are required in the CASLIB statement, except the SCHEMA= option.

In this example, the Greenplum data is stored at the location designated by the SERVER= and DATABASE= options.

```
caslib gpcaslib desc='Greenplum Caslib'
     dataSource=(srctype='greenplum',
                 server='GPserver'
                 username='user1',
                 password='myPwd',
                 database='GPdatabase',
                 schema='myschema');
```

Example 2: Load Greenplum Data into SAS Cloud Analytic Services Using PROC CASUTIL

```
proc casutil;
   list files incaslib="gpcaslib";          1
   load casdata="myGPdata" incaslib="gpcaslib" outcaslib="casuser"
        casout="class_from_Greenplumcaslib";       2
   list files incaslib="casuser";                  3
   contents casdata="%upcase(class_from_Greenplumcaslib)"
incaslib="casuser";
                                                   4
quit;
```

1 List the tables in Gpcaslib before loading your data.

2 Load the table myGPdata from Gpcaslib into caslib Casuser. Call the new table class_from_Greenplumcaslib.

Note: You must specify Greenplum table names using the capitalization that is used in the database.

3 List the tables in Casuser to see the newly created table, class_from_Greenplumcaslib, that you loaded.

4 List information about the newly loaded table, including column names, data types, and so on.

Hadoop Data Connector

Enables you to transfer data between Hive and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

Restrictions: The Hadoop data connector for the SAS Viya platform is supported only for Hive.

SAS Data Connect Accelerator for Hadoop does not support the ARRAY, MAP, STRUCT, UNION, BINARY, or BOOLEAN data types.

Requirement: To transfer data in parallel or use in-database scoring requires that you have SAS In-Database Technologies for Hadoop and SAS Embedded Process installed in the data source. SAS Embedded Process is delivered with SAS In-Database Technologies for Hadoop, which is packaged with some SAS Viya platform offerings and can be added to other offerings.

Supports: For supported Hadoop distributions, see "Requirements for SAS/ACCESS Interface to Hadoop" in *System Requirements for the SAS Viya Platform*.

Note: Starting in 2023.10, SAS redistributes the Simba Apache Hive JDBC Data Connector from insightsoftware for connectivity to Hadoop. This driver replaces the CData JDBC Driver for Apache Hive. If you used CData properties in your connection, you must make changes. For more information, see "Troubleshooting Hadoop Connection Problems" in *SAS/ACCESS for Relational Databases: Reference*.

See: LIBNAME Statement for Hadoop

Examples: Add a basic caslib for Hive that can access data in HDFS.

```
caslib hvcaslib desc='hadoop caslib'
     dataSource=(srctype='hadoop',
                 server='hiveServer',
                 username='myuser1',
                 password='myPwd',
                 hadoopJarPath='<Hadoop_jar_path_directory>',
                 hadoopConfigDir='<Hadoop_configuration_directory>',
                 schema='<Hive_schema_name>');
```

Load Hadoop source data using PROC CASUTIL.

```
proc casutil;
      load incaslib='hvcaslib' casdata='cars'
      casout='cars_CAS'
      where='cylinders=8'
      options=(dataTransferMode='serial');
run;
```

```
quit;
```

Details

Data Connector Options for Hadoop

Data connector options are used in the context of different statements that connect your data in Hadoop with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements and action calls, see "Where to Specify Data Connector Options" on page 175.

Table 7.12 *Data Connector Options for Hadoop*

Data Connector Options	Default Value
allowSelfSignedCerts=	value configured in hive-site.xml file
authenticationDomain= [1]	none
authType=	LDAP
bufferSize=	1048576
bulkLoad=	TRUE
catalog=	Active caslib
charMultiplier=	1.0
classPath=	none
dataTransferMode=	"SERIAL"
dbCreateTableOpts=	none
dbMaxText=	1024
dispatchMethod=	STANDARD
driverClass=	"com.simba.hive.jdbc.HS2driver"
hadoopConfigDir=	none
hadoopJarPath=	none
hdfsTempDir=	"/TMP"

Data Connector Options	Default Value
httpPath=	none
insertBuff=	Calculated automatically based on row size
invalidDate=	WARN
jobManagementURL=	none
krbAuthType=	none
krbHostFQDN=	none
krbRealm=	none
krbServiceName=	none
login_timeout=	30
mapReduceJobName= [1]	value configured in hive-site.xml file
mapReduceQueueName= [1]	value configured in hive-site.xml file
maxLoadThreads=	24
maxSaveThreads=	24
numReadBuffers=	4
numReadNodes=	1
numWriteNodes=	1
password=	none
platform=	spark
port=	10000
post_table_opts=	none
pre_table_opts=	none
properties=	none
readBuff=	Calculated automatically based on row size

Data Connector Options	Default Value
schema=	'default'
scratchDb=	none
server=	none
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "hadoop" for Hadoop data.
sslKeyStore=	none
sslKeyStorePwd=	none
sslTrustStore= [1]	value configured in hive-site.xml file
statusInterval=	0 (no message)
tezQueueName= [1]	value configured in hive-site.xml file
trustStorePassword= [1]	value configured in hive-site.xml file
uri=	none
userName=	none
useSSL=	FALSE
webHdfsUrl=	none
zooKeeperNamespace= [1]	value configured in hive-site.xml file
zooKeeperPort= [1]	none
zooKeeperQuorum= [1]	value configured in hive-site.xml file

1 These options enable you to specify Hadoop configuration options on the CASLIB statement instead of in a configuration file. When a configuration file also exists, the CASLIB settings override the settings in the configuration file.

Hadoop Naming

The data connector and data connect accelerator can load Hive tables with names up to 128 characters or with column names that are up to 128 characters.

Authentication to a Hive (Hadoop) Data Source

Depending on the user setup, credentials might be required to access the data in a Hive data source. Your system administrator defines an authentication domain that is associated with a pair of user ID and password values. The authentication domain and associated credentials are then available to you and other users who need to access data.

Typically, you supply credentials when you add a caslib. To do this, use the CASLIB statement and specify a value for either authenticationDomain= or both userName= and password=. These credentials are then used for any statement that accesses the data using that caslib.

You can specify different credentials when you issue a statement that accesses the data. If you originally specified an authentication domain when you added a caslib, then you must specify values for both userName= and password= to override the original credentials. You can also override authentication domain credentials with a different authenticationDomain= value.

When a Kerberos connection is used, do not specify the authenticationDomain=, userName=, or password= options.

Parallel Data Transfer for Cloudera

Overview of Parallel Data Transfer

Parallel data transfer includes parallel loading and parallel saving of data. This feature relies on the SAS Embedded Process, which is shipped with SAS In-Database Technologies for Hadoop. The SAS Embedded Process must be installed and configured. For more information, see *SAS In-Database Products: Deployment and Administration Guide*.

Requirements for Parallel Data Transfer

Set the following data connector options in the CASLIB statement to enable parallel loading:

- dataTransferMode="parallel", to request parallel data transfer
- jobManagementUrl=, to specify the URL to an Apache Livy end point
- hadoopConfigDir=, to specify the client-side location of the SAS Embedded Process sparkep-defaults.conf and spark_defaults.conf configuration files

For an example, see "Example 5: Access Hive Using Parallel Data Transfer".

Load a Connection to a Hadoop Data Source

When you load a connection to a data source, you specify the type of data source to which SAS Cloud Analytic Services is connecting. However, a connection is not made until you load data from the data source. To load the connection to the data source and to add a caslib that accesses that data source, use the CASLIB statement.

Supported Hive Data Types

The Hadoop data connector supports these data types for loading Hive and data into CAS. This table shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.13 *Supported Hive Types*

Hive Data Type	CAS Data Type
	Character
CHAR	CHAR
VARCHAR[1]	VARCHAR
STRING	VARCHAR
	Numeric
ARRAY	ERROR (not supported)
BIGINT	INT64
BINARY	ERROR (supported as VARCHAR for serial data transfer, not supported for parallel data transfer)
BOOLEAN	ERROR (supported as INT32 for serial data transfer, not supported for parallel data transfer)
DECIMAL	DOUBLE
DOUBLE	DOUBLE
FLOAT	DOUBLE
	INT32
INTEGER	INT32
INTERVAL	DOUBLE
MAP	ERROR (not supported)
	DOUBLE

Hive Data Type	CAS Data Type
SMALLINT	INT32
STRUCT	ERROR (not supported)
TINYINT	INT32
UNIONTYPE	ERROR (not supported)
	Date and Time
DATE	DOUBLE
TIME	DOUBLE
TIMESTAMP (DATETIMES)[2]	DOUBLE

1 Any VARCHAR data types that are greater than 65535 are automatically promoted to a string type upon CREATE.

2 SAS Data Connect Accelerator for Hadoop supports this data type up to the Year 4000 with millisecond precision.

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is

supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Overview of Examples

The following examples provide a basic introduction to common SAS/ACCESS to Hadoop configurations. These examples assume the administrator has previously performed the tasks in "Configuring SAS/ACCESS Interface to Hadoop" in *SAS/ACCESS for Relational Databases: Reference* to configure access to HDFS. After HDFS access is configured, you enable HDFS access by specifying the hadoopJarPath= and hadoopConfigPath= options in the CASLIB statement. Bulk loading is available by default when HDFS access is configured; therefore, there is no need to specify bulkLoad=true on the LIBNAME statement.

Use of the username= and password= options specify that LDAP authentication is used. Use of the authenticationDomain= option is preferred to avoid exposing sensitive credentials.

Example 1: Connect to Hive with Dynamic Service Discovery (ZooKeeper)

A typical HADOOP caslib specifies ZooKeeper information that provides for Dynamic Service Discovery. Apache ZooKeeper is a distributed coordination system that allows Hive to run in HiveServer2 high availability mode. In high availability mode:

- When running multiple HiveServer2 instances, each server registers itself in ZooKeeper.
- Clients (such as the Apache Hive driver that is redistributed with the SAS Viya platform) query ZooKeeper to get the list of available HiveServer2 instances. This ensures automatic failover if one HiveServer2 instance goes down.

To enable Dynamic Service Discovery with SAS/ACCESS to Hadoop, specify the ZooKeeper options zooKeeperQuorum=, zooKeeperNamespace=, and zooKeeperPort=.

```
caslib zoocaslib desc='ZooKeeper caslib'
   dataSource=(srctype='hadoop',username='myuser1',password='myPwd',
               dataTransferMode='serial',
               hadoopJarPath='Hadoop_jar_path_directory',
               hadoopConfigDir='Hadoop_configuration_directory',

zooKeeperQuorum="node1.company.com,node2.company.com,node3.company.com"
               zooKeeperNamespace='hiveserver2',
               zooKeeperPort=2181,
               schema='Hive_schema_name');
```

Example 2: Connect to Hive without Dynamic Service Discovery

To connect without using ZooKeeper, specify the server= option. The server= option specifies the HiveServer endpoint. When server= is specified, HiveServer2 high availability will not be used by SAS/ACCESS to Hadoop. This option is appropriate when your Hadoop endpoint is not configured with ZooKeeper.

Access to HDFS is serial.

```
caslib hvcaslib desc='hadoop caslib'
     dataSource=(srctype='hadoop',
                 server='hiveServer',
                 username='myuser1',
                 password='myPwd',
                 hadoopJarPath='Hadoop_jar_path_directory',
                 hadoopConfigDir='Hadoop_configuration_directory',
                 schema='Hive_schema_name');
```

Example 3: Connecting to Hive with Kerberos Authentication

This connection uses the default JDBC driver that is redistributed with the SAS Viya platform. Four of the options are required only for connecting to a Hive server that is configured with Kerberos authentication. For more information about the options, see krbAuthType=, krbHostFQDN=, krbRealm=, and krbServiceName=. These options pass Kerberos connection properties to the Simba driver. For more information about the Simba properties, see "Using Kerberos" and "Connector Configuration Options" on the insightsoftware website.

```
caslib hadoopkrb
     dataSource=(srctype='hadoop',
                 server='hive_server_name',
                 schema='Hive_schema_name',
                 use_ssl=true,
                 krbAuthType=2,
                 krbHostFQDN='myhost.company.com',
                 krbRealm='example.com',
                 krbServiceName='hive',
                 bulkload=false);
```

Example 4: Specify Hive as a Data Source for a Caslib

This example makes a Hive connection without HDFS. Using this caslib, you can read Hive data in the specified data source, but you cannot write to it. Because the Hadoop data connector has bulk loading enabling by default, you must specify bulkLoad=false in the CASLIB statement.

The CASLIB statement initializes the data source and adds the caslib for Hadoop. No connection is made to the data source until a statement that accesses the data is called.

```
caslib hvcaslib desc='hadoop caslib'
     dataSource=(srctype='hadoop',
                 server='hiveServer',
                 username='myuser1',
                 password='myPwd',
                 schema='Hive_schema_name',
                 bulkload=false);
```

Example 5: Access Hive Using Parallel Data Transfer

To enable parallel data transfer, you must specify dataTransferMode="parallel", hadoopConfigDir=, and jobManagementURL= (alias restURL=) data connector options.

The CASLIB statement initializes the data source and adds the caslib for Hadoop. No connection is made to the data source until a statement that accesses the data is called.

```
caslib hvcaslib desc='hadoop caslib'
   dataSource=(srctype='hadoop',
               server='hiveServer',
               schema='Hive_schema_name'
               username='myuser1',
               password='myPwd',
               webHdfsUrl='URL',
               dataTransferMode="parallel",
               hadoopConfigDir='path_to_EP_config_files',
               jobManagementURL='Apache_Livy_end_point');
```

Example 6: Load a Hive Table into CAS Using PROC CASUTIL

```
proc casutil;
   list files incaslib="hvcaslib";                    1
   load casdata="myHDdata" incaslib="hvcaslib" outcaslib="casuser"
        casout="HDdata_from_hvcaslib";                2
   list tables incaslib="casuser";                    3
   contents casdata="HDdata_from_hvcaslib" incaslib="casuser";
                                                      4
quit;
```

1 List the tables in hvcaslib before loading your data.

2 Load the table myHDdata from Hive into the caslib casuser. Call the new table HDdata_from_hvcaslib.

3 List the tables in caslib casuser to see the newly created HDdata_from_hvcaslib table that you loaded.

4 List information about the newly loaded table, including column names, data types, and so on.

Impala Data Connector

Enables you to transfer data between Impala and CAS.

Valid in:	CASLIB statement, PROC CASUTIL statements (see options for details)
Note:	All users with SAS/ACCESS Interface to Impala can use SAS Data Connector to Impala.
See:	Impala LIBNAME statement
Examples:	Load Impala source data and add a caslib to it.

```
caslib imcaslib sessref=mysess
  dataSource=(srctype='impala',
              server='impala01.example.com',
              username='myuser1'
              password='myPwd'
              database='mydb';
```

Load Impala source data using PROC CASUTIL.

```
proc casutil
      incaslib='imcaslib'
      sessref=mysess;
   load incaslib='imcaslib' casdata='cars'
      casout='cars_CAS'
      where='cylinders=8';
run;
quit;
```

Details

Data Connector Options for Impala

Data connector options are used in the context of different statements that connect your data in Impala with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.14 *Data Connector Options for Impala*

Data Connector Options	Default Value
authenticationDomain=	none
bulkLoad=	TRUE
catalog=	Active caslib
charMultiplier=	1.0
client_encoding=	UTF-8
conOpts=	none
dbCreateTableOpts=	none
dbType=	none
dm_unicode=	UCS-2/UTF-16

Data Connector Options	Default Value
hadoopConfigDir=	none
hadoopJarPath=	none
hdfsTempDir=	"/TMP"
impala_dsn=	none
insertBuff=	Calculated automatically based on row size
krbHostFQDN=	none
krbRealm=	none
krbServiceName=	none
numReadBuffers=	4
numReadNodes=	1
numWriteNodes=	1
password=	none
port=	21050
readBuff=	Calculated automatically based on row size
schema=	none
scratchDb=	none
server=	none
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "impala" for Impala data.
useNarrowCharacterTypes=	FALSE
userName=	none

Impala Naming

The data connector can load Impala tables with names up to 128 characters or with column names that are up to 128 characters.

Authentication to an Impala Database

Credentials are required to access the data in an Impala database. Your system administrator defines an authentication domain that is associated with a pair of user ID and password values. The authentication domain and associated credentials are then available to you and other users who need to access data.

Typically, you supply credentials when you add a caslib. To do this, use the CASLIB statement and specify a value for either AUTHENTICATIONDOMAIN= or both USERNAME= and PASSWORD=. These credentials are then used for any statement that accesses the data using that caslib.

You can specify different credentials when you issue a statement that accesses the data. If you originally specified an authentication domain when you defined a caslib, then you must specify values for both USERNAME= and PASSWORD= to override the original credentials. You can also override authentication domain credentials with a different AUTHENTICATIONDOMAIN= value.

Although it is not typical, it is possible to supply USERNAME= and PASSWORD= in separate statements. For example, you might supply a USERNAME= value in the CASLIB statement and then supply a PASSWORD= value when you specify a LOAD statement that accesses the data. In this situation, you must supply the PASSWORD= value for each statement that accesses the data.

Load a Connection to an Impala Data Source

When you load a connection to a data source, you specify the type of data source to which SAS Cloud Analytic Services is connecting. However, a connection is not made until you load data from the data source. To load the connection to the data source and to define a caslib that accesses that data source, use the CASLIB statement.

Supported Impala Data Types

The Impala data connector supports these data types for Impala. This table shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.15 *Supported Impala Data Types*

Impala Data Type	CAS Data Type
	Character
CHAR	CHAR
VARCHAR	VARCHAR

Impala Data Type	CAS Data Type
STRING	VARCHAR
	Numeric
BIGINT	INT64
BOOLEAN	DOUBLE
DECIMAL(*p*, *s*)	DOUBLE
DOUBLE	DOUBLE
FLOAT	DOUBLE
INTEGER	INT32
SMALLINT	INT32
TINYINT	INT32
	Date and Time
DATE[1]	DOUBLE (apply the desired date format, such as date7. or date9.)
TIMESTAMP(*p*) (without time zone)	DOUBLE (formatted as DATETIME*w.d*)

1 The DATE type is supported with Impala 3.3 and later.

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Specify an Impala Database as a Data Source for a Caslib in SAS Cloud Analytic Services

```
caslib imcaslib desc='Impala Caslib'
   dataSource=(srctype='Impala',
               server='impalaServer',
               username='myuser1',
               password='myPwd',)
               schema='mydb';
```

Example 2: Load an Impala Table into SAS Cloud Analytic Services Using PROC CASUTIL

```
proc casutil;
   list files incaslib="Impalacaslib";                    1
   load casdata="myIMdata" incaslib="Impalacaslib" outcaslib="casuser"
        casout="IMdata_from_Impalacaslib";                2
   list tables incaslib="casuser";                        3
   contents casdata="IMdata_from_Impalacaslib" incaslib="casuser";
                                                          4
quit;
```

1 List the tables in Impalacaslib before loading your data.

2 Load the table myIMdata from Impala into caslib Casuser. Call the new table IMdata_from_Impalacaslib.

3 List the tables in caslib to see the newly created IMdata_from_Impalacaslib table that you loaded.

4 List information about the newly loaded table, including column names, data types, and so on.

Example 3: Use the Impala Data Connector to Bulk Load Data from CAS to Impala

```
caslib implib sessref=mysessref
    datasource=(srctype="impala",
                server="server.company.com",
                schema="mydb",
                hadoopJarPath="/u/myadmin/hadoopjars/cdh/prod",
                hadoopConfigDir="/u/myadmin/hadoopcfg/cdh/prod",
                bulkload=yes
               );

proc cas;
   session mysessref;
   loadtable caslib="implib" path="IMPTAB"
      options={numReadNodes=1} casout="CASTAB";
run;
   save caslib="implib" name="IMPTAB_BULK"
      table="CASTAB" options={numWriteNodes=1};
run;
quit;
```

Example 4: Connect to Impala with Kerberos Authentication

Three of these options are required for connecting to a Hive server that is configured with Kerberos authentication. For more information about these options, see krbHostFQDN=, krbRealm=, and krbServiceName=. These options pass connection properties to an environment that uses Kerberos.

```
caslib imcaslib desc='Impala Caslib'
   dataSource=(srctype='Impala',
               server='impalaServer',
               schema='mydb',
               krbHostFQDN=myhost.company.com,
               krbRealm=example.com,
               krbServiceName=impala
               use_ssl=yes);
```

Informix Data Connector

Enables you to transfer data between an Informix data source and CAS.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, FILE, LOAD, SAVE statements

Note: All users with SAS/ACCESS Interface to Informix can use SAS Data Connector to Informix.

See: Informix LIBNAME statement connection

Example: Establish a connection to Informix data.

```
caslib Infcaslib desc="InformixToCaslib"
     datasource=(srctype="informix"
```

```
host='informix-host'
port=14463
server="DBMS-server-on-host"
database="database-name"
username="user1"
password="password1");
```

Details

Data Connector Options for Informix

Data connector options are used in the context of different statements that connect your data in an Informix data source with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.16 *Data Connector Options for Informix Data Sources*

Data Connector Option	Default Value
authenticationDomain=	none
blDbaccessPath=	**<INFORMIX_SDK_PATH>/bin/dbaccess**
blDefaultDir=	current location for storing temporary data files
blDelDatFile=	TRUE
blDelimiter=	| (pipe)
blEnableLog=	FALSE
blLogDir=	blDefaultDir= value, if specified, or the location where temporary files are stored
blNfxSdkPath=	none
bulkLoad=	FALSE
bulkUnload=	FALSE
catalog=	none
charMultiplier=	1.0
client_encoding=	none

Data Connector Option	Default Value
conOpts=	none
database=	none
dbCreateTableOpts=	none
host=	none
insertBuff=	Calculated automatically based on row size
numReadBuffers=	4
numReadNodes=	0 (use all nodes)
numWriteNodes=	0 (use all nodes)
password=	none
port=	none Use port=14463 for Informix.
readBuff=	Calculated automatically based on row size
schema=	none
server=	none
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "Informix" for Informix data sources.
useNarrowCharacterTypes=	FALSE
userName=	none

Bulk Loading and Bulk Unloading Data between CAS and Informix

Support has been added for bulk loading and bulk unloading data between Informix and CAS. Bulk loading enables you to move large amounts of data from CAS to Informix as efficiently as possible. Similarly, bulk unloading enables you to move

large amounts of data from Informix to CAS. To enable bulk data transfer, set bulkLoad= on page 440 and bulkUnload= to TRUE.

Use the following data connector options when bulk loading or bulk unloading data:

- blDbaccessPath=
- blDefaultDir=
- blDelDatFile=
- blDelimiter=
- blEnableLog=
- blLogDir=
- blNfxSdkPath=

Supported Informix Data Types

The following table lists the supported data types that the Informix data connector can load from Informix into SAS Cloud Analytic Services. This table also shows the resulting data type in CAS.

Note: Be aware that when performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that are read are rounded to meet this condition. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data.

Table 7.17 *Data Type Conversions When Loading Data from Informix into CAS*

Informix Data Type	CAS Data Type
BYTE	VARCHAR
CHAR	VARCHAR
DATE	DOUBLE (formatted as DATE)
DATETIME	DOUBLE (formatted as DATETIME)
DECIMAL	DOUBLE
DOUBLE	DOUBLE
FLOAT	DOUBLE
INT	INTEGER (INT32)
INT8	INT64

Informix Data Type	CAS Data Type
INTEGER	INTEGER (INT32)
LVARCHAR	VARCHAR
NCHAR	VARCHAR
NUMERIC(*p,s*)	DOUBLE
NVARCHAR	VARCHAR
REAL	DOUBLE
SMALLINT	INTEGER (INT32)
VARCHAR	VARCHAR

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types. Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved.

A SAS DOUBLE value maintains approximately 15 digits of precision. When you read values that contain more than 15 decimal digits of precision into a DATA step, the data is converted to a DOUBLE value. When you read these values into a procedure that is not included with the SAS Viya platform, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Establish a Connection between an Informix Database and a Caslib

Use the CASLIB statement to add an Informix database as a data source for SAS Cloud Analytic Services.

In this example, GLOBAL specifies that the data source is potentially available to all sessions. You might not have access to the GLOBAL option if you are not an administrator. For more information, see "CASLIB Statement" on page 745.

```
caslib Informixcaslib desc="InformixtoCaslib"
     datasource=(srctype="informix"
                 host='informix-host'
                 port=14463
                 server="DBMS-server-on-host"
                 database="database-name"
                 username="user1"
                 password="password1") global;
```

Example 2: PROC CASUTIL: Load Data from an Informix Database into SAS Cloud Analytic Services

```
proc casutil;
  list files incaslib="informixcaslib";       /* 1 */
  load casdata="myDBdata" incaslib="informixcaslib" outcaslib="casuser"
       casout="class_from_informixcaslib";    /* 2 */
  list tables incaslib="casuser";             /* 3 */
  contents casdata="class_from_informixcaslib" incaslib="casuser";
                                              /* 4 */
quit;
```

1 List the files in informixcaslib before loading your data.

2 Load the table myDBdata from an external database into memory in caslib Casuser. Call the new table class_from_informixcaslib.

3 List the tables in casuser to see the newly created table, class_from_informixcaslib, that you loaded.

4 List information about the newly loaded table, including column names, data types, and so on.

JDBC Data Connector

Enables you to transfer data between a JDBC data source and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

Note: All users with SAS/ACCESS Interface to JDBC can use SAS Data Connector to JDBC.

See: JDBC LIBNAME statement

Example: Add a caslib that connects to a JDBC data source.

```
caslib jdcaslib sessref=mysess
  dataSource=(srctype='jdbc',
              url="jdbc:myDriver://<connection-options>"
              username='myuser1'
              password='myPwd'
              class="JDBC-driver-class");
```

Details

Data Connector Options for JDBC

Data connector options are used in the context of different statements that connect your data in JDBC with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.18 *Data Connector Options for JDBC*

Data Connector Options	Default Value
authenticationDomain=	none
catalog=	Active caslib
charMultiplier=	1.0
class=	none
classPath=	none
dbCreateTableOpts=	none
dbMaxText=	32767
dbType=	none
insertBuff=	Calculated automatically based on row size
invalidDate=	WARN
login_timeout=	30
numReadBuffers=	4

numReadNodes=	1
numWriteNodes=	1
password=	none
query_timeout=	none
readBuff=	Calculated automatically based on row size
schema=	none
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "jdbc" for JDBC data sources.
url=	none
useNarrowCharacterTypes=	FALSE
userName=	none

JDBC Naming

The data connector can load JDBC tables with names up to 128 characters or with column names that are up to 128 characters.

Authentication to a JDBC Database

Credentials are required to access the data in a JDBC database. Your system administrator defines an authentication domain that is associated with a pair of user ID and password values. The authentication domain and associated credentials are then available to you and other users who need to access data.

Typically, you supply credentials when you add a caslib. To do this, use the CASLIB statement and specify a value for either authenticationDomain= or both userName= and password=. These credentials are then used for any statement that accesses the data using that caslib.

You can specify different credentials when you issue a statement that accesses the data. If you originally specified an authentication domain when you defined a caslib, then you must specify values for both userName= and password= to override the original credentials. You can also override authentication domain credentials with a different authenticationDomain= value.

Although it is not typical, it is possible to supply userName= and password= in separate statements. For example, you might supply a userName= value in the

CASLIB statement and then supply a password= value when you specify a LOAD statement that accesses the data. In this situation, you must supply the password= value for each statement that accesses the data.

Load a Connection to a JDBC Data Source

When you load a connection to a data source, you specify the type of data source to which SAS Cloud Analytic Services is connecting. However, a connection is not made until you load data from the data source. To load the connection to the data source and to define a caslib that accesses that data source, use the CASLIB statement.

Supported JDBC Data Types

This table shows the data types that can be loaded from JDBC into CAS. This table also shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.19 *Supported JDBC Data Types*

JDBC Data Type	CAS Data Type
	Character
CHAR	CHAR
VARCHAR	VARCHAR
STRING	VARCHAR
	Numeric
BIGINT	INT64
BOOLEAN	DOUBLE
DECIMAL(*p*, *s*)	DOUBLE
DOUBLE	DOUBLE
FLOAT	DOUBLE
INTEGER	INT32
SMALLINT	INT32
TINYINT	INT32
	Date and Time

JDBC Data Type	CAS Data Type
TIMESTAMP(*p*) (without time zone)	DOUBLE (formatted as DATETIME*w.d*)

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Example: Specify a JDBC Connection to PostgreSQL via JDBC and Load a PostgreSQL Table

```
caslib jdcaslib desc='JDBC Caslib'
   dataSource=(srctype='jdbc',                    1
               url="jdbc:postgresql://<PostgreSQL-server>:<PostgreSQL-
port>/
                        <PostgreSQL-database>",
               username='myuser1',
               password='myPwd'
               class="org.postgresql.Driver");
```

```
proc casutil;
   list files incaslib="jdcaslib";                    2
   load casdata="myDBdata" incaslib="jdcaslib" outcaslib="casuser"
        casout="DBdata_from_jdcaslib";                3
   list tables incaslib="casuser";                    4
   contents casdata="DBdata_from_jdcaslib" incaslib="casuser";
                                                      5
quit;
```

1 Create the caslib Jdcaslib that connects to a PosgreSQL database via JDBC.

2 List the tables in Jdcaslib before loading your data.

3 Load the table myDBdata from your JDBC data source into caslib Casuser. Call the new table DBdata_from_jdcaslib.

4 List the tables in caslib Casuser to see the newly created DBdata_from_jdcaslib table that you loaded.

5 List information about the newly loaded table, including column names, data types, and so on.

Microsoft SQL Server Data Connector

Enables you to transfer data between Microsoft SQL Server and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

See: Microsoft SQL Server engine

Examples: Establish a connection between your Microsoft SQL Server database and SAS Cloud Analytic Services.

```
caslib sqlservercaslib desc='Microsoft SQL Server Caslib'
     dataSource=(srctype='sqlserver'
                 username='user1'
                 password='myPwd'
                 sqlserver_dsn="configured-DSN-name"
                 catalog='*');
```

Override the user and password values.

```
proc casutil;
   load casdata="mycas.sqlsvrexamp" casout="mySQLsvrdata" incaslib="sqlservercaslib"
      outcaslib="casuser"
      dataSourceOptions=(
         username='user5'
         password='myPwd');
quit;
```

Details

Data Connector Options for Microsoft SQL Server

Data connector options are used in the context of different statements that connect your data in Microsoft SQL Server with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.20 *Data Connector Options for Data Sources That Are Compatible with Microsoft SQL Server*

Data Connector Options	Default Value
accountName=	none
applicationID=	none
authenticationDomain=	none
authenticationScope=	`https://database.windows.net/user_impersonation`
blAuthDomain=	none
blFormat=	DELIMITED
blOptions=	none
bulkLoad=	FALSE
catalog=	Active caslib
charMultiplier=	1.0
client_encoding=	Shell encoding
clientSecret=	none
compress=	FALSE
connectionTypeForCreate=	fedsql
conOpts=	none
dbCreateTableOpts=	none
dbType=	none

Data Connector Options	Default Value
defaultDir=	*SAS-utility-directory*
deleteDataFiles=	TRUE
delimiter=	bell character (ASCII 0x07)
dsnSuffix=	dfs.core.windows.net
escape=	FALSE
fileSystem=	none
folder=	none
identity=	none
insertBuff=	Calculated automatically based on row size
insertMultiValue=	FALSE (most Microsoft SQL Server data sources) TRUE (Microsoft Fabric Data Warehouse)
insertSQL=	TRUE
invalidDate=	WARN when loading multiple records ERROR when loading records individually
maxErrors=	0
numDataFiles=	4
numReadBuffers=	4
numReadNodes=	0 (use all nodes)
numWriteNodes=	0 (use all nodes)
password=	none
readBuff=	Calculated automatically based on row size
schema=	none
secret=	none
sliceColumn=	none

Data Connector Options	Default Value
sliceExpressions=	none
sqlServer_dsn=	none
srcType=	none Use "sqlserver" for Microsoft SQL Server data.
timeout=	60
useLog=	FALSE
useNarrowCharacterTypes=	FALSE
userName=	none

Authenticate to Microsoft Azure with Microsoft Entra ID

Single sign-on for the SAS Viya platform uses Microsoft Entra ID (formerly Azure Active Directory) to control authentication. This requires an administrator to configure Microsoft Entra ID as an OpenID Connect (OIDC) provider. For more information, see "Scenario: OIDC with Microsoft Entra ID" in *SAS Viya Platform: Authentication*.

To use single sign-on, establish a connection to Microsoft SQL Server that uses the default authentication scope for Microsoft Azure. To use a different authentication scope, you can specify the authScope= option. Here is a sample CASLIB statement that can connect to a Microsoft SQL Server instance on Microsoft Azure:

```
caslib db datasource=(srctype='sqlserver',
                      sqlserver_dsn="configured-DSN-name",
                      filesystem="SQLserver-file-system",
                      catalog="*");
```

When authenticating with single sign-on, do not specify the userName=, password=, or authenticationDomain= options. These options disable single sign-on. The conOpts= option also disables single sign-on unless you specify the authScope= option as well.

For bulk loading to Azure, additional authentication is required due to the SAS/ACCESS bulk-loading process. For instructions, see "Authentication to Azure for Bulk Loading" on page 242.

Working with Microsoft Fabric Data Warehouse and Microsoft Fabric Lakehouse

Overview of Authentication and Features of Microsoft Fabric

You can read and write data between CAS and the Synapse Data Warehouse endpoints of Microsoft Fabric Data Warehouse. Similarly, you can read data into CAS from Microsoft Fabric Lakehouse.

Accessing Microsoft Fabric Data Warehouse or Microsoft Fabric Lakehouse requires using Microsoft Entra ID for authentication. You can authenticate using single sign-on or other authentication methods, such as authentication methods 13 (Active Directory Password), 36 (Active Directory Service Principal), 37 (Active Directory Managed Identity). This is typically set up by an administrator. For more information, see "How To" in *SAS Viya Platform: Authentication* or see your Microsoft Fabric documentation.

It is recommended that you use bulk loading to write data to Microsoft Fabric Data Warehouse. Follow the steps in these topics to bulk load data:

- "Authentication to Azure for Bulk Loading"
- "Overview of Bulk Loading to ADLS2"
- "Inserting Multiple Values for Microsoft Fabric Data Warehouse" on page 244

For information about supported data types with Microsoft Fabric, see "Supported Data Types for Microsoft Fabric" on page 245.

If you are using the Compute server, then you can connect to Microsoft Fabric by using a LIBNAME statement for Microsoft SQL Server. For more information, see "LIBNAME Statement for Microsoft SQL Server" in *SAS/ACCESS for Relational Databases: Reference*.

Known Restrictions and Limitations with Microsoft Fabric

Here are some known restrictions and limitations that pertain to Microsoft Fabric Data Warehouse or Microsoft Fabric Lakehouse:

- Table and column names in Microsoft Fabric are always case sensitive. Ensure that table and column names in SAS programs use accurate cases.
- The order of rows that are returned are not returned in the order in which they were inserted. A SELECT query with PROC SQL does not return rows in a particular order.
- Microsoft Fabric does not support Cursor operations.

Bulk Loading to Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse

Overview of Bulk Loading to ADLS2

You can use the Microsoft SQL Server data connector to bulk load data to Azure Synapse Analytics (SQL DW) or Microsoft Fabric Data Warehouse. Tables are

loaded to Microsoft Azure Data Lake Storage Gen2 (ADLS2) storage. Then, the Azure COPY command is used to transfer the data from external storage to a table in Azure Synapse. Supported formats for files that you can bulk load are delimited files (CSV) and compressed delimited files (CSV files compressed with GZIP). You create delimited and compressed delimited files on a CAS node and then bulk load the data to ADLS2 storage via the bulk-loading options in your CASLIB or PROC CASUTIL SAVE statement.

Starting in 2026.05, uncompressed CSV files can be streamed directly into ADLS2 storage. Use the numDataFiles= option to control the number of streams that load the data. For more information, see "numDataFiles= Data Connector Option" on page 535.

Here are the options that you use when you bulk load data to ADLS2:

accountName=	dnsSuffix=
applicationId=	escape=
blAuthDomain=	fileSystem=
blFormat=	folder=
blOptions=	identity=
bulkLoad=	maxErrors=
clientSecret=	numDataFiles=
compress=	secret=
defaultDir=	timeout=
deleteDataFiles=	useLog=
delimiter=	

Note: Do not specify applicationID= if single sign-on with OIDC is configured. For details, see "Authentication to Azure for Bulk Loading" on page 242.

For more information, see "Data Connector Options for Microsoft SQL Server" on page 238. You can see an example of using the bulk-loading options in "Example 3: Bulk Load to ADLS2 Storage" on page 248.

Authentication to Azure for Bulk Loading

For bulk loading, additional authentication is required due to the process that is described in "Overview of Bulk Loading to ADLS2" on page 241.

You have several choices for authenticating to Azure. For any of these methods, the user ID that is used to authenticate must be assigned Contributor and Storage Blob Data Contributor roles in the ADLS account configuration.

- Client ID and client secret. SAS/ACCESS attempts to authenticate first with this method by checking for blAuthDomain= first, and then applicationID= and clientSecret values:
  - accountName=
  - applicationID=
  - blAuthDomain=

Note: If you set blAuthDomain=, do not also specify applicationID= and clientSecret=.

- clientSecret=

  Note: See the steps to acquire the client token value in the Details for this option.

- fileSystem=
- identity=
- secret=

Note: Support for authentication with client ID and client secret values was added in 2025.08.

- Single sign-on with Microsoft Entra ID. This authentication method uses the OIDC authentication protocol.
  - Your administrator configures single sign-on. For more information, see "Scenario: OIDC with Microsoft Entra ID" in *SAS Viya Platform: Authentication*.
  - The following options are all required:

    accountName=

    fileSystem=

    identity=

    secret=
- Device-code authentication. This method is available when you do not authenticate with another method.
  - Use the following options:

    The AZURETENANTID= server or session option is required.

    The AZUREAUTHCACHELOC= server option is optional.

    For more information, see "Configuration File Options" in *SAS Viya Platform: SAS Cloud Analytic Services*.
  - The following options are all required:

    accountName=

    applicationID=

    fileSystem=

    identity=

    secret=
  - On a user's first attempt to access data by using device-code authentication, an error message is printed in the SAS log. The user must authenticate with

Microsoft. For details about that process, see "Requirements for Device Code Flow Authentication" on page 140.

Inserting Multiple Values for Microsoft Fabric Data Warehouse

To load data to Microsoft Fabric Data Warehouse, you might find that data loads occur faster when SAS/ACCESS uses multi-value Inserts. To enable this syntax, set insertMultiValue=TRUE and ensure that you set bulkLoad=FALSE. Using multi-value Inserts is often faster when you are working with smaller tables. For more information, see "insertMultiValue= Data Connector Option" on page 512.

> **TIP** Try different values for insertBuff= to improve load performance.

Supported Microsoft SQL Server Data Types

The following table shows the data types that can be loaded from Microsoft SQL Server into CAS. This table also shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.21 *Supported Microsoft SQL Server Data Types*

Microsoft SQL Server Data Type	CAS Data Type
Character	
BIT(*n*)	VARCHAR
CHAR	CHAR
NCHAR	CHAR
NVARCHAR	VARCHAR
VARCHAR	VARCHAR
TEXT	VARCHAR
Numeric	
BIGINT	INT64
INTEGER	INT32
SMALLINT	INT32
TINYINT	INT32

Microsoft SQL Server Data Type	CAS Data Type
DOUBLE PRECISION	DOUBLE
DECIMAL(*p*, *s*)	DOUBLE
FLOAT	DOUBLE
NUMERIC	DOUBLE
REAL	DOUBLE
MONEY	DOUBLE
SMALLMONEY	DOUBLE
Date and Time	
DATE	DOUBLE (formatted at DATE*w.*)
TIME(*p*) (without time zone)	DOUBLE (formatted as TIME*w.*)
DATETIME(*p*) (without time zone)	DOUBLE (formatted as DATETIME*w.*)
DATETIME2(*p*)	DOUBLE (formatted as DATETIME*w.*)

Supported Data Types for Microsoft Fabric

This table shows the data types that can be loaded from Microsoft Fabric Data Warehouse or Microsoft Fabric Lakehouse. This table also shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.22 *Supported Microsoft Fabric Data Types*

Microsoft SQL Server Data Type	CAS Data Type
Character	
CHAR(*n*)	CHAR
TEXT	VARCHAR

Microsoft SQL Server Data Type	CAS Data Type
UNIQUEIDENTIFIER	CHAR
VARCHAR(*n*)	VARCHAR
Numeric	
BIGINT	INT64
INT	INT32
SMALLINT	INT32
BIT	INT32
DECIMAL(*p*, *s*)	DOUBLE
FLOAT	DOUBLE
NUMERIC	DOUBLE
REAL	DOUBLE
Date and Time	
DATE	DOUBLE (formatted at DATE*w.*)
TIME(*p*) (without time zone)	DOUBLE (formatted as TIME*w.*)
DATETIME2(*p*)	DOUBLE (formatted as DATETIME*w.*)
Binary	
VARBINARY(*n*)	VARCHAR
VARBINARY(*max*)	VARCHAR

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Add a Microsoft SQL Server Database as a Data Source for SAS Cloud Analytic Services

Use the CASLIB statement to establish a connection between your Microsoft SQL Server source data and a caslib, SQLservercaslib. All options in this example are required in the CASLIB statement except for PASSWORD=.

In this example, the Microsoft SQL Server data is stored at the location designated by the SQLSERVER_DSN= option.

```
caslib SQLservercaslib desc='Microsoft SQL Server Caslib'
     dataSource=(srctype='sqlserver',
                 username='myuser1',
                 password='myPwd',
                 sqlserver_dsn='dbsqlserver');
```

Example 2: Load Microsoft SQL Server Data into SAS Cloud Analytic Services Using PROC CASUTIL

```
proc casutil;
   list files incaslib="SQLservercaslib";          1
   load casdata="mySQLsvrdata" incaslib="SQLservercaslib"
outcaslib="casuser"
        casout="class_from_SQLservercaslib";       2
```

```
   list files incaslib="casuser";                    3
   contents casdata="class_from_SQLservercaslib" incaslib="casuser";
                                                     4
quit;
```

1 List the tables in SQLservercaslib before loading your data.

2 Load the table mySQLsvrdata from SQLservercaslib into caslib Casuser. Call the new table class_from_SQLservercaslib.

3 List the tables in Casuser to see the newly created table, class_from_SQLservercaslib, that you loaded.

4 List information about the newly loaded table, including column names, data types, and so on.

Example 3: Bulk Load to ADLS2 Storage

Here is an example that shows how you might bulk load data to ADLS2 storage. This example sets the AZURETENANTID= value in an OPTIONS statement, using the CASSESSOPS= option. The CASLIB statement specifies that you are using a Microsoft SQL Server data connector to connect the CAS server to the Myschema schema in the Datasource database on Azure. Refer to this connection with the Sqls casref. The Shared Access Signature is the authentication method for the COPY command, and the shared access signature is provided with the secret= data connector option. Any temporary files are deleted from the CAS server and from Azure storage after a process completes.

The CASUTIL procedure takes input from the Casuser casref and outputs to the Sqls Azure location. The SAVE statement bulk loads the TEST1 CAS table to the Test1_out table in Azure. The defaultDir= option specifies where temporary files reside. These temporary files contain a log of any rejected rows and associated errors. The accountName= and applicationID= values specify the user ID and application ID, respectively, to use when writing to Azure storage. These values must be specified before you can write to Azure, because you did not specify them in the CASLIB statement. You must also provide the fileSystem= value. The data is not being compressed, and the data that you are loading replaces any existing Test1_out table.

```
options cassessopts=(azuretenantid="user-tenant-id");

caslib sqls datasource=(srctype='sqlserver',
                        database='datasource',
                        username='user-id',
                        password='password',
                        schema='myschema',
                        identity='Shared Access Signature',
                        secret='shared-access-signature',
                        deleteDataFiles=true);

proc casutil incaslib="CASUSER" outcaslib="sqls";
    save casdata="TEST1" casout="test1_out"
    options={bulkload=true,
             defaultdir='default-directory',
             accountName='account-name',
             applicationId='application-id',
             fileSystem='filesystem-name',
```

```
                    compress=false} replace;
run;
```

MongoDB Data Connector

Enables you to transfer data between MongoDB and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

Note: All users with SAS/ACCESS Interface to MongoDB can use SAS Data Connector to MongoDB.

See: MongoDB LIBNAME statement

Examples: Establish a connection between your MongoDB database and SAS Cloud Analytic Services.

```
caslib mycaslib desc='MongoDB Caslib'
     dataSource=(srctype='mongodb'
                 server='mymachine.com'
                 username='user1'
                 password='myPwd'
                 db="database-name"
                 schema_collection="mySchema");
```

Override the user and password values.

```
proc casutil;
   load casdata="mycas.examp" casout="myDBdata" incaslib='mycaslib'
       outcaslib='casuser'
       dataSourceOptions=(
         username='user5'
         password='myPwd5');
quit;
```

Details

About the MongoDB Data Connector

The MongoDB data connector enables you to establish a connection between the MongoDB database and CAS. The MongoDB data connector enables you to load data into CAS and to create, modify, and delete tables in the MongoDB database.

For additional information about working with a MongoDB database and working with SAS/ACCESS schemas for MongoDB, see "SAS/ACCESS Interface to MongoDB" in *SAS/ACCESS for Nonrelational Databases: Reference*.

Data Connector Options for MongoDB

Data connector options are used in the context of different statements that connect your data in MongoDB with CAS. For each described option, the applicable

statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.23 *Data Connector Options for MongoDB*

Data Connector Option	Default Value
auth=	AUTO
catalog=	Active caslib
charMultiplier=	1.0
conOpts=	none
database=	none
dbType=	none
insertBuff=	Calculated automatically based on row size
max_char_len=	32767
numReadBuffers=	4
numReadNodes=	1
numWriteNodes=	1
password=	none
platform=	none
port=	27017
readBuff=	Calculated automatically based on row size
schema=	none
schema_auth=	AUTH= value (unless schema_port= differs from port= or schema_server= differs from server=. The default is the password= value in that case.)
schema_collection=	sas-default-schema
schema_db=	database= value
schema_password=	password= value

Data Connector Option	Default Value
schema_platform=	platform= value
schema_port=	port= value
schema_server=	server= value
schema_ssl_ca=	ssl_ca= value
schema_ssl_cert=	ssl_cert= value
schema_uid=	user= value
server=	none
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "mongodb" for MongoDB data.
ssl_ca=	none
ssl_cert=	none
useNarrowCharacterTypes=	FALSE
userName=	none

Importance of Specifying the SCHEMA_COLLECTION= Data Connector Option

It is recommended that you always specify the SCHEMA_COLLECTION= option in your CASLIB statements for MongoDB. This requires that a connection refer to a JSON file that describes the schema for the specified collection. If a schema file for the given collection does not exist, then a scan is performed and the file is created. Specifying the SCHEMA_COLLECTION= option improves data connector performance, because a new scan is not required each time that you access MongoDB. Specifying SCHEMA_COLLECTION= also ensures that all users who connect using the same SCHEMA_COLLECTION= value are accessing the same stored schema.

If you do not specify the SCHEMA_COLLECTION= option, the default schema value, sas-default-schema, is assumed. If this default schema already exists, you might be accessing data that you do not intend to access. For example, another user might have created this schema since your last SAS session. If the default

schema does not already exist, SAS/ACCESS scans the MongoDB data to generate a new stored schema for sas-default-schema. This is a resource intensive task.

Creating a Temporary Schema

If you want to work with a schema without storing the schema file, you can create a temporary schema. To do this, specify SCHEMA_COLLECTION='$temp$'. You might wish to do this if you do not have permission to Write to a MongoDB data source, or if you want to perform testing. However, be aware that although the schema file is not stored, any changes that you make to the source data, such as adding tables or deleting data are passed onto the MongoDB data.

Deleting MongoDB Data

You can delete specific rows from a table in the MongoDB schema using the DELETE statement for PROC FEDSQL. Specify table names as they are listed in the output from a LIST statement in PROC CASUTIL.

In this example, the first DELETE statement deletes all rows where Surname is 'Bloggs' from the Sample_name table. The next DELETE statement deletes the second entry in the array data in table Sample_creditcards.

```
caslib mycaslib desc='MongoDB Caslib'
         dataSource=(srctype='mongodb'
                     server='mymachine.com'
                     username='user1'
                     password='myPwd'
                     db="database-name"
                     schema_collection="mySchema");

proc fedsql;
   delete from mycaslib."sample_name" where "surname"='Bloggs';
   delete from mycaslib."sample_creditcards" where "_index"=1;
quit;
```

Supported MongoDB Data Types

The following table shows the data types that can be loaded from MongoDB into CAS. This table also shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.24 *Supported MongoDB Data Types*

MongoDB Data Type	CAS Data Type	Notes
		Character
JAVASCRIPT	CHAR	Default format is *$w.*
REGEX	CHAR	Default format is *$w.*

MongoDB Data Type	CAS Data Type	Notes
STRING	CHAR	Default format is *$w.*
		Numeric
BOOL	NUMERIC	Default format is *w.* In SAS, the value 0 is False, and any other numeric value is interpreted as True.
DECIMAL	DOUBLE	Default format is *w.*
DOUBLE	DOUBLE	Default format is *w.*
INT	NUMERIC	Default format is *w.*
LONG	NUMERIC	Default format is *w.*
		Others
ARRAY	Varies	Converted to a child table in SAS with variables of the appropriate type.
BINDATA	CHAR	Contains a hexadecimal representation of the binary data. Default format is *$w.*
DATE	DATETIME	Date values become SAS dates, formatted based on your specifications. For more information, see "How Date Values Are Handled" in *SAS/ACCESS for Nonrelational Databases: Reference*.
NULL	CHAR	MongoDB NULL values are read in as missing character values.
OBJECT	Varies	JSON objects typically become child tables with variables of the appropriate data type. However, if a JSON object that is not accounted for in the current schema map is encountered, then that object is returned as a missing value. This might occur if a JSON object is added after the initial scan to determine the schema for a collection.
OBJECTID	CHAR	Default format is *$w.*

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Example: Add a MongoDB Database as a Data Source and Load Data into SAS Cloud Analytic Services

```
caslib mycaslib desc='MongoDB Caslib'                    1
     dataSource=(srctype='mongodb',
                 server='MongoDBserver'
                 username='user1',
                 password='myPwd',
                 database='myDatabase'
                 schema_collection='mySchema');

proc casutil;
   list files incaslib="mycaslib";                       2
   load casdata="myDBdata" incaslib="mycaslib" outcaslib="casuser"
        casout="class_from_mycaslib";                    3
   list files incaslib="casuser";                        4
   contents casdata="class_from_mycaslib" incaslib="casuser";
```

```
                                                                    5
   quit;
```

1 Establish a connection between your MongoDB source data and a caslib, mycaslib.

2 List the tables in Mycaslib before loading your data.

3 Load the table myDBdata from Mycaslib into caslib Casuser. Call the new table class_from_mycaslib.

4 List the tables in Casuser to see the newly created table, class_from_mycaslib, that you loaded.

5 List information about the newly loaded table, including column names, data types, and so on.

MySQL Data Connector

Enables you to transfer data between MySQL and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

See: MySQL LIBNAME statement, "Where to Specify Data Connector Options"

Examples: Establish a connection between your MySQL database and SAS Cloud Analytic Services.

```
caslib mycaslib desc='MySQL Caslib'
     dataSource=(srctype='mysql'
                 host='mySQLserver'
                 username='user1'
                 password='myPwd'
                 database="database-name");
```

Override the user and password values.

```
proc casutil;
   load casdata="examp" casout="myDBdata" incaslib='mycaslib'
       outcaslib='casuser'
       dataSourceOptions=(
         username='user5'
         password='myPwd');
quit;
```

Details

Data Connector Options for MySQL

Data connector options are used in the context of different statements that connect your data in MySQL with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about

where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.25 Data Connector Options for Data Sources That Are Compatible with MySQL

Data Connector Options	Default Value
append=	FALSE
authenticationDomain=	none
authScope=	`https://ossrdbms-aad.database.windows.net/user_impersonation`
baseDir=	none
blDefaultDir=	current location for storing temporary data files
bulkLoad=	FALSE
charMultiplier=	1.0
conOpts=	none
database=	none
dbType=	none
default_auth_plugin=	CACHING_SHA2_PASSWORD
enable_cleartextPlugin=	FALSE
host=	none
numReadBuffers=	4
numReadNodes=	1
numWriteNodes=	1
password=	none
port=	none
readBuff=	Calculated automatically based on row size

Data Connector Options	Default Value
schema=	none
sliceColumn=	none
sliceExpressions=	none
socket=	`/tmp/mysql.sock`
srcType=	none Use "mysql" for MySQL data.
ssl_ca=	none
ssl_cert=	none
ssl_cipher=	none
ssl_key=	none
useNarrowCharacterTypes=	FALSE
userName=	none

Authenticate to Microsoft Azure with Single Sign-On

Single sign-on for the SAS Viya platform uses Microsoft Entra ID (formerly Azure Active Directory) to control authentication. This requires an administrator to configure Microsoft Entra ID as an OpenID Connect provider. For more information, see "Scenario: OIDC with Microsoft Entra ID" in *SAS Viya Platform: Authentication*.

To use single sign-on authentication to connect to Microsoft Azure using Microsoft Entra ID, omit the userName=, password=, or authenticationDomain= options. This establishes a connection to MySQL that uses the default authentication scope for Microsoft Azure. To use a different authentication scope, you can specify the authScope= option. Here is a sample CASLIB statement that can connect to a MySQL instance on Microsoft Azure:

```
caslib mysqldb datasource=(srctype='mysql',
                        server="mydataenvironment.mysql.database.azure.com",
                        database="mysqldata");
```

Note: If single sign-on is not configured, you can specify the Microsoft Entra ID user ID and password (access token) directly with the userName= and password= options. When you use the userName= and password= options, you must also specify enable_clearTextPlugin=YES.

Bulk Loading to MySQL

Bulk loading enables you to move large amounts of data from CAS to MySQL as efficiently as possible. Bulk loading uses a MySQL LOAD DATA command.

Here are the options that you use when you bulk load data to MySQL:

- blDefaultDir=
- bulkLoad=

Supported MySQL Data Types

The following table shows the data types that can be loaded from MySQL into CAS. This table also shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.26 *Supported MySQL Data Types*

MySQL Data Type	CAS Data Type
Character	
CHAR(*n*)	CHAR
VARCHAR(*n*)	VARCHAR
TEXT	VARCHAR
TINYTEXT	VARCHAR
MEDIUMTEXT	VARCHAR
LONGTEXT	VARCHAR
BLOB	VARCHAR
TINYBLOB	VARCHAR
MEDIUMBLOB	VARCHAR
LONGBLOB	VARCHAR
ENUM	CHAR
SET	CHAR
Numeric	
BIT	INT32

MySQL Data Type	CAS Data Type
INT	INT32
TINYINT	INT32
SMALLINT	INT32
MEDIUMINT	INT32
BIGINT	INT64
DECIMAL	DOUBLE
DOUBLE	DOUBLE
FLOAT	DOUBLE
Date and Time	
DATE	DOUBLE (formatted as DATE*w.*)
DATETIME	DOUBLE (formatted as DATETIME*w.d*)
TIME	DOUBLE (formatted as TIME*w.d*)
TIMESTAMP	DOUBLE (formatted as DATETIME*w.d*)
YEAR	INT32
Other	
JSON[1]	VARCHAR

1 Support of the JSON data type requires MySQL 5.7.8 or higher.

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Add a MySQL Database as a Data Source for SAS Cloud Analytic Services

Use the CASLIB statement to establish a connection between your MySQL source data and a caslib, MySQLcaslib. All of the options supplied in this example are required in the CASLIB statement, except the password= option.

In this example, the MySQL data is stored at the location designated by the host= and database= options.

```
caslib mycaslib desc='MySQL Caslib'
     dataSource=(srctype='mysql',
                 host='MySQLserver'
                 username='user1',
                 password='myPwd',
                 database='myDatabase');
```

Example 2: Load MySQL Data into SAS Cloud Analytic Services Using PROC CASUTIL

```
proc casutil;
   list files incaslib="mycaslib";                    1
   load casdata="myDBdata" incaslib="mycaslib" outcaslib="casuser"
        casout="class_from_mycaslib";                 2
   list files incaslib="casuser";                     3
   contents casdata="class_from_mycaslib" incaslib="casuser";
                                                      4
```

```
quit;
```

1. List the tables in Mycaslib before loading your data.
2. Load the table myDBdata from Mycaslib into caslib Casuser. Call the new table class_from_mycaslib.
3. List the tables in Casuser to see the newly created table, class_from_mycaslib, that you loaded.
4. List information about the newly loaded table, including column names, data types, and so on.

Netezza Data Connector

Enables you to transfer data between Netezza and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

Note: All users with SAS/ACCESS Interface to Netezza can use SAS Data Connector to Netezza

See: Netezza LIBNAME statement connection

Examples: Establish a connection between your Netezza database and SAS Cloud Analytic Services.

```
caslib netezzacaslib desc='Netezza Caslib'
     dataSource=(srctype='netezza'
                 server='NZserver'
                 username='user1'
                 password='myPwd1'
                 database="NZdatabase-name"
                 schema='myschema');
```

Override the user and password values.

```
proc casutil;
   load casdata="nzexamp" casout="myNZdata" incaslib="netezzacaslib"
      outcaslib="casuser"
      dataSourceOptions=(
         username='user5'
         password='myPwd2');
quit;
```

Details

Data Connector Options for Netezza

Data connector options are used in the context of different statements that connect your data in Netezza with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about

where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.27 *Data Connector Options for Netezza*

Data Connector Option	Default Value
authenticationDomain=	none
catalog=	Active caslib
charMultiplier=	1.0
conOpts=	none
database=	none
insertBuff=	Calculated automatically based on row size
netezza_dsn=	none
numReadBuffers=	4
numWriteNodes=	1
password=	none
port=	5480
readBuff=	Calculated automatically based on row size
schema=	none
server=	none
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "netezza" for Netezza data.
useNarrowCharacterTypes=	FALSE
userName=	none

Case Sensitivity with Netezza

All quoted values are passed to the Netezza database exactly as you type them. This means that you must specify all values, such as table names or ID values, using the same capitalization that is used in the Netezza database.

Supported Netezza Data Types

This table shows the data types that can be loaded from Netezza into CAS. It also shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.28 *Supported Netezza Data Type*

Data Type	CAS Data Type
Character	
CHAR(*n*)	CHAR
INTERVAL[fields]	CHAR
NCHAR(*n*)	CHAR
NVARCHAR(*n*)	VARCHAR
VARCHAR(*n*)	VARCHAR
Binary	
ST_GEOMETRY	VARCHAR
VARBINARY(*n*)	VARCHAR
Numeric	
BIGINT	INT64
BYTEINT	INT32
BOOLEAN	INT32
DOUBLE PRECISION	DOUBLE
INTEGER	INT32
NUMERIC(*p*, *s*)	DOUBLE
REAL	DOUBLE

Data Type	CAS Data Type
SMALLINT	INT32
Date and Time	
DATE	DOUBLE (formatted as DATE9.)
TIME	DOUBLE (formatted as TIME8.)
TIMESTAMP	DOUBLE (formatted as DATETIME25.6)

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Add a Netezza Database as a Data Source for SAS Cloud Analytic Services

Use the CASLIB statement to establish a connection between your Netezza source data and a caslib, Netezzacaslib. All options in this example are required in the CASLIB statement except the SCHEMA= option.

In this example, the Netezza data is stored at the location designated by the SERVER= and DATABASE= options.

```
caslib Netezzacaslib desc='Netezza Caslib'
     dataSource=(srctype='netezza',
                 server='NZserver'
                 username='user1',
                 password='myPwd1',
                 database='NZdatabase',
                 schema='myschema');
```

Example 2: Load Netezza Data into SAS Cloud Analytic Services Using PROC CASUTIL

```
proc casutil;
   list files incaslib="Netezzacaslib";                    1
   load casdata="myNZdata" incaslib="Netezzacaslib" outcaslib="casuser"
        casout="class_from_Netezzacaslib";                 2
   list files incaslib="casuser";                          3
   contents casdata="%upcase(class_from_Netezzacaslib)"
incaslib="casuser";
                                                           4
quit;
```

1 List the tables in Netezzacaslib before loading your data.

2 Load the table myNzdata from Netezzacaslib into caslib Casuser. Call the new table class_from_Netezzacaslib.

Note: You must specify Netezza table names using the capitalization that is used in the database.

3 List the tables in Casuser to see the newly created table, class_from_Netezzacaslib, that you loaded.

4 List information about the newly loaded table, including column names, data types, and so on.

ODBC Data Connector

Enables you to transfer data between an ODBC data source and CAS.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, SAVE statements

Note: All users with SAS/ACCESS Interface to ODBC can use SAS Data Connector to ODBC.

See: ODBC LIBNAME statement connection

Example: Establish a connection to ODBC data. (Note: The GLOBAL option is restricted to administrators.)

```
caslib odbccaslib desc="SQLviaODBCtoCaslib"
     datasource=(srctype="odbc"
                      username="user1"
                      password="password1"
                      odbc_dsn="dbodbc") global;
```

Details

Data Connector Options for ODBC

Data connector options are used in the context of different statements that connect your data in an ODBC compatible data source with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.29 *Data Connector Options for ODBC Data Sources*

Data Connector Option	Default Value
authenticationDomain=	none
catalog=	Active caslib
charMultiplier=	1.0
client_encoding=	none
conOpts=	none
dbCreateTableOpts=	none
dm_unicode=	UTF-8
insertBuff=	Calculated automatically based on row size
insertSQL=	FALSE
invalidDate=	WARN when loading multiple records

Data Connector Option	Default Value
	ERROR when loading records individually
numReadBuffers=	4
numReadNodes=	1
numWriteNodes=	1
odbc_dsn=	none
password=	none
readBuff=	Calculated automatically based on row size
scanStringColumns=	none
schema=	none
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "odbc" for ODBC data sources.
useNarrowCharacterTypes=	FALSE
userName=	none

Supported ODBC Data Types

The following table lists the supported data types that the ODBC data connector can load from ODBC into SAS Cloud Analytic Services. This table also shows the resulting data type in CAS.

Note: Be aware that when performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that are read are rounded to meet this condition. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data.

Table 7.30 Data Type Conversions When Loading Data from ODBC into CAS

ODBC Data Type	CAS Data Type
	Character
SQL_BINARY	CHAR
SQL_CHAR	CHAR
SQL_LONGVARBINARY	VARCHAR
SQL_LONGVARCHAR	VARCHAR
SQL_VARBINARY	VARCHAR
SQL_VARCHAR	VARCHAR
SQL_WCHAR	CHAR
SQL_WLONGVARCHAR	VARCHAR
SQL_WVARCHAR	VARCHAR
	Numeric
SQL_BIGINT	BIGINT (INT64)
SQL_BIT	DOUBLE
SQL_DECIMAL	DOUBLE
SQL_DOUBLE	DOUBLE
SQL_FLOAT	DOUBLE
SQL_INTEGER	INTEGER (INT32)
SQL_NUMERIC	DOUBLE
SQL_REAL	DOUBLE
SQL_SMALLINT	INTEGER (INT32)
SQL_TINYINT	INTEGER (INT32)
	Date and Time
SQL_TYPE_DATE	DOUBLE (formatted as DATE)

ODBC Data Type	CAS Data Type
SQL_TYPE_TIME	DOUBLE (formatted as TIME)
SQL_TYPE_TIMESTAMP	DOUBLE (formatted as DATETIME)

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Establish a Connection between a Microsoft SQL Database and a Caslib through ODBC

Use the CASLIB statement to add a Microsoft SQL database as a data source for SAS Cloud Analytic Services using an ODBC connection. For connections to Microsoft SQL, specify CATALOG="*".

In this example, GLOBAL specifies that the data source is potentially available to all sessions. You might not have access to the GLOBAL option if you are not an administrator. For more information, see "CASLIB Statement" on page 745.

```
caslib odbccaslib desc="SQLviaODBCtoCaslib"
     datasource=(srctype="odbc"
                      username="user1"
                      password="password1"
                      odbc_dsn="dbodbc"
                      catalog="*") global;
```

Example 2: PROC CASUTIL: Load Data from an External Database into SAS Cloud Analytic Services

```
proc casutil;
  list files incaslib="odbccaslib";       /* 1 */
  load casdata="myDBdata" incaslib="odbccaslib" outcaslib="casuser"
       casout="class_from_odbccaslib";    /* 2 */
  list tables incaslib="casuser";         /* 3 */
  contents casdata="class_from_odbccaslib" incaslib="casuser";
                                          /* 4 */
quit;
```

1. List the files in odbccaslib before loading your data.
2. Load the table myDBdata from an external database into memory in caslib Casuser. Call the new table class_from_oraclecaslib.
3. List the tables in casuser to see the newly created table, class_from_odbccaslib, that you loaded.
4. List information about the newly loaded table, including column names, data types, and so on.

Oracle Data Connector

Enables you to transfer data between Oracle and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

Note: All users with SAS/ACCESS Interface to Oracle can use SAS Data Connector to Oracle.

See: Oracle LIBNAME statement connection

Examples: Establish a connection between your Oracle database and SAS Cloud Analytic Services.

```
caslib orcaslib desc='Oracle Caslib'
   datasource=(srctype='oracle'
               username='MYUSR1'
               password='myPwd'
               path="//machine.lnx.com:1521/exadat");
```

Override USER= and PASSWORD= values.

```
proc casutil;
   load casdata="%upcase(mycas.orexamp)" incaslib="orcaslib" outcaslib="casuser"
      dataSourceOptions=(username='USER5' password='pwd5');
```

Details

Data Connector Options for Oracle

Data connector options are used in the context of different statements that connect your data in Oracle with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.31 *Data Connector Options for Oracle*

Data Connector Option	Default Value
append=	FALSE
authenticationDomain=	none
blErrors= on page 419	1000000
blSkip= on page 433	0
blLoad= on page 422	none
blParallel= on page 429	FALSE
blRecoverable= on page 431	FALSE
blSkipIdxMaint= on page 433	FALSE
blSkipUnusableIdx= on page 434	TRUE **Note:** For some databases, the default is TRUE. For more information, see blSkipUnusableIdx= Data Connector Option Details on page 435.
bulkLoad= on page 440	FALSE
charMultiplier=	1.0
conOpts=	none
dbCreateTableOpts=	none

Data Connector Option	Default Value
dbType=	none
insertBuff=	Calculated automatically based on row size
invalidDate=	WARN
max_string_size=	NONE
numReadBuffers=	4
numReadNodes=	1
numWriteNodes=	1
ora_encoding=	UNICODE
password=	none
path=	none
readBuff=	Calculated automatically based on row size
scanStringColumns=	none
schema=	none
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "oracle" for Oracle data.
useNarrowCharacterTypes=	FALSE
userName=	none

Connect CAS to Oracle Using an Oracle Wallet

An Oracle Wallet is an encrypted file that contains the database credentials that you use to connect to Oracle. When you use Oracle Wallet to control authentication, you are not required to store user IDs or passwords in your SAS programs.

Use the path= option to specify the location and other variables that you use to access an Oracle Wallet. In the following example, save the connection information for accessing an Oracle Wallet in a macro variable called Wallet_path. The

userName= and password= options are shown in the CASLIB statement, but these values might not be required if your wallet contains user authentication certificates. See your Oracle Wallet documentation for more information about how to configure and access a wallet.

```
%let wallet_path=(description =
                     (address_list=
                        (address =
                           (protocol=tcps)
                           (host=<oracleserver>.oraclecloud.com)
                           (port=<port-value>)
                        )
                        (connect_data =
                           (server = dedicated)
                           (service_name=<my-service-name>)
                        )
                        (security=(my_wallet_directory = <wallet-path>))
                  );

caslib mydb desc='Oracle Wallet CASLIB'
   datasource=(srctype='oracle',
           username='<my-userID>',
           password='<my-password>',
           path="&wallet_path");
```

For information about using an Oracle Wallet in a LIBNAME statement, see "Connect to Oracle by Using the Oracle Wallet" in *SAS/ACCESS for Relational Databases: Reference*.

Case Sensitivity with Oracle

All quoted values are passed to the Oracle database exactly as you enter them. This means that you must specify all values, such as table names or ID values and using the same capitalization as the Oracle database uses.

Supported Oracle Data Types

The Oracle data connector supports these data types for loading data from Oracle into CAS. This table shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.32 *Supported Oracle Data Types*

Oracle Data Type	CAS Data Type
Character	
CHAR	CHAR
CLOB	VARCHAR

Oracle Data Type	CAS Data Type
VARCHAR	VARCHAR
Numeric	
ARRAY	ERROR (not supported)
BOOLEAN	ERROR (not supported)
DOUBLE	DOUBLE
Others	
Other large objects (for example, BLOB)	ERROR (not supported)
MAP	ERROR (not supported)
STRUCT	ERROR (not supported)
UNION	ERROR (not supported)

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the

CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Add an Oracle Database as a Data Source for SAS Cloud Analytic Services

Use the CASLIB statement to establish a connection between your Oracle source data and a caslib, Oralib. All of the options supplied in this example are required in the CASLIB statement, except the PASSWORD= option.

In this example, Oracle data is stored in the location that the PATH= option designates.

```
caslib oralib
   datasource=(srctype='oracle',
               username='MYUSER1',
               password='myPwd',
               path="myORAdata");
```

Example 2: Load Oracle Data into SAS Cloud Analytic Services Using PROC CASUTIL

```
proc casutil;
   list tables incaslib="casuser";                    1
   load casdata="MYORADATA" incaslib="orcaslib" outcaslib="casuser"
        casout="ORAdata_from_orcaslib";               2
   list tables incaslib="casuser";                    3
   contents casdata="%upcase(class_from_orcaslib)" incaslib="casuser";
                                                      4
quit;
```

1 List the tables in casuser before loading your data.

2 Load the table myORAdata from Oracle into caslib Casuser. Call the new table ORAdata_from_orcaslib.

Note: You must list Oracle table names with capitalization that matches that in the Oracle database.

3 List the tables in casuser again to see the newly created table, ORAdata_from_orcaslib, that you loaded.

4 List information about the newly loaded table, including column names, data types, and so on.

ORC Data Connector

Enables you to load data from Apache Optimized Row Columnar (ORC) tables into CAS.

Valid in: CASLIB statement

Applies to: CAS procedure, CASUTIL procedure

Restrictions: Only Linux for x64 is supported.

Partitioned and ACID tables in Hive are not supported.

Complex data types (LIST, MAP, STRUCT, UNION) are not supported.

The root object type of an ORC table is a STRUCT. (A STRUCT is the only schema that is supported.) The root object type contains a field for each of the top-level columns.

Requirements: When connecting to Azure Data Lake Storage Gen2 using the device code flow, you must set the AZURETENANTID= system option or the AZURETENANTID= session option. You should also review and set the AZUREAUTHCACHELOC= system option if you want a different value from the default.

This data connector supports only ORC files. You must specify FILETYPE="ORC" or a file name that ends with .orc as the file type. Otherwise, the file is not found.

Supports: You can use this with a path or a Microsoft Azure Data Lake Storage Gen2 caslib.

Notes: To access data in your home directory instead of in a local or shared network storage location, use the CASUSER caslib.

All users with a SAS Viya platform offering can use SAS Data Connector to ORC.

Tip: To support ORC tables that Hive creates, the ORC data connector can access a directory of tables as a single logical table. The name of the directory must be specified as the table name. It must also be a subdirectory or subfolder under the defined caslib. To access a directory as a logical table, all tables in the directory must be ORC tables and they must also all have the same schema.

See: For details about the related SAS engine, see *SAS® Viya® Engines for Open-Source File Formats: Reference*.

For information about authentication methods, see "Authentication"and these options: AZUREAUTHCACHELOC= system option, AZURETENANTID= session option, AZURETENANTID= system option

Examples: Add a PATH caslib to access ORC tables in a local or shared network storage location.

```
caslib mycaslib desc='ORC Caslib'
      datasource=(srctype='path'
                  path="/mytest/customer");
```

Add an Azure Data Lake Storage caslib using single sign-on to access ORC tables in Azure Data Lake Storage Gen2.

```
caslib mycaslib desc='ORC in ADLS Caslib'
datasource={srctype='adls'
            accountName='mystorage'
            filesystem='mycontainer'
```

```
            applicationId="d1e9g428-c440-604m-3m32xf787-xf787p1025n1"
           };
```

Load an encrypted ORC table into SAS Cloud Analytic Services.

```
proc casutil;
   load casdata="cars.orc"
        importoptions=(filetype="orc" encryptionPassword="pasquotank");
run;
```

Details

Overview

Apache ORC is an open-source file format that was originally developed for use in Hadoop. To access ORC tables requires a dataSource= option on the addCaslib action with srcType="ADLS" or srcType="PATH" as the specified value.

To support ORC tables that are created by Hive, the ORC data connector can access a directory of tables as a single logical table. The name of the directory must be specified as the table name. It must also be a subdirectory or subfolder under the defined caslib. To access a directory as a logical table, all tables in the directory must be ORC tables and they must also all have the same schema.

The ORC data connector does not support partitioned tables in Hive.

Data Connector Options for ORC

Data connector options are used in the context of different statements that connect your data in ORC data with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.33 *Data Connector Options for ORC*

Data Connector Option	Default Value
compress=	ZLIB
fileType=	none Use "ORC" for ORC table data.
srcType=	none Use "ADLS" or "PATH" for ORC table data.

Supported ORC Data Types

As shown in the table below, the ORC data connector supports the data types for loading data from an ORC table into SAS Cloud Analytic Services. The table shows

the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Complex data types (LIST, MAP, STRUCTURE, UNION) are not supported. Tables that have these complex types cannot be accessed through the ORC data connector.

Table 7.34 *Supported ORC Data Types*

ORC Data Type	CAS Data Type
	Character
CHAR	CHAR[1]
VARCHAR	VARCHAR[2]
STRING	VARCHAR[2]
	Numeric
BIGINT (64-bit)	INT64
BINARY	VARCHAR[2]
DECIMAL	DOUBLE
BOOLEAN (1-bit)	INT32
DOUBLE	DOUBLE
FLOAT	DOUBLE
INT (32-bit)	INT32
SMALLINT (16-bit)	INT32
TINYINT (8-bit)	INT32
	Date and Time
DATE	DOUBLE[3]
TIMESTAMP	DOUBLE[3]
	Others

ORC Data Type	CAS Data Type
complex types[4]	error

1 Values with a length of less than or equal to 255 are a CHAR data type in CAS. Values with a length of greater than 255 are a VARCHAR data type in CAS. Values with lengths that are shorter than the specified length value are padded with spaces.

2 The maximum allowed length in the data store is 65,535. If no length exists in the ORC file, no VARCHAR length in CAS is defined.

3 An ORC type of DATE and TIMESTAMP is stored as a DOUBLE in CAS with a format of DATE9. and DATETIME21.2, respectively.

4 These include LIST, MAP, STRUCT, and UNION.

Issues When Converting Data from ORC to CAS

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.

Examples

Example 1: Access an ORC Data Set Using the PATH CASLIB and the CAS Procedure

```
cas mysess user=&SYSUSERID;

proc cas;

/* Define a PATH type CASLIB called "casuser" and */
/* point it to the location of the ORC data set.  */
addcaslib /
caslib="casuser"
datasource={srctype="path"}
path="/mytest1/wky/orcdata";

/* Use the CAS Procedure to load the ORC data set. */
loadtable /
   caslib="casuser"
   path="car_sales"

/* If the file name does not end with .orc, */
/* the filetype=”orc” must be specified.    */
   importoptions={filetype="orc"}
   casout={name="xyz" replace=TRUE};
run;
quit;
```

Example 2: Create a Microsoft Azure Data Lake Storage Caslib Using Single Sign-on and Save an ORC Data Set with the CAS Procedure

```
proc cas;
addcaslib /
   datasource={srctype="adls"
               accountName="myAccount"
               filesystem="mylib"
              }
   name="adlsdata"
   subdirs=true
   path="/mydata/test";
run;

table.save /
   caslib="adlsdata"
   table="xyz"
   name="ORCsave"
   replace=True
   exportoptions={filetype="orc"};
run;
quit;
```

Example 3: Create a Microsoft Azure Data Lake Storage Caslib and Save an ORC Data Set with the CAS Procedure

```
/* Have a CAS session with the Azure Tenant ID set as a session option
*/
/* or set globally with the AZURETENANTID= system option.
*/
cas casauto
   sessopts=(azureTenantId='b1c24e5c-3625-4593-a430-9552373a0c2f');

proc cas;
addcaslib /
   datasource={srctype="adls"
               accountName="myAccount"
               filesystem="mylib"
               applicationId="xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
              }
   name="adlsdata"
   subdirs=true
   path="/mydata/test";
run;

table.save /
   caslib="adlsdata"
   table="xyz"
   name="ORCsave"
   replace=True
   exportoptions={filetype="orc"};
run;
quit;
```

PC Files Data Connector

Enables you to load data from files of various file types into CAS.

Valid in: CASLIB statement

Applies to: CAS procedure

Note: All users with SAS/ACCESS Interface to PC Files can use SAS Data Connector to PC Files.

Example: Load the Titanic table into SAS Cloud Analytic Services. Data was obtained from http://biostat.mc.vanderbilt.edu/DataSets.

```
options validvarname=any;
proc casutil;
   load file="/path/to/titanic3.xls" casout="titanic3"
      importOptions=(filetype="xls" getnames=true);
quit;
```

Details

Data Connector Options for PC Files

Data connector options are used in the context of different statements that connect your data in PC Files with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.35 *Data Connector Options for PC Files*

Data Connector Option	Default Value
dbCreateTableOpts=	none
fileType=	XLS
getNames=	TRUE
getSheetNames=	FALSE
path=	none
range=	none (imports an entire worksheet)

Data Connector Option	Default Value
sheet=	none (imports an entire worksheet)

SAS Data and AI Studio Tip

Note: SAS Data and AI Studio was known as SAS Studio prior to 2026.06.

If you can navigate to the file in the **Server Files and Folders** section, you can use the CASUTIL procedure with the LOAD FILE= syntax to load data into SAS Cloud Analytic Services.

For large files, if SAS Cloud Analytic Services can access the file, you can use the CASUTIL procedure with the LOAD CASDATA= syntax.

Specifying a Range of Cells

The specification for the RANGE= option is "*sheet name*$*top left cell*:*bottom right cell*".

- If you omit the worksheet name, the first worksheet in the workbook is used. However, you must include the dollar sign ($) character if you want to specify a range of cells on the first worksheet. For example, RANGE="$A1:E1" specifies to read five cells from the first row of the first worksheet.
- If you omit the top left cell value, A1 is used as the first cell.
- If you omit the bottom right cell value, the last row and column in the worksheet is used.

Supported PC Files Data Types

The PC Files data connector supports these data types for loading data from PC Files into CAS. This table shows the resulting data type for the data after it has been loaded into SAS Cloud Analytic Services. This table shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.36 *Supported PC Files Data Types*

PC Files Data Type	CAS Data Type
DTA[1]	VARCHAR
JMP	VARCHAR
SPSS	VARCHAR
XLS	VARCHAR

PC Files Data Type	CAS Data Type
XLSX	VARCHAR

1 When string data is being exported, this data type supports Unicode and more than 32,767 variables.

Examples

Example 1: Use getNames= in a Caslib to Import Table Column Names

This example uses testlib caslib to import table column names from an XLSX file with getNames=.

```
proc cas;
   session mysess;
      parmlist={
      casLib="testlib",
      importOptions={fileType="xlsx", getnames="true",
      sheet="test_sheet"} path="mytest.xlsx"
      casout={caslib="testlib",name="test_out"}
      };
   loadtable result=r / parmlist;
run;
```

Example 2: Load a Table into CAS

In this example, the Titanic table is loaded into SAS Cloud Analytic Services. Data was obtained from http://biostat.mc.vanderbilt.edu/DataSets.

```
options validvarname=any;
proc casutil;
   load file="/path/to/titanic3.xls" casout="titanic3"
      importOptions=(filetype="xls" getnames=true);
quit;
```

Example 3: Use range= with the CASUTIL Procedure to Specify Data to Import

This example shows how to use the range= option to specify a block of cells to import. The file is available from http://catalog.data.gov/dataset/2010-federal-stem-education-inventory-dataset. One characteristic of the file is that the first two rows are used to provide column descriptions and the columns exceed SAS naming rules. The range= option skips the first two rows. getnNames=false specifies to read data in Row 3 as data values rather than column names.

Disclaimer: SAS may reference other websites or content or resources for use at Customer's sole discretion. SAS has no control over any websites or resources that are provided by companies or persons other than SAS. Customer acknowledges and agrees that SAS is not responsible for the availability or use of any such external sites or resources, and does not endorse any advertising, products, or other

materials on or available from such websites or resources. Customer acknowledges and agrees that SAS is not liable for any loss or damage that may be incurred by Customer or its end users as a result of the availability or use of those external sites or resources, or as a result of any reliance placed by Customer or its end users on the completeness, accuracy, or existence of any advertising, products, or other materials on, or available from, such websites or resources.

```
proc casutil;
   load file="2010 Federal STEM Education Inventory Dataset.xls"
      casout="stem2010"
      importoptions=(filetype="xls"
         getnames=false range="Sheet1$A3:IV254");

  /* Be sure that the data type is listed as DOUBLE
   * for Columns G, H, and I.                        */
   contents casdata="stem2010";
run;

libname mycas cas;
proc print data=mycas.stem2010(caslib="casuser" obs=2);
   var a -- j;
run;
```

PostgreSQL Data Connector

Enables you to transfer data between PostgreSQL and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

Note: All users with SAS/ACCESS Interface to PostgreSQL can use SAS Data Connector to PostgreSQL.

See: PostgreSQL LIBNAME statement

Examples: Establish a connection between your PostgreSQL database and SAS Cloud Analytic Services.

```
caslib postgrescaslib desc='PostgreSQL Caslib'
     dataSource=(srctype='postgres'
                 server='PGserver'
                 username='user1'
                 password='myPwd'
                 database="PGdatabase-name"
                 schema='myschema');
```

Override the user and password values.

```
proc casutil;
   load casdata="pgexamp" casout="myPGdata" incaslib="postgrescaslib"
      outcaslib="casuser"
      dataSourceOptions=(
         username='user5'
         password='myPwd'
            );
quit;
```

Details

Data Connector Options for PostgreSQL

Data connector options are used in the context of different statements that connect your data in PostgreSQL with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.37 *Data Connector Options for PostgreSQL*

Data Connector Option	Default Value
allow_unquoted_names=	FALSE
append=	FALSE
authenticationDomain=	none
authenticationScope=	`https://ossrdbms-aad.database.windows.net/user_impersonation`
blDelimiter=	, (comma)
blEscape=	" (double quotation mark)
blFormat=	TEXT
blNull=	'\N' for TEXT files or unquoted empty value for CSV files
blQuote=	" (double quotation mark)
blWriteBuff=	10000
bulkLoad=	FALSE
bulkUnload=	FALSE
catalog=	Active caslib
charMultiplier=	1.0
client_encoding=	Shell encoding
conOpts=	none
database=	none

Data Connector Option	Default Value
dbCreateTableOpts=	none
dbType=	none
insertBuff=	Calculated automatically based on row size
invalidDate=	WARN when loading multiple records ERROR when loading records individually
numReadBuffers=	4
numReadNodes=	0 (use all nodes)
numWriteNodes=	0 (use all nodes)
password=	none
port=	5432
ptg_dsn=	none
readBuff=	Calculated automatically based on row size
schema=	none
server=	none
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "postgres" for PostgreSQL data.
sslMode=	prefer
useNarrowCharacterTypes=	FALSE
userName=	none

Authenticate to Microsoft Azure with Single Sign-on

Single sign-on for the SAS Viya platform uses Microsoft Entra ID (formerly Azure Active Directory) to control authentication. This requires an administrator to configure Microsoft Entra ID as an OpenID Connect provider. For more information, see "Scenario: OIDC with Microsoft Entra ID" in *SAS Viya Platform: Authentication*.

To use single sign-on, establish a connection to PostgreSQL that uses the default authentication scope for Microsoft Azure. To use a different authentication scope, you can specify the AUTHSCOPE= option. Here is a sample CASLIB statement that can connect to a PostgreSQL instance on Microsoft Azure:

```
caslib pgres datasource=(srctype='postgres',
                         server="mydataenvironment.postgres.database.azure.com",
                         database=mypgdata,
                         schema=myschema);
```

When authenticating with single sign-on, do not specify the PASSWORD= or AUTHENTICATIONDOMAIN= options. These options disable single sign-on. The CONOPTS= option also disables single sign-on unless you specify the AUTHSCOPE= option as well.

Bulk Loading and Bulk Unloading Data between PostgreSQL and CAS

Starting in 2025.03, bulk loading and bulk unloading data between PostgreSQL and CAS use the COPY API. Using the COPY API can improve performance when you transfer large amounts of data. To enable bulk loading, set bulkLoad=TRUE; to enable bulk unloading, set bulkUnload=TRUE.

The following data connector options can be used to configure bulk loading and unloading data between PostgreSQL and CAS:

- blDelimiter=
- blEscape=
- blFormat=
- blNull=
- blQuote=
- blWriteBuff=

Note: The blWriteBuff= option is valid only for bulk loading data from CAS to PostgreSQL.

Bulk Loading Date and Time Values to PostgreSQL

When you are bulk loading data that includes date, time, or datetime values to PostgreSQL, be sure to assign a datetime format to your data in CAS. This is important because date, time, and datetime values are stored in SAS as DOUBLE values. When you bulk load data to PostgreSQL, you are using a PostgreSQL utility to move the data, and PostgreSQL does not convert the DOUBLE values to datetime values automatically. Therefore, when you want to bulk load these values from CAS to PostgreSQL, first assign a datetime format to those values. For more information, see "Working with SAS Dates and Times" in *SAS Formats and Informats: Reference*.

Here is an example that shows how to assign a format to data in CAS so that it is written in the correct format to PostgreSQL. You load data into CAS in a DATA step. In the INPUT statement, you supply the informat (time16.6) so that CAS uses

the values as time values. You must also assign the format to the d_timefmt variable in the FORMAT statement. This ensures that your data is written correctly when you bulk load it to PostgreSQL.

Note: In addition to using the dbType= data connector option to assign the time data type to the output, you must also format the data that is passed to the PostgreSQL bulk-load utility. Otherwise, PostgreSQL does not correctly convert an unformatted SAS DOUBLE value to a PostgreSQL time value.

```
data caslib.timedata;
   set caslib.rawdata;
   input id d_timefmt : time16.6;
   format d_timefmt time16.6;
run;

proc casutil;
   save casdata="timedata" incaslib="caslib"
        outcaslib="datalib" casout="PG_time"
        options={bulkload=true, dbtype="d_timefmt='time'"}
        replace;
quit;
```

Case Sensitivity with PostgreSQL

All quoted values are passed to the PostgreSQL database exactly as you type them. This means that you must specify all values, such as table names or ID values, using the same capitalization that is used in the PostgreSQL database.

Data Types for PostgreSQL

Supported PostgreSQL Data Types

The following table shows the data types that can be loaded from PostgreSQL into CAS. This table also shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.38 *Supported PostgreSQL Data Types*

PostgreSQL Data Type	CAS Data Type
Character	
CHARACTER(*n*)	CHAR
CHARACTER VARYING(*n*)	VARCHAR
TEXT	VARCHAR
Numeric	

PostgreSQL Data Type	CAS Data Type
BIGINT	INT64
BIGSERIAL	INT64
DECIMAL(*p*, *s*)	DOUBLE
DOUBLE PRECISION	DOUBLE
INTEGER	INT32
NUMERIC(*p*, *s*)	DOUBLE
REAL	DOUBLE
SERIAL	INT32
SMALLINT	INT32
SMALLSERIAL	INT32
Date and Time	
DATE	DOUBLE (formatted as DATE*w.*)
INTERVAL [fields] (*p*)	VARCHAR
TIME(*p*) (without time zone)	DOUBLE (formatted as TIME*w.d*)
TIMESTAMP(*p*)	DOUBLE (formatted as DATETIME*w.d*)
Network Address	
CIDR	VARCHAR
INET	VARCHAR
MACADDR	VARCHAR
MACADDR8	VARCHAR
Other	

PostgreSQL Data Type	CAS Data Type
BIT(1)	VARCHAR
BOOLEAN	VARCHAR
JSON, JSONB	VARCHAR
MONEY	DOUBLE
UUID	CHAR

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

PostgreSQL Specifics for Character Data

The PostgreSQL data connector can load or save (write) CHAR and VARCHAR data in PostgreSQL that is up to 65535 characters. As long as all processing of this data occurs in CAS, the data is not truncated. However, be careful when using procedures such as PROC SQL. Data that is written using PROC SQL might be truncated to 32767 characters. Similarly, data that is modified in a DATA step that is not running in CAS might be truncated. For more information, see "VARCHAR Data and the CAS Server".

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the

CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Add a PostgreSQL Database as a Data Source for SAS Cloud Analytic Services

Use the CASLIB statement to establish a connection between your PostgreSQL source data and a caslib, PostgreSQLcaslib. All of the options supplied in this example are required in the CASLIB statement, except the SCHEMA= option.

In this example, the PostgreSQL data is stored at the location designated by the SERVER= and DATABASE= options.

```
caslib PostgreSQLcaslib desc='PostgreSQL Caslib'
     dataSource=(srctype='postgres',
                 server='PGserver'
                 username='user1',
                 password='myPwd',
                 database='PGdatabase',
                 schema='myschema');
```

Example 2: Load PostgreSQL Data into SAS Cloud Analytic Services Using PROC CASUTIL

```
proc casutil;
   list files incaslib="PostgreSQLcaslib";          1
   load casdata="myPGdata" incaslib="PostgreSQLcaslib"
outcaslib="casuser"
        casout="class_from_PostgreSQLcaslib";       2
   list files incaslib="casuser";                   3
   contents casdata="%upcase(class_from_PostgreSQLcaslib)"
incaslib="casuser";
                                                    4
quit;
```

1 List the tables in PostgreSQLcaslib before loading your data.

2 Load the table myPGdata from PostgreSQLcaslib into caslib Casuser. Call the new table class_from_PostgreSQLcaslib.

Note: You must specify PostgreSQL table names using the capitalization that is used in the database.

3 List the tables in Casuser to see the newly created table, class_from_PostgreSQLcaslib, that you loaded.

4 List information about the newly loaded table, including column names, data types, and so on.

Salesforce Data Connector

Enables you to transfer data between Salesforce and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

Note: All users with SAS/ACCESS Interface to Salesforce can use SAS Data Connector to Salesforce.

See: Salesforce LIBNAME statement

Examples: Establish a connection between your Salesforce database and SAS Cloud Analytic Services.

```
caslib mycaslib desc='Salesforce Caslib'
     dataSource=(srctype='sforce'
                 username='user1'
                 password='myPwd');
```

Override the user and password values.

```
proc casutil;
   load casdata="sfexamp" casout="mySFdata" incaslib="mycaslib"
      outcaslib="casuser"
      dataSourceOptions=(
         username='user5'
         password='myPwd');
quit;
```

Details

Data Connector Options for Salesforce

Data connector options are used in the context of different statements that connect your data in Salesforce with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.39 *Data Connector Options for Salesforce*

Data Connector Option	Default Value
authenticationDomain=	none
authEndpoint=	none
bulk_api=	NO

Data Connector Option	Default Value
bulk_refresh=	1000 (1 second)
bulk_timeout=	90000 (90 seconds)
charMultiplier=	1.0
conOpts=	none
dbType=	none
mutual_auth=	NO
numReadBuffers=	4
numReadNodes=	1
numWriteNodes=	1
password=	none
proxy_host=	none
proxy_pass=	none
proxy_port=	8080
proxy_user=	none
readBuff=	Calculated automatically based on row size
show_metadata=	NO
show_recycled=	NO
sliceExpressions=	none
srcType=	none Use "sforce" for Salesforce data.
useNarrowCharacterTypes=	FALSE
use_native_names=	NO
userName=	none

Supported Salesforce Data Types

The table below shows the data types that can be loaded from Salesforce into CAS. It also shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.40 *Supported Salesforce Data Types*

Salesforce Data Type	CAS Data Type
Character	
ADDRESS	VARCHAR
ANYTYPE	VARCHAR
BASE64	VARCHAR
COMBOBOX	VARCHAR
DATACATEGORYGROUPREFERENCE	VARCHAR
EMAIL	VARCHAR
ENCRYPTEDSTRING	VARCHAR
ID	VARCHAR
JUNCTIONIDLIST	VARCHAR
LOCATION	VARCHAR
MASTERRECORD	VARCHAR
MULTIPICKLIST	VARCHAR
PHONE	VARCHAR
PICKLIST	VARCHAR
REFERENCE	VARCHAR
STRING	VARCHAR
TEXTAREA	VARCHAR
URL	VARCHAR
Numeric	

Salesforce Data Type	CAS Data Type
BOOLEAN	DOUBLE
BYTE	DOUBLE
CURRENCY	DOUBLE
DOUBLE	DOUBLE
INT	DOUBLE
PERCENT	DOUBLE
Date and Time	
TIME	DOUBLE (formatted as TIME*w.d*
TIMESTAMP	DOUBLE (formatted as DATETIME*w.d*)

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a

DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Add a Salesforce Database as a Data Source for SAS Cloud Analytic Services

Use the CASLIB statement to establish a connection between your Salesforce source data and a caslib, Salesforcecaslib. All options except the PASSWORD= option in this example are required in the CASLIB statement.

```
caslib mycaslib desc='Salesforce Caslib'
     dataSource=(srctype='sforce',
                 username='user1',
                 password='myPwd');
```

Example 2: Load Salesforce Data into SAS Cloud Analytic Services Using PROC CASUTIL

```
proc casutil;
   list files incaslib="mycaslib";                                    1
   load casdata="myDBdata" incaslib="mycaslib" outcaslib="casuser"
        casout="class_from_mycaslib";                                 2
   list files incaslib="casuser";                                     3
   contents casdata="class_from_mycaslib" incaslib="casuser";         4
quit;
```

1 List the tables in Mycaslib before loading your data.

2 Load the table myDBdata from Mycaslib into caslib Casuser. Call the new table class_from_mycaslib.

3 List the tables in Casuser to see the newly created table, class_from_mycaslib, that you loaded.

4 List information about the newly loaded table, including column names, data types, and so on.

SAP HANA Data Connector

Enables you to transfer data between SAP HANA and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

Note: All users with SAS/ACCESS Interface to SAP HANA can use SAS Data Connector to SAP HANA.

See: SAP HANA LIBNAME statement

Examples: Load an SAP HANA data source and add a caslib to it.

```
caslib hacaslib sessref=mysess
   datasource=(srctype='hana',
               server='hana server',
               username='user1',
               password='myPwd',
               tableType='column'
               schema='hana schema'
               instance='00');
```

Load SAP HANA source data using PROC CASUTIL.

```
proc casutil
      incaslib='hacaslib'
      sessref=mysess;
   list files;
   run;
quit;
```

Details

Data Connector Options for SAP HANA

Data connector options are used in the context of different statements that connect your data in SAP HANA with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.41 *Data Connector Options for SAP HANA*

Data Connector Option	Default Value
authenticationDomain=	none
charMultiplier=	1.0
conOpts=	none
dbCreateTableOpts=	none
dbType=	none
insertBuff=	Calculated automatically based on row size
instance=	00
numReadBuffers=	4
numReadNodes=	1

Data Connector Option	Default Value
numWriteNodes=	1
password=	none
port=	30015
readBuff=	Calculated automatically based on row size
schema=	none
server=	none
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "hana" for SAP HANA data.
tableType=	ROW
userName=	none

Supported SAP HANA Data Types

The SAP HANA data connector supports these data types for loading data from SAP HANA into CAS. This table shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.42 *Supported SAP HANA Data Types*

SAP HANA Data Type	CAS Data Type
Character	
ALPHANUM	VARCHAR
CHAR	CHAR
NCHAR	CHAR
NVARCHAR	VARCHAR
SHORTTEXT	VARCHAR

SAP HANA Data Type	CAS Data Type
TEXT	VARCHAR
VARCHAR	VARCHAR
Binary and Others	
VARBINARY	VARCHAR
BLOB	VARCHAR
CLOB	VARCHAR
NCLOB	VARCHAR
BINTEXT	VARCHAR
Numeric	
BOOLEAN	INT32
BIGINT (INT64)	INT64
DECIMAL(*p*, *s*) or DEC(*p*, *s*)	DOUBLE
DOUBLE	DOUBLE
FLOAT(*n*	DOUBLE
INTEGER	INT32
REAL	DOUBLE
SMALLDECIMAL	
SMALLINT (INT32)	INT32
TINYINT (INT32)	
Date and Time	
DATE	DOUBLE (formatted as DATE)
SECONDDATE	DOUBLE (formatted as DATETIME)
TIME	DOUBLE (formatted as TIME)

SAP HANA Data Type	CAS Data Type
TIMESTAMP	DOUBLE (formatted as DATETIME)

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Example: Specify SAP HANA as a Data Source for a Caslib in SAS Cloud Analytic Services

Use the CASLIB statement to initialize the data source and add the caslib for SAP HANA. No connection is made to the data source until a statement that accesses the data is called. The data is read into the caslib hacaslib.

```
caslib hacaslib sessref=mysess desc="hana caslib"
   dataSource=(srctype="hana",
               server="hanaServer",
               username="myuser1",
               password="myPwd",
               tableType="column",
```

```
schema="<Hana_schema_name>",
instance="00");
```

SAP IQ Data Connector

Enables you to transfer data between a SAP IQ data source and CAS.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, SAVE statements

See: SAP IQ LIBNAME statement connection

Example: Establish a connection to SAP IQ data.

```
caslib sapiqcaslib desc="SAPIQtoCaslib"
     datasource=(srctype="sapiq"
                 port=1111
                 server="DBMS-server-on-host"
                 host="SAPIQ-host"
                 database="database-name"
                 username="user1"
                 password="password1"
                 );
```

Details

Data Connector Options for SAP IQ

Data connector options are used in the context of different statements that connect your data in an SAP IQ compatible data source with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.43 *Data Connector Options for SAP IQ Data Sources*

Data Connector Option	Default Value
authenticationDomain=	none
bulkLoad=	FALSE
bulkUnload=	FALSE
catalog=	Active caslib
charMultiplier=	1.0

Data Connector Option	Default Value
clientDir=	none
compress=	FALSE
conOpts=	none
database=	none
dbCreateTableOpts=	none
deleteDataFiles=	TRUE
host=	none
insertBuff=	Calculated automatically based on row size
numReadBuffers=	4
numReadNodes=	1
numWriteNodes=	1
password=	none
port=	none
readBuff=	Calculated automatically based on row size
schema=	none
server=	none
serverDir=	none
siq_dsn=	none
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "sapiq" for SAP IQ data sources.
useNarrowCharacterTypes=	TRUE
usePipe=	FALSE

Data Connector Option	Default Value
userName=	none

Bulk Loading and Unloading with the SAP IQ Data Connector

Bulk loading is the fastest way to move data from CAS to your SAP IQ data source. To use the bulk-load facility to transfer data to SAP IQ, set bulkLoad=TRUE. Similarly, bulk unloading is the fastest way to move data from SAP IQ to CAS. To enable bulk unloading, set bulkUnload=TRUE.

These data connector options are also related to bulk loading and bulk unloading:

- clientDir=
- compress=

  Note: The compress= option applies to bulk loading only.

- deleteDataFiles=
- serverDir=
- usePipe=

  Note: The usePipe= option applies to bulk loading only.

Supported SAP IQ Data Types

The following table lists the supported data types that the SAP IQ data connector can load from SAP IQ into SAS Cloud Analytic Services. This table also shows the resulting data type in CAS.

Note: Be aware that when performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that are read are rounded to meet this condition. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data.

Table 7.44 *Data Type Conversions When Loading Data from SAP IQ into CAS*

SAP IQ Data Type	CAS Data Type
	Character
BINARY	CHAR
CHAR	CHAR or VARCHAR[1]

SAP IQ Data Type	CAS Data Type
LONGVARBINARY	VARCHAR
LONGVARCHAR	VARCHAR
TEXT	VARCHAR
UNIQUEIDENTIFIER	CHAR
UNIQUEIDENTIFIERSTR	VARCHAR
VARBINARY	VARCHAR
VARCHAR	VARCHAR
	Numeric
BIGINT	BIGINT (INT64)
BIT	INTEGER (INT32)
DECIMAL	DOUBLE
DOUBLE	DOUBLE
FLOAT	DOUBLE
INT	INTEGER (INT32)
INTEGER	INTEGER (INT32)
NUMERIC	DOUBLE
REAL	DOUBLE
SMALLINT	INTEGER (INT32)
TINYINT	INTEGER (INT32)
	Date and Time
DATE	DOUBLE (formatted as DATE)
DATETIME	DOUBLE (formatted as DATETIME)
SMALLDATETIME	DOUBLE (formatted as DATETIME)

SAP IQ Data Type	CAS Data Type
TIME	DOUBLE (formatted as TIME)
TIMESTAMP	DOUBLE (formatted as DATETIME)

1 SAP IQ CHAR values are loaded into CAS as CHAR values if the SAP IQ DBMS option ODBC_DISTINGUISH_CHAR_AND_VARCHAR is turned on. Otherwise, SAP IQ CHAR values are loaded into CAS as VARCHAR values.

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Working with CHAR Values in SAP IQ

To load CHAR values from SAP IQ into CAS as CHAR values, ensure that you set ODBC_DISTINGUISH_CHAR_AND_VARCHAR=ON. Otherwise, CHAR values are loaded into CAS as VARCHAR values. The ODBC_DISTINGUISH_CHAR_AND_VARCHAR option is a SAP IQ DBMS option. For more information, see your SAP IQ documentation.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Establish a Connection between a SAP IQ Data Source and a Caslib

Use the CASLIB statement to add a caslib that connects between your SAP IQ data source and CAS.

In this example, GLOBAL specifies that the data source is potentially available to all sessions. You might not have access to the GLOBAL option if you are not an administrator. For more information, see "CASLIB Statement" on page 745.

```
caslib sapiqcaslib desc="SAPIQtoCaslib"
     datasource=(srctype="sapiq"
                 host='sapiq-host'
                 server="DBMS-server-on-host"
                 port=1111
                 database="database-name"
                 username="user1"
                 password="password1"
                 ) global;
```

Example 2: PROC CASUTIL: Load Data from an External Database into SAS Cloud Analytic Services

```
proc casutil;
  list files incaslib="sapiqcaslib";       /* 1 */
  load casdata="myDBdata" incaslib="sapiqcaslib" outcaslib="casuser"
       casout="class_from_sapiqcaslib";   /* 2 */
  list tables incaslib="casuser";         /* 3 */
  contents casdata="class_from_sapiqcaslib" incaslib="casuser";
                                          /* 4 */
quit;
```

1 List the files in sapiqcaslib before loading your data.

2 Load the table myDBdata from an external database into memory in caslib Casuser. Call the new table class_from_sapiqcaslib.

3 List the tables in casuser to see the newly created table, class_from_sapiqcaslib, that you loaded.

4 List information about the newly loaded table, including column names, data types, and so on.

SAS Data Sets Data Connector

Enables you to load data from SAS data sets into CAS.

Applies to: CAS procedure, CASUTIL procedure

Restrictions: Although this data connector supports cross-environment data access (CEDA), it does not support DELETE operations with a CEDA data set. Therefore, if CEDA processing is invoked, you must use the DATASETS procedure to delete a SAS data set. See "Cross-Environment Data Access (CEDA)" on page 308.

Data sets with names that are greater than 32 bytes in UTF8 encoding cannot be accessed.

Requirement: Table names must be specified only in lowercase letters.

Example: Load a password-protected SAS data set into SAS Cloud Analytic Services.

```
proc casutil;
   load casdata="salary.sas7bdat" casout="salary"
      importoptions=(filetype="basesas" password="myPwd");
run;
```

Details

Data Connector Options for SAS Data Sets

Data connector options are used in the context of different statements that connect your data in SAS data sets with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.45 *Data Connector Options for SAS Data Sets*

Data Connector Option	Default Value
alter=	none
charMultiplier=	1.0
compress=	NO
dataTransferMode=	AUTO
encoding=	UTF-8
encrypt=	NO
encryptionPassword=	none
fileType=	BASESAS
password=	none
pointObs=	none

Data Connector Option	Default Value
read=	none
reuse=	NO
varcharConversion=	none
write=	none

Access SAS Data Sets

To access SAS data sets, you must specify a DATASOURCE= argument when you add a caslib with `srctype="PATH"` as the specified value. You can use the SAS Data Connector to SAS Data Sets to access any SAS data set.

For large data sets in particular, if SAS Cloud Analytic Services can access the SAS data set, you can use the CASUTIL procedure with the LOAD CASDATA= syntax.

If you can access the SAS data set with the LIBNAME statement, you can use the CASUTIL procedure with the LOAD DATA= syntax to load the data into SAS Cloud Analytic Services. For an example, see sashelp.iris in load a client-side file on page 8. See the CASUTIL procedure for details about the LOAD CASDATA and LOAD DATA syntax.

Supported SAS Data Sets Data Types

The SAS data sets data connector supports these data types for loading data from SAS data sets into SAS Cloud Analytic Services. This table shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.46 *Supported SAS Data Sets Data Types*

SAS Data Sets Data Type	CAS Data Type
CHAR	CHAR, VARCHAR
DOUBLE	DOUBLE

Cross-Environment Data Access (CEDA)

Although this data connector supports cross-environment data access (CEDA), it does not support DELETE operations with CEDA processing. Therefore, if CEDA processing is invoked, you must use the DATASETS procedure to delete a SAS data set.

CEDA enables a data set that is created on one computer architecture to be read by CAS on the same or different computer architecture. For more information, see "Cross-Environment Data Access" in *SAS Programmer's Guide: Essentials*.

Here are SAS log messages that are specific to CAS. To resolve a truncation error, see "charMultiplier= Data Connector Option" on page 443.

```
ERROR: BASE Data Connector, Cross Environment Data Access (CEDA) translation
failed. Some character data was lost during transcoding while loading the table.
Truncation occurred during transcoding. The charMultiplier= import option may
prevent the error.

ERROR: BASE Data Connector, variable name variable-name exceeds byte limit.

ERROR: BASE Data Connector, Cross Environment Data Access (CEDA) translation
failed. Some character data was lost during transcoding while loading the table.
Unsupported character error occurred during transcoding.

ERROR: BASE Data Connector, variable name variable-name cannot be transcoded to
file encoding.
```

Examples

Example 1: Access a SAS Data Set Using the CASLIB Statement and the CASUTIL Procedure

```
/* Replace sasdata and the path with values for your site. */
libname sasdata '/data';

proc contents data=sasdata.customer;
run;

/* Start a CAS session. */
cas casauto;

/* Assign a Path caslib to the same location as the SAS library. */
caslib sasdata datasource=(srctype="path") path="/data";

/* Use the CASUTIL procedure with the LOAD CASDATA= syntax */
/* to have the server load the SAS data set from disk.     */
proc casutil;
  load casdata="customers.sas7bdat" casout="customers";
quit;

/* Use the CAS LIBNAME statement connection to print       */
/* the in-memory copy of the customers data. */
libname mycas cas sessref=casauto;

proc print data=mycas.customers;
run;
```

For information about the SAS language elements used in this example, see this list.

- "CASLIB Statement" on page 745
- Chapter 3, "CASUTIL Procedure," on page 57
- "LIBNAME Statement: CAS Engine" on page 765

Example 2: Access a SAS Data Set Using the CAS Procedure and Actions

```
/* Start a CAS session. */
cas casauto;

proc cas;
/* Add a PATH type CASLIB called sasdata and      */
/* point it to where the SAS data set is located. */
  table.addcaslib /
    caslib="sasdata"
    datasource={srctype="path"}
    path="/data";
run;

/* Use the loadTable action to load the SAS data set from disk. */
  table.loadTable /
    path="customers.sas7bdat"
    importOptions={fileType="basesas"}
    casout={name="customers", replace=True};
run;

/* Use the fetch action to print the first five records */
/* from the in-memory copy of the customers data.       */
  table.fetch / table="customers" to=5;
run;
```

For information about the CAS procedure and the actions used in this example, see this list.

- "CAS Procedure" in *SAS Cloud Analytic Services: CASL Reference*
- "Add caslib" in *SAS Viya Platform: System Programming Guide*
- "Load table" in *SAS Viya Platform: System Programming Guide*
- "Fetch rows" in *SAS Viya Platform: System Programming Guide*

SAS Federation Server Data Connector

Enables you to transfer data between a SAS Federation Server data source and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

Note: All users with SAS Federation Server can use SAS Data Connector to SAS Federation Server.

Example: Establish a connection to SAS Federation Server data. (Note: The GLOBAL option is restricted to administrators.)

```
caslib fedsvrcaslib desc="SQLviaFedSvrtoCaslib"
   datasource=(srctype="fedsvr"
               server="hostname"
               port="port-value"
               username="myusr1"
```

```
password="mypwd1"
catalog="mycatalog"
schema="myschema"
dsn="ora_dsn"
remoteDriverOptions="driver-options
global);
```

Details

About the SAS Federation Server Data Connector

Before you can use the SAS Federation Server data connector, you must first create a SAS Federation Server caslib. Here are the steps:

1 Assume the Superuser role when prompted during sign-in to SAS Environment Manager. For details, see "Assume the Superuser Role" on the "Servers Page" in *SAS Environment Manager: User's Guide*.

2 Select the **Data Sources** tab, and select the server from the list of available CAS servers. Click the connect icon to open the Connection Settings window, where you can enter caslib information. Enter FEDSVR as the caslib name. From the **Advanced** window, enter the appropriate connection information for each parameter name/value pair. For details, see "Data Sources Tab: Connect to External Data Sources" and "Connecting to Databases" in *SAS Data Hub: User's Guide*.

Data Connector Options for SAS Federation Server

Data connector options are used in the context of different statements that connect your data in a SAS Federation Server data source with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.47 *Data Connector Options for SAS Federation Server*

Data Connector Option	Default Value
authenticationDomain=	none
catalog=	Active caslib
charMultiplier=	1.0
conOpts=	none
dbCreateTableOpts=	none
dsn= on page 484	none

Data Connector Option	Default Value
insertBuff=	Calculated automatically based on row size
numReadBuffers=	4
numReadNodes=	1
numWriteNodes=	1
password=	none
port=	none
readBuff=	Calculated automatically based on row size
remoteDriverOptions=	none
schema=	none
server=	none
sliceColumn=	none
sliceExpressions=	none
useNarrowCharacterTypes=	FALSE
userName=	none

Supported SAS Federation Server Data Types

This table lists the supported data types that the SAS Federation Server data connector can load from ODBC into SAS Cloud Analytic Services and shows the resulting data type in CAS.

Note: SAS Cloud Analytic Services

Table 7.48 *Data Type Conversions When Loading Data from SAS Federation Server into CAS*

SAS Federation Server Data Type	CAS Data Type
	Character
CHAR(*n*)	CHAR

SAS Federation Server Data Type	CAS Data Type
NCHAR	CHAR
NVARCHAR	VARCHAR
VARBINARY(*n*)	VARCHAR
VARCHAR	VARCHAR
Numeric	
BIGINT	DOUBLE (formatted as 20)[1]
DOUBLE	DOUBLE
FLOAT(*p*)	DOUBLE
INTEGER	DOUBLE
REAL	DOUBLE
SMALLINT	DOUBLE
TINYINT	DOUBLE
Date and Time	
DATE	DOUBLE (formatted as DATE9.)
TIME(*p*)	DOUBLE (formatted as TIME8.)
TIMESTAMP(*p*)	DOUBLE (formatted as DATETIME19.2.)

1 The range for valid SAS date values are from 1582-01-01 to 9999-12-31. Dates outside that date range are not supported and are treated as invalid dates.

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Establish a Connection between a Microsoft SQL Database and a Caslib through SAS Federation Server

Use the CASLIB statement to add a Microsoft SQL database as a data source for SAS Cloud Analytic Services using a SAS Federation Server connection. For connections to Microsoft SQL, specify CATALOG="*".

In this example, GLOBAL specifies that the data source is potentially available to all sessions. You might not have access to the GLOBAL option if you are not an administrator. For more information, see "CASLIB Statement" on page 745.

```
caslib fedsvrcaslib desc="SQLviaFedSvrtoCaslib"
   datasource=(srctype="fedsvr"
               server="hostname"
               port="port-value"
               username="myusr1"
               password="mypwd1"
               catalog="mycatalog"
               schema="myschema"
               dsn="dbdsn"
               remoteDriverOptions="REMFS-driver-options"
               global);
```

Example 2: PROC CASUTIL: Load Data from an External Database into SAS Cloud Analytic Services

```
proc casutil;
```

```
   list files incaslib="fedsvrcaslib";     /* 1 */
   load casdata="myDBdata" incaslib="fedsvrcaslib" outcaslib="casuser"
        casout="class_from_fedsvrcaslib";  /* 2 */
   list tables incaslib="casuser";         /* 3 */
   contents casdata="class_from_fedsvrcaslib" incaslib="casuser";
                                           /* 4 */
quit;
```

1 List the files in fedsvrcaslib before loading your data.

2 Load the table myDBdata from an external database into memory in caslib Casuser. Call the new table class_from_fedsvrcaslib.

3 List the tables in casuser to see the newly created table, class_from_fedsvrcaslib, that you loaded.

4 List information about the newly loaded table, including column names, data types, and so on.

SAS SpeedyStore Data Connector

Enables you to transfer data between SingleStore and CAS, using SAS SpeedyStore.

Valid in: CASLIB Statement, PROC CASUTIL

Restriction: Do not use this data connector for a SingleStore database that is external to the SAS Viya platform. For more information, see "Two Data Connectors: When to Use SingleStore Standard and When to Use SAS SpeedyStore" on page 317.

Requirement: To use this data connector, you must license and deploy SAS SpeedyStore (formerly SAS with SingleStore). See *SAS SpeedyStore: Administration and Configuration Guide*.

Tip: For the host= option, specify the child aggregator endpoint.

Example: Establish a connection between your SingleStore database and SAS Cloud Analytic Services.

```
caslib singlestorelib
   datasource=(
               srctype='singlestore'
               database='my_database'
               epDatabase='metadata_database'
               host='svc-sas-singlestore-cluster-dml'
               pass='my_password'
               port='3306'
               user='my_username'
   )
;
```

Details

Data Connector Options for SingleStore

Table 7.49 *Data Connector Options for SingleStore*

Data Connector Options	Default Value
aggregatePushdown=	false
authenticationDomain=	none
backingStore=	"INPLACEPREFERRED"
createMultipassColumn=	The default depends on other conditions. See the option for more information.
createViewSnapshot=	"never"
database=	none
epDatabase=	Value of the database= option.
epProperties=	none
host=	none
multipassMemory=	"cache"
password=	none
port=	3306
saveMultipassColumn=	FALSE
srcType=	none Use "singlestore" for SingleStore data that is integrated in SAS SpeedyStore. See "Two Data Connectors: When to Use SingleStore Standard and When to Use SAS SpeedyStore" on page 317.
ssl_ca=	Inherits from SAS Viya configuration.
udfDatabase=	Value of the database= option.
userName=	none

Authenticate to Microsoft Azure with Single Sign-On

An administrator can enable single sign-on authentication by configuring Microsoft Entra ID (formerly Azure Active Directory) as an OIDC provider. Additional steps are required for SAS SpeedyStore. For the configuration steps, see "Enabling Single Sign-On for Microsoft Azure" in *SAS SpeedyStore: Administration and Configuration Guide*.

In the SingleStore caslib, if you use single sign-on authentication, omit the userName=, password=, and authenticationDomain= options.

If single sign-on authentication is not enabled, then specify the userName= and password= options, or the authenticationDomain= option, to authenticate.

Support for SAS SpeedyStore

SAS SpeedyStore includes two main components for connecting to data:

- The SAS SpeedyStore data connector enables CAS to access tables stored in a SingleStore database that is integrated in SAS. You must license the product SAS SpeedyStore. See *SAS SpeedyStore: Administration and Configuration Guide*. See also "Requirements for SAS SpeedyStore" in *System Requirements for the SAS Viya Platform*.

  A standard data connector is also available to access tables in an external SingleStore database. For more information, see "Two Data Connectors: When to Use SingleStore Standard and When to Use SAS SpeedyStore" on page 317.

- The SingleStore LIBNAME statement connection enables SAS Compute Server to access your SingleStore compatible data source. For more information, see "LIBNAME Statement for SingleStore" in *SAS/ACCESS for Relational Databases: Reference* and the topics that follow this one.

The SAS SpeedyStore data connector offers the following benefits:

- The SAS SpeedyStore data connector provides parallel data transfer between CAS and SingleStore. Parallel data transfer can improve performance for very large tables.
- By using SAS Embedded Process, the SAS SpeedyStore data connector automatically pushes WHERE expressions and computed columns to be processed in SingleStore. Performing these operations in the database reduces the amount of data sent through the network.

Two Data Connectors: When to Use SingleStore Standard and When to Use SAS SpeedyStore

Two data connectors are available for SingleStore:

- The SingleStore *standard* data connector provides the *standard* ability to connect to an external SingleStore database. To use this data connector, specify `srctype='singlestore_standard'` For more information, see "SingleStore Standard Data Connector" on page 333.
- The SAS SpeedyStore data connector is available only if you license the product SAS SpeedyStore. This data connector provides parallel data transfer and in-database features in addition to the standard features. The product SAS

SpeedyStore includes an integrated instance of SingleStore. To use this data connector, specify `srctype='singlestore'`.

Note: To learn whether SAS SpeedyStore is licensed, contact your site administrator.

Here are some guidelines about using the two data connectors:

- If SAS SpeedyStore is not licensed, then use the SingleStore standard data connector.
- If SAS SpeedyStore is licensed but the SingleStore database is external to SAS, then use the SingleStore standard data connector.
- If SAS SpeedyStore is licensed and the SingleStore database is integrated with SAS, then use the SAS SpeedyStore data connector.

Here are some important differences:

- Many options are supported in one data connector and not in the other data connector. Refer to the options that are documented for each data connector.
- Parallel data transfer and other in-database features are provided by the SAS SpeedyStore data connector and not by the SingleStore standard data connector.
- The system requirements for the data sources can differ, which could result in behavior differences. The SingleStore version that is integrated in SAS SpeedyStore could be a later version than the minimum required version for the SAS Viya platform. Note that you cannot move an external SingleStore database to function as an internal database for use with SAS SpeedyStore.

Restrictions for CAS Features

You might need to adjust your code for the following behavior:

- Compression options are ignored when you create data. SingleStore compresses data by default. SingleStore compression is not reported in the output from the tableInfo action.
- Input tables can be columnstore or rowstore. Output tables are always columnstore.
- When you edit a table, a copy of the table is created. The copy is referred to as a *snapshot*. This copy helps to maintain data consistency. The copy contains a multipass column for multipass processing. CAS attempts to create the snapshot copy in the same database as the original table. If the snapshot copy cannot be created, then the operation fails. For example, if the user has insufficient permissions, the copy cannot be made.
- To edit a view, you must set the option createViewSnapshot="onLoad". Under the default setting createViewSnapshot="never", views are streamed without creating a snapshot copy of the view result set, and an attempt to edit a view causes an error. For more information about reading SingleStore views, see "createViewSnapshot= Data Connector Option" on page 459.
- If you create a SingleStore view by using external tools outside SAS, and the view references a table that was created by CAS, some column attributes in the

view are not supported. CAS stores column attributes such as SAS formats and labels and certain data type attributes in the column comments when you save a table to SingleStore. The SingleStore definition of a view does not include column comments. Therefore, if the table that underlies a view was created by CAS, then any column attributes that are stored in column comments are not part of the view definition when the view is loaded into CAS.

- WHERE expressions have several requirements. See "Restrictions for WHERE Processing and Computed Columns" on page 322.
- Missing values are stored as null values in SingleStore. SAS special missing values are not supported, so they are also stored as null values. SingleStore null values are converted to SAS missing values when they are streamed into CAS.
- Table names and column names cannot contain 4-byte characters. This is a limitation of SingleStore. Beginning with 2025.04, 4-byte characters are supported in table labels and column labels. The SingleStore default encoding is `utf8mb4` as of 2024.10. For multibyte data within tables, you must ensure that columns are defined with sufficient length. Truncation can cause partial characters, which results in an error in the log, and data fails to write to the database. For more information, see "Character Set Considerations" in *SAS SpeedyStore: Administration and Configuration Guide*.
- Beginning with 2026.05, the instance of SingleStoreDB that is included with SAS SpeedyStore has a default collation of `utf8mb4_bin`, which causes string comparisons to be case-sensitive. Previous versions have a default collation of `utf8mb4_general_ci`, which is not case-sensitive. New clusters are subject to the new default if you do not set the server collation in your deployment. Existing clusters continue to use the previous collation setting or the previous default if the collation was not set. In previous releases, SAS recommended setting the server collation to `utf8mb4_bin` in the deployment before any data is loaded into SingleStore. If you followed that recommendation, or if you are not using data that was previously loaded into SingleStore, then no changes are necessary. If you did not set the collation and used the default, the difference can lead to unexpected result sets if case distinctions are meaningful. You can change the setting for existing tables. For more information, see "Setting or Modifying the Collation" in *SAS SpeedyStore: Administration and Configuration Guide*.
- GPU processing is not supported.
- For the saveAttrs= parameter in the table.save action and the loadAttrs= parameter in the table.loadTable action, extended attributes tables typically have a naming convention of *table-name*.ATTRS.sashdat. If extended attributes tables are saved to SingleStore, the SASHDAT file extension is not used: *table-name*.ATTRS. In addition, SingleStore table names are limited to 64 characters. Due to the .ATTRS suffix, the table name is limited to 58 characters.
- For more limits, such as column name length, refer to the SingleStore documentation website.
- The table.tableInfo action returns a missing value (.) for the Source Modified Time value.
- The table.tableDetails action returns only the number of rows and the data size. Other details are returned as 0 (zero).

- The groupBy= parameter in the table.save action is not supported.
- The severityEst= parameter in the cdm.cdm action is not supported. However, severity model parameter estimates can be read through the severityStore= parameter.
- The image.flattenImages action is supported when the output table has fewer than 4K columns. FlattenImages generates a table with fewer than 4K columns when the input image *width* * *height* * 3 + 1 is less than or equal to 4096. If this value is greater than 4096, the output table is created and an error is generated when the table is saved to SingleStore.
- The applyRowOrder=TRUE parameter, applicable to the following actions, is not supported:
  - Clustering action set: kClus
  - Decision Tree action set: dtreeTrain, dtreePrune, dtreeSplit, dtreeScore, forestTrain, forestScore, gbtreeTrain, gbtreeScore
  - Factorization Machine action set: factmac
  - Robust Multivariate Outlier action set: mvoutlier
  - Sampling and Partitioning action set: srs, stratified, oversample, kfold
  - Support Vector Machine action set: svmTrain
- The following action sets or actions are not currently supported:
  - audio action set
  - deepLearn action set
  - deepNeural action set
  - deepRnn action set
  - graphSemiSupLearn action set
  - langModel action set
  - ldaTopic action set
  - searchAnalytics action set
  - smartData action set

Improving Performance Using In-Database Processing

When Is In-Database Processing Used?

In-database processing occurs under the following circumstances:

- When you use a WHERE expression or a computed column, the processing is automatically passed to SingleStore:
  - Some WHERE expressions are processed directly by SingleStore. Some are processed by the SAS Embedded Process, running in SingleStore. Using a WHERE expression filters the data and can reduce the amount of data that is streamed to CAS.

  - All computed columns are processed by the SAS Embedded Process, running in SingleStore. Computed columns are specified in the computedVarsProgram= parameter of an action. For more information, see "Using the computedVars and computedVarsProgram Parameters" in *SAS Cloud Analytic Services: CASL Programmer's Guide*.
  - For important restrictions and requirements, see "Restrictions for WHERE Processing and Computed Columns" on page 322.
- For the table.fetch action, if sortBy= is specified for a raw numeric sort, the sort is processed in the database.
- Formatting is performed inside the database by the SAS Embedded Process when the following types of code are processed:
  - a PUT function is in a WHERE clause
  - a PUT function is in a computed columns program
  - in an aggregate pushdown operation, the aggregation groups by formatted values (summary or distinct action with a groupby= parameter)
  - in an aggregate pushdown operation, the aggregation counts distinct formatted values (distinct action with raw=false, which is the default)
- The data connector transfers data in parallel to and from CAS. This parallel transfer of data, also known as *streaming*, does not use the SAS Embedded Process. Therefore, the dataTransferMode= option is not needed.
- The data connector supports in-database model scoring, using PROC SCOREACCEL or CAS actions. For an example, see "Publish, Run, and Delete a Model in SingleStore" in *SAS In-Database Products: User's Guide*.
- SAS Visual Analytics can generate a model that is stored as an analytic store (ASTORE) object. That model can be run from a CAS computed column, which is specified in a computedVarsProgram= parameter. Computed columns are processed in the database. This type of model processing is not the same as in-database model publishing and scoring, which uses DS2 processing.

> **TIP** In-database DATA step and PROC ACCELERATOR are supported. These SAS SpeedyStore features use the SingleStore LIBNAME engine, not the SAS SpeedyStore data connector. For more information, see "DATA Step Acceleration" in *SAS In-Database Products: User's Guide* and "ACCELERATOR Procedure" in *SAS In-Database Products: User's Guide*.

Details about SAS Embedded Processing

Some in-database processing uses the SAS Embedded Process. This feature is a high-performance parallel service process that runs in every node of the SingleStore instance that is included in SAS SpeedyStore. If you license SAS SpeedyStore, installation of the SAS Embedded Process is not required as for other data sources. SingleStore is delivered with the SAS Viya platform order, and SAS Embedded Process is already installed in SingleStore.

When a CAS action is running and the SAS Embedded Process is used to read a table whose data resides in SingleStore, here is how the processing is performed:

- The SAS Embedded Process is invoked using a SingleStore external function (not a SAS function).
- The external function is created at the beginning of the Read operation in the SingleStore database associated with the caslib containing the table.
- The external function is dropped at the end of the Read operation.

When you assign a SingleStore caslib, you can use the epDatabase= option to specify a database for SAS Embedded Process to store metadata. A dedicated database is recommended. See "Creating the Metadata Database for the SAS Embedded Process" in *SAS SpeedyStore: Administration and Configuration Guide*.

Restrictions for WHERE Processing and Computed Columns

When you use a WHERE expression or a computed column, the processing is automatically passed to SingleStore. For more information, see "Improving Performance Using In-Database Processing" on page 320.

Here are some requirements and restrictions:

- When you assign a caslib to the SingleStore data source, you must specify internal service names for the host and port. If you specify external names, then full in-database processing is not supported.
- If a WHERE expression or computed column includes a user-defined format, you must perform the following tasks:

  1. Use the sessionProp.saveFmtLib action to save the format library as a SingleStore database table. The parameter saveForEP=TRUE optimizes performance. Under the default value, TRUE, the format library is saved as a rowstore reference table, and a copy is placed on every leaf. Format libraries tend to be relatively small but heavily used. Set saveForEP=FALSE if the format library is very large.
  2. Set the S2FORMATSEARCH= session option or cas.S2FORMATSEARCH= server option to list one or more database table names that contain the formats.

  For a detailed example, see "Example 2: Publish User-Defined Formats to a SingleStore Database" on page 329. For more information about the action, see "Save format library" in *SAS Viya Platform: System Programming Guide*. For the option syntax, see "S2FORMATSEARCH= Session Option" on page 687, or see "Configuration File Options Reference" in *SAS Viya Platform: SAS Cloud Analytic Services* for the cas.S2FORMATSEARCH= server option. For details about creating and using user-defined formats, see *SAS Cloud Analytic Services: User-Defined Formats*.
- No more than 4,096 columns can be used in a WHERE expression or computed column code that is passed to SingleStore.
- The whereTable= parameter in CAS actions is not supported.

Note: The default value for the saveForEP= parameter is different in the table.save action than in the sessionProp.saveFmtLib action. If you save an analytic store to SingleStore by using the table.save action, the default value saveForEP=FALSE is usually appropriate. The parameter affects all blob tables that are saved to

SingleStore. Analytic stores (astores) are one type of blob table. If you have moderately sized analytic stores, then you might consider setting saveForEP=TRUE. When you set saveForEP=TRUE, a copy of the blob table is placed on every leaf. That behavior is optimal for analytic stores that are used by the SAS Embedded Process. However, if analytic stores are very large, or if you are saving other types of blob tables, then setting saveForEP=TRUE can negatively affect performance.

Improving Performance for Multipass Analytics

Multipass analytics require data to be read more than once. Because the data connector streams data on demand, multiple passes can affect performance. The multipassMemory="cache" option can help this performance issue by reducing data movement. The requested data is temporarily loaded into the CAS disk cache and is released from the cache after the multipass action completes. You can further improve performance if you filter the data that is loaded.

Other choices are available for different use cases. The multipassMemory="stream" setting helps to conserve memory use, and uses a multipass column to improve performance. The backingStore="CASDISKCACHE" option completely avoids streaming, and loads the entire table in the CAS disk cache in the traditional manner. For a comparison of these choices, see "multipassMemory= Data Connector Option" on page 532.

Note that the singlePass= parameter has no effect on multipass behavior. For use with other data connectors, see "Understanding the SinglePass Parameter" in *SAS Viya Platform: System Programming Guide*.

Improving Performance Using SQL Pushdown

SQL pushdown is supported for these actions:

- Aggregate pushdown is supported by the simple.distinct action beginning in 2025.03.
- Aggregate pushdown is supported by the simple.summary action beginning in 2024.11.
- SELECT DISTINCT pushdown is supported by the simple.groupBy action beginning in 2024.09.

Aggregate pushdown enables CAS actions to push down aggregation requests to be processed in the database. The aggregated data is streamed to CAS. In general, performance improves when the aggregation can run in the database, instead of streaming the full data to CAS for aggregation. Here is how to process an aggregation request in the database:

- Either specify the aggregatePushdown=true data connector option or specify sql=true in an action that supports aggregate pushdown. If the sql= parameter is specified in the action, then that setting takes precedence over the aggregatePushdown= data connector option. If sql= is not set, then use aggregatePushdown=true to push down aggregation requests.
- Do not use the groupbyTable parameter in the simple.distinct action. This parameter is not supported for pushdown.

- Do not use any of the following parameters in the simple.summary action: freq=, groupByTable=, repeat=, weight=, whereTable=. These parameters are not supported for pushdown.

SELECT DISTINCT pushdown is not an aggregation request, but it does reduce the result set that is returned to CAS. Here is how to implement SELECT DISTINCT pushdown:

- Specify the aggregatePushdown=true data connector option. (The sql= parameter is not available in simple.groupBy.)
- Do not use any of the following parameters in the simple.groupBy action: aggregator=, freq=, scoreGt=, scoreLt=, weight=. These parameters are not supported for pushdown.

When an aggregation request is pushed down to the database, the following message is written in the SAS log:

```
NOTE: The aggregation is being calculated in the database.
```

For more information, see "aggregatePushdown= Data Connector Option" on page 389.

If a SAS log message indicates that a user-defined function is required, then download the open-aggregates user-defined functions (UDFs). See "udfDatabase= Data Connector Option" on page 594 for more information.

Reading Data Types from SingleStore

The following table shows the data types that can be loaded from SingleStore into CAS. This table also shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.50 *Supported SingleStore Data Types*

SingleStore Data Type	CAS Data Type
	Character
CHAR(*n*)	CHAR(*n*)
LONGTEXT	VARCHAR
MEDIUMTEXT	VARCHAR
TEXT	VARCHAR
TINYTEXT	VARCHAR
VARCHAR	VARCHAR
	Binary

SingleStore Data Type	CAS Data Type
BINARY	BINARY(*n*)
BLOB	VARBINARY
LONGBLOB	VARBINARY
MEDIUMBLOB	VARBINARY
TINYBLOB	VARBINARY
VARBINARY	VARBINARY
	Numeric
BIGINT	INT64
BOOL	INT32
DECIMAL	DOUBLE
DOUBLE	DOUBLE
FLOAT	DOUBLE
INT	INT32
MEDIUMINT	INT32
SMALLINT	INT32
TINYINT	INT32
	Date and Time
DATE	DOUBLE (formatted as DATE*w*. if no format is specified)
DATETIME	DOUBLE (formatted as DATETIME*w.d* if no format is specified)
DATETIME(6)	DOUBLE (formatted as DATETIME*w.d* if no format is specified)
TIME	DOUBLE (formatted as TIME*w.d* if no format is specified)

SingleStore Data Type	CAS Data Type
TIME(6)	DOUBLE (formatted as TIME*w.d* if no format is specified)
TIMESTAMP	DOUBLE (formatted as DATETIME*w.d* if no format is specified)
TIMESTAMP(6)	DOUBLE (formatted as DATETIME*w.d* if no format is specified)
YEAR	INT32
	Other:
BIT	INT64
ENUM	VARCHAR
GEOGRAPHY	ERROR (not supported)
GEOGRAPHYPOINT	CHAR(144)
JSON	VARCHAR
SET	VARCHAR

Writing Data Types to SingleStore

The following table describes how CAS writes data to SingleStore data types.

Table 7.51 *Writing CAS Columns to SingleStore Data Types*

CAS Data Type	SingleStore Data Type
	Binary
BINARY(*n*)	BINARY(*n*)
VARBINARY	LONGBLOB
	Character
CHAR(*n*) (where *n* < 256)	CHAR(*n*)
CHAR(*n*) (where 256 <= *n* < 21845)	VARCHAR(*n*)

CAS Data Type	SingleStore Data Type
CHAR(*n*) (where *n* >= 21845)	LONGTEXT
VARCHAR(*n*) (where *n* < 21845)	VARCHAR(*n*)[1]
VARCHAR(*n*) (where *n* >= 21845)	LONGTEXT
VARCHAR(*)	LONGTEXT
Numeric	
DOUBLE with format *w*. (where *w* <= 6)	INT
DOUBLE with format *w*. (where 7 <= *w* <=17)	BIGINT
DOUBLE with a datetime format[2]	DATETIME(6)[3]
DOUBLE with a date format[2]	DATE
DOUBLE with a time format[2]	TIME(6)[3]
DOUBLE with any other format, or no format	DOUBLE
INT32	INT
INT64	BIGINT

1 When a VARCHAR(*n*) value is updated or appended, if the value has more than *n* characters, the value is truncated to *n* characters.

2 For a list of date and time formats, see "Date and Time" in *SAS Formats and Informats: Reference*.

3 Prior to 2023.12, the column converted to DATETIME(6) or TIME(6) only when the format's decimal (*d*) was nonzero. The column converted to DATE or TIME otherwise.

Here is an example. A table is created in SingleStore by using a previously assigned caslib. In this code, informats are specified in the INPUT statement to properly read the data into the time_col and time_col_dec columns. The corresponding formats are permanently assigned by using the FORMAT statement. A format is necessary for writing a date or time data type to SingleStore. The time_col column is written as a TIME data type. The time_col_dec column is written as a TIME(6) data type, due to the `.2` decimal value in the format `time11.2`.

```
libname mylib cas caslib="mycaslib";
data mylib.timedata;
   input item $ time_col time8. +1 time_col_dec time11.2;
   format time_col time8. time_col_dec time11.2;
   datalines;
apple 01:01:01 01:01:01.01
orange 11:59:59 11:59:59.44
;
```

```
run;
```

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Add a SingleStore Database as a Data Source for SAS Cloud Analytic Services

Use the CASLIB statement to establish a connection between your SingleStore source data and a caslib, singlestorelib.

```
caslib singlestorelib
   datasource=(
     srctype="singlestore"
     database="my_database"
     epDatabase="metadata_database"
     host="svc-sas-singlestore-cluster-dml"
     pass="my_password"
     port="3306"
     user="my_username"
```

```
   )
;
```

Example 2: Publish User-Defined Formats to a SingleStore Database

Before you run this example, submit the CASLIB statement that you create in "Example 1: Add a SingleStore Database as a Data Source for SAS Cloud Analytic Services" on page 328.

The following block of code creates a format library named myfmtlib that contains a format named agefmt. Next, the code creates a sample table from a Sashelp table and saves the table to SingleStore.

```
proc format casfmtlib="myfmtlib" sessref=casauto;
   value agefmt
      low-12 = "group 1"
      13-14 = "group 2"
      15-high = "group 3";
run;

proc casutil;
   load data=sashelp.class
   casout="myclass" replace;
run;
proc casutil;
   save casdata='myclass'
   casout="myclass_save"
   outcaslib="singlestorelib" replace;
run;
```

The next block of code saves the format library to the SingleStore database, because a later step uses the format in a WHERE expression. WHERE expressions are processed in the database, so the format library must be available. The format library is saved as a table named myfmtlib_save in the database that was specified in the singlestorelib caslib. In-database processing must be able to search the format library and find the format. The code uses the S2FORMATSEARCH= session option to specify the table that contains the format library. If the formats need to be shared across sessions, then specify the cas.S2FORMATSEARCH= server option instead of the session option.

```
proc cas;
   savefmtlib / fmtlibname="myfmtlib"
                name="myfmtlib_save"
                caslib="singlestorelib";
run;
quit;

options cassessopts=(s2formatsearch="singlestorelib.myfmtlib_save");

/* Uncomment this code to make the format library available globally */
/* proc cas;                                                         */
/*    assumerole / adminrole='superuser';                            */
/*    configuration.setservopt / s2formatsearch='s2.s2formatname';   */
```

```
/* run;                                                              */
/* quit;                                                             */
```

The next block of code fetches rows that meet a condition in a where= parameter. Then the code uses a WHERE clause in a PRINT procedure. Both uses of a WHERE expression cause the row selection to occur in the database. In-database processing requires the user-defined format agefmt to be saved in the database and made available by using the S2FORMATSEARCH= option.

```
proc casutil;
   load casdata="myclass_save"
   incaslib="singlestorelib"
   casout="myclass_load"
   outcaslib="singlestorelib";
run;

proc cas;
   fetch /
      table={caslib="singlestorelib"
             name="myclass_load"
             where='put(age, agefmt.) = "group 2"'
             }
           sortby={"Name"}
   ;
run;
quit;

libname mycas cas sessref=casauto caslib="singlestorelib";
proc print data=mycas.myclass_load;
   format age agefmt.;
   where age > 14;
run;
```

Output 7.1 *PROC PRINT Output*

Obs	Name	Sex	Age	Height	Weight
1	Philip	M	group 3	72.0	150.0
2	Ronald	M	group 3	67.0	133.0
3	William	M	group 3	66.5	112.0
4	Janet	F	group 3	62.5	112.5
5	Mary	F	group 3	66.5	112.0

Use the following code if you need to delete myclass_save and myfmtlib_save from the database.

```
proc cas;
   deletesource
      source="myclass_save" caslib="singlestorelib" ;
   deletesource
      source="myfmtlib_save" caslib="singlestorelib" ;
run;
quit;
```

For more information about the action, see "Save format library" in *SAS Viya Platform: System Programming Guide*. For the option syntax, see "S2FORMATSEARCH= Session Option" on page 687, or see "Configuration File Options Reference" in *SAS Viya Platform: SAS Cloud Analytic Services* for the cas.S2FORMATSEARCH= server option. For details about creating and using user-defined formats, see *SAS Cloud Analytic Services: User-Defined Formats*.

Example 3: Access Data from a SingleStore Pipeline

This example demonstrates that the data connector supports the SingleStore Pipelines feature. For more information about pipelines, see the SingleStore documentation website.

The user submits the following SQL code outside of SAS. For example, the code could be submitted in SingleStoreDB Studio. The code creates a new table named student_grades in the SingleStore database. Then the code defines a SingleStore pipeline to load data from Azure blob storage into student_grades. When the pipeline is started, the pipeline loads all the data from the storage location into the student_grades table.

```
/* Set the working database. */
use test ;

/* Create the student_grades table. */
create table if not exists student_grades (
      student varchar (20),
      class varchar (20),
      grade double,
      semester double,
      shard key () )
   autostats_cardinality_mode=incremental
autostats_histogram_mode=create
   autostats_sampling=on sql_mode='strict_all_tables' ;

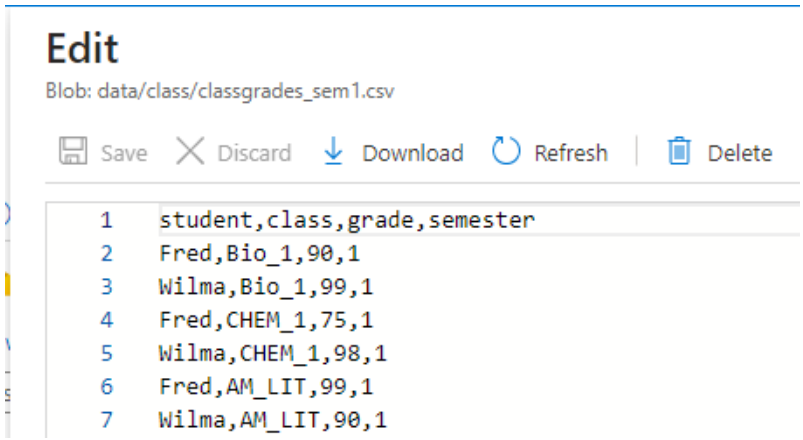
/* Define the pipeline. */
create aggregator pipeline if not exists classgrades_pl
as load data azure 'my_demo/data/class'
/* Replace the ADLS account name and key. */
credentials '{"account_name": "my_account_name", "account_key":
"my_account_key"}'
into table student_grades
fields terminated by ','
(student,class,grade,semester);

/* Start the pipeline. */
start pipeline if not running classgrades_pl ;

/* Stop the pipeline later, after the demo is finished. */
/* stop pipeline if running classgrades_pl; /*
```

The following file is available in the Azure blob storage location that is associated with the classgrades_pl pipeline. The CSV file contains the semester 1 grades for two students.

Output 7.2 *Raw Data for Semester 1 Grades*



In the SAS Viya platform, the user starts the CAS server, assigns a caslib, and submits a FEDSQL procedure query.

```
cas mysession;

caslib s2 desc="My SingleStore Caslib"
   dataSource=(srctype='singlestore',
   database="my_database",
   epdatabase="my_ep_database",
   host="my_host",
   pass="my_password",
   port=my_port,
   user="my_user_ID");

proc casutil incaslib="s2" outcaslib="s2" ;
   load casdata="student_grades" casout="student_grades" promote;
quit;

proc fedsql sessref=mysession ;
   select student, round(avg(grade),.01) as gpa from s2.student_grades
group by student ;
quit ;
```

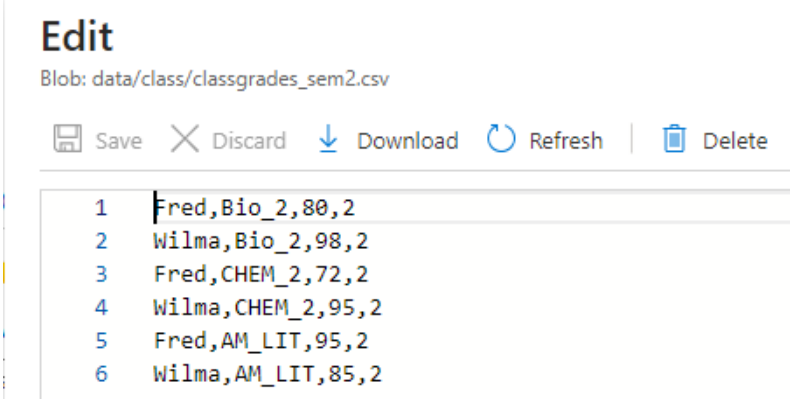
The PROC FEDSQL query of the student_grades table produces the following output.

Output 7.3 *PROC FEDSQL Output of Grade-Point Averages for Semester 1*

student	GPA
Fred	88.000000
Wilma	95.670000

At the end of semester 2, the user adds another file in the Azure blob storage location. The CSV file contains the semester 2 grades for the two students. If the user's SingleStore pipeline is running, it automatically appends the new data in the student_grades table.

Output 7.4 *Raw Data for Semester 2 Grades*



The user runs the same PROC FEDSQL code as for semester 1. Due to the additional data, the output has changed. Both students had lower grades in semester 2, so their grade-point averages are lower, as shown in the following output:

Output 7.5 *PROC FEDSQL Output of Grade-Point Averages for Semester 1 and 2*

student	GPA
Fred	85.170000
Wilma	94.170000

SingleStore Standard Data Connector

Enables you to transfer data between SingleStore and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

Restriction: Do not use the standard data connector for a SingleStore database that is integrated in SAS SpeedyStore (formerly SAS with SingleStore). For more information, see "Two Data Connectors: When to Use SingleStore Standard and When to Use SAS SpeedyStore" on page 337.

See: "LIBNAME Statement for SingleStore" in *SAS/ACCESS for Relational Databases: Reference*

SingleStore LIBNAME statement, "Where to Specify Data Connector Options"

Examples: Establish a connection between your SingleStore database and SAS Cloud Analytic Services.

```
caslib mycaslib desc='SingleStore Caslib'
     dataSource=(srctype='singlestore_standard'
                 host='server-name'
                 username='user1'
                 password='myPwd'
                 database="database-name");
```

Override the user and password values.

```
proc casutil;
   load casdata="examp" casout="myDBdata" incaslib='mycaslib'
       outcaslib='casuser'
       dataSourceOptions=(
         username='user5'
         password='myPwd');
quit;
```

Details

Data Connector Options for SingleStore Standard

Data connector options are used in the context of different statements that connect your data in SingleStore with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.52 *Data Connector Options for Data Sources That Are Compatible with SingleStore Standard*

Data Connector Options	Default Value
append=	FALSE
authenticationDomain=	none
authScope=	`https://storage.azure.com/user_impersonation`
blDefaultDir=	current location for storing temporary data files
bulkLoad=	FALSE
charMultiplier=	1.0
conOpts=	none
database=	none
dbType=	none
enable_cleartextPlugin=	FALSE
host=	none

Data Connector Options	Default Value
numReadBuffers=	4
numReadNodes=	1
numWriteNodes=	1
password=	none
port=	none
readBuff=	Calculated automatically based on row size
schema=	none
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "singlestore_standard" for SingleStore data that is external to SAS. See "Two Data Connectors: When to Use SingleStore Standard and When to Use SAS SpeedyStore" on page 337.
ssl_ca=	none
ssl_cert=	none
ssl_cipher=	none
ssl_key=	none
useNarrowCharacterTypes=	FALSE
userName=	none

Authenticate to Microsoft Azure with Single Sign-On for the SingleStore Standard Data Connector

An administrator can enable single sign-on (SSO) authentication by configuring Microsoft Entra ID (formerly Azure Active Directory) as an OpenID Connect (OIDC) provider. The SingleStore user authenticates with a JSON Web Token (JWT).

Note: These instructions are not appropriate to enable SSO for a connection to the SingleStore server that is integrated in the product SAS SpeedyStore. Do not use the *SingleStore standard data connector* (`srctype='singlestore_standard'`) for an integrated SingleStore server. Instead, use the *SAS SpeedyStore data connector* (`srctype='singlestore'`). For instructions to enable SSO for that data connector, see "Authenticate to Microsoft Azure with Single Sign-On" on page 317. For information about the differences, see "Two Data Connectors: When to Use SingleStore Standard and When to Use SAS SpeedyStore" on page 337.

To enable SSO for the SingleStore standard data connector, which connects to an external SingleStore server, use the following instructions:

- Follow the steps in "Scenario: OIDC with Microsoft Entra ID" in *SAS Viya Platform: Authentication*.

  When you configure OIDC provider properties for Microsoft Entra ID for **sas.logon.oauth.providers**, you must include `profile` in the list of values for the **scopes** configuration field. This ensures the use of user principal name (UPN) for the user name claim.

- In SingleStore, you must set the following two variables:

  ```
  jwks_username_field='upn'
  jwks_endpoint = endpoint-to-retrieve-JWTs-from-Entra-ID
  ```

  Use the Azure portal to learn the value for `jwks_endpoint`. Click **Overview**. On the App registrations Overview page, click **Endpoints**. Copy the URL that appears in the OpenID Connect metadata document field and open it in a new browser tab. Retrieve the value for `jwks_uri` and specify it in the `jwks_endpoint` variable. The value is in a form similar to the following:

  ```
  https://login.microsoftonline.com/entra_tenant_id/discovery/v2.0/keys
  ```

  Submit the following commands to review the SingleStore settings for `jwks_username` and `jwks_endpoint`:

  ```
  show global variables like 'jwks_username%';
  show global variables like 'jwks_endpoint';
  ```

- You can use either LDAP or SCIM for identity management. You cannot use both.

  - If you want to use LDAP, SingleStore provides a command-line tool to synchronize your Lightweight Directory Access Protocol (LDAP) users with SingleStore. For more information, search for `sdb-admin sync-ldap` in the SingleStore documentation website.

  - If you want to use SCIM, then you must specify the SSL_CA= option. The location of the certificate authority (CA) file for the SAS Viya platform is **`/security/trustedcerts.pem`** when Front-door TLS and Full-stack TLS is enabled. For more information, see the SSLCALISTLOC= system option.

To use single sign-on authentication to connect to Microsoft Azure using Microsoft Entra ID, do not specify the userName=, password=, or authenticationDomain= options. When you omit those options, you establish a connection to SingleStore that uses the default authentication scope for Microsoft Azure. To use a different authentication scope, you can specify the authScope= option.

If single sign-on authentication is not enabled, then specify the userName= and password= options, or the authenticationDomain= option, to authenticate.

Two Data Connectors: When to Use SingleStore Standard and When to Use SAS SpeedyStore

Two data connectors are available for SingleStore:

- The SingleStore *standard* data connector provides the *standard* ability to connect to an external SingleStore database. To use this data connector, specify `srctype='singlestore_standard'`
- The SAS SpeedyStore data connector is available only if you license the product SAS SpeedyStore. This data connector provides parallel data transfer and in-database features in addition to the standard features. The product SAS SpeedyStore includes an integrated instance of SingleStore. To use this data connector, specify `srctype='singlestore'`. For more information, see "SAS SpeedyStore Data Connector" on page 315. For more information about the product, see *SAS SpeedyStore: Administration and Configuration Guide*.

  Note: To learn whether SAS SpeedyStore is licensed, contact your site administrator.

Here are some guidelines about using the two data connectors:

- If SAS SpeedyStore is not licensed, then use the SingleStore standard data connector.
- If SAS SpeedyStore is licensed but the SingleStore database is external to SAS, then use the SingleStore standard data connector.
- If SAS SpeedyStore is licensed and the SingleStore database is integrated with SAS, then use the SAS SpeedyStore data connector.

Here are some important differences:

- Many options are supported in one data connector and not in the other data connector. Refer to the options that are documented for each data connector.
- Parallel data transfer and other in-database features are provided by the SAS SpeedyStore data connector and not by the SingleStore standard data connector.
- The system requirements for the data sources can differ, which could result in behavior differences. The SingleStore version that is integrated in SAS SpeedyStore could be a later version than the minimum required version for the SAS Viya platform. Note that you cannot move an external SingleStore database to function as an internal database for use with SAS SpeedyStore.

Bulk Loading to SingleStore

Bulk loading enables you to move large amounts of data from CAS to SingleStore as efficiently as possible. Bulk loading uses a SingleStore LOAD DATA command.

Here are the options that you use when you bulk load data using the SingleStore standard data connector:

- blDefaultDir=
- bulkLoad=

Supported SingleStore Data Types

The following table shows the data types that can be loaded from SingleStore into CAS using the SingleStore standard data connector. This table also shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.53 *Supported SingleStore Data Types*

SingleStore Data Type	CAS Data Type
Character	
CHAR(*n*)	CHAR
VARCHAR(*n*)	VARCHAR
TEXT	VARCHAR
TINYTEXT	VARCHAR
MEDIUMTEXT	VARCHAR
LONGTEXT	VARCHAR
BLOB	VARCHAR
TINYBLOB	VARCHAR
MEDIUMBLOB	VARCHAR
LONGBLOB	VARCHAR
ENUM	CHAR
SET	CHAR
JSON	VARCHAR
Numeric	
BIT	INT32
INT	INT32
TINYINT	INT32

SingleStore Data Type	CAS Data Type
SMALLINT	INT32
MEDIUMINT	INT32
BIGINT	INT64
DECIMAL	DOUBLE
DOUBLE	DOUBLE
FLOAT	DOUBLE
Date and Time	
DATE	DOUBLE (formatted as DATE*w.*)
DATETIME	DOUBLE (formatted as DATETIME*w.d*)
DATETIME(6)	DOUBLE (formatted as DATETIME*w.d*)
TIME	DOUBLE (formatted as TIME*w.d*)
TIME(6)	DOUBLE (formatted as TIME*w.d*)
TIMESTAMP	DOUBLE (formatted as DATETIME*w.d*)
TIMESTAMP(6)	DOUBLE (formatted as DATETIME*w.d*)
YEAR	INT32

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Add a SingleStore Database as a Data Source for SAS Cloud Analytic Services

Use the CASLIB statement to establish a connection between your SingleStore source data and a caslib, SingleStorecaslib. All of the options supplied in this example are required in the CASLIB statement, except the password= option.

In this example, the SingleStore data is stored at the location designated by the host= and database= options.

```
caslib mycaslib desc='SingleStore standard Caslib'
     dataSource=(srctype=singlestore_standard,
                 host='server-name'
                 username='user1',
                 password='myPwd',
                 database='myDatabase');
```

Example 2: Load SingleStore Data into SAS Cloud Analytic Services Using PROC CASUTIL

```
proc casutil;
   list files incaslib="mycaslib";          1
   load casdata="myDBdata" incaslib="mycaslib" outcaslib="casuser"
        casout="class_from_mycaslib";       2
```

```
   list files incaslib="casuser";                3
   contents casdata="class_from_mycaslib" incaslib="casuser";
                                                 4
quit;
```

1 List the tables in Mycaslib before loading your data.

2 Load the table myDBdata from Mycaslib into caslib Casuser. Call the new table class_from_mycaslib.

3 List the tables in Casuser to see the newly created table, class_from_mycaslib, that you loaded.

4 List information about the newly loaded table, including column names, data types, and so on.

Snowflake Data Connector

Enables you to transfer data between Snowflake and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

See: Snowflake LIBNAME statement

Examples: Establish a connection between your Snowflake database and SAS Cloud Analytic Services.

```
caslib snowcaslib desc='Snowflake Caslib'
     dataSource=(srctype='snowflake'
                 server='SFserver'
                 schema='myschema'
                 username='user1'
                 password='myPwd'
                 database="sfdatabase-name");
```

Override the user and password values.

```
proc casutil;
   load casdata="mycas.pgexamp" casout="mySFdata" incaslib="snowcaslib"
      outcaslib="casuser"
      dataSourceOptions=(
         username='user5'
         password='myPwd');
quit;
```

Details

Data Connector Options for Snowflake

Data connector options are used in the context of different statements that connect your data in Snowflake with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about

where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.54 *Data Connector Options for Snowflake*

Data Connector Option	Default Value
accountName=	none
applicationId=	none
authenticationDomain=	none
authenticationScope=	none
awsConfig=	`~/.aws/config`
azureSAS=	none
blNumDataFiles=	2
blNumReadThreads=	none
blOptions=	none
bucket=	none
bulkLoad=	none
bulkUnload=	FALSE
charMultiplier=	1.0
client_encoding=	none
compress=	FALSE
conOpts=	none
credentialsFile=	none
credentialsProfile=	none
database=	none
dbCreateTableOpts=	none
dbType=	none

Data Connector Option	Default Value
defaultDir=	*temporary-file-directory* that is specified by the UTILLOC= system option
deleteDataFiles=	TRUE
delimiter=	bell character (ASCII 0x07)
dnsSuffix=	dfs.core.windows.net
escape=	FALSE
fileSystem=	none
insertBuff=	Calculated automatically based on row size
key=	none
numReadBuffers=	4
numReadNodes=	0 (use all nodes)
numWriteNodes=	0 (use all nodes)
password=	none
profile=	default
queryTag=	none
readBuff=	Calculated automatically based on row size
region=	none
role=	none
s3Config=	none
scanStringColumns=	FALSE
schema=	none
secret=	none
server=	none
sliceColumn=	none

Data Connector Option	Default Value
sliceExpressions=	none
snowflake_dsn=	none
srcType=	none Use "snowflake" for Snowflake data.
timeout=	60
token=	none
useNarrowCharacterTypes=	FALSE
userName=	none
useSSL=	TRUE
warehouse=	none

Authenticate to Microsoft Azure with Microsoft Entra ID

Single sign-on for the SAS Viya platform uses Microsoft Entra ID (formerly Azure Active Directory) to control authentication. This requires an administrator to configure Microsoft Entra ID as an OpenID Connect provider. For more information, see "Scenario: OIDC with Microsoft Entra ID" in *SAS Viya Platform: Authentication*.

To use single sign-on, establish a connection to Snowflake that uses the default authentication scope for Microsoft Azure. To use a different authentication scope, you can specify the AUTHSCOPE= option. Here is a sample CASLIB statement that can connect to a Snowflake instance on Microsoft Azure:

```
caslib db datasource=(srctype='snowflake',
                      snowflake_dsn="Snowflake-datasource-name";
```

When authenticating with single sign-on, do not specify the USERNAME=, PASSWORD=, or AUTHENTICATIONDOMAIN= options. These options disable single sign-on. The CONOPTS= option also disables single sign-on unless you also specify the AUTHSCOPE= option.

Bulk Loading and Bulk Unloading with Microsoft Azure Synapse Analytics

Overview of Bulk Loading to and Bulk Unloading from ADLS2

You can use the Snowflake data connector to bulk load data to Azure Synapse Analytics (SQL DW) and to bulk unload data to SAS Viya. Tables are transferred to and from a Microsoft Azure Data Lake Storage Gen2 (ADLS2) storage location. The

Azure COPY command is used to transfer the data from external storage to a table in Azure Synapse. Supported formats for files that you can bulk load are delimited files (CSV) and compressed delimited files (CSV files compressed with GZIP). You create delimited and compressed delimited files on a CAS node and then bulk load the data to ADLS2 storage via the bulk-loading options in your CASLIB or PROC CASUTIL SAVE statement.

Here are the options that you use when you bulk load data to or bulk unload data from ADLS2.

- accountName=
- applicationId=
- azureSAS=
- bulkLoad= or bulkUnload=
- compress=
- deleteDataFiles=
- delimiter=
- dnsSuffix=
- escape=

  Note: The escape= option applies to bulk loading data only.

- fileSystem=
- timeout=

You can see an example of using bulk-loading options in "Example 3: Bulk Load to ADLS2 Storage".

Requirements for Bulk Loading to and Bulk Unloading from ADLS2

Bulk loading data to and bulk unloading from Azure Synapse Analytics (SQL DW) requires that you specify the Azure tenant ID value with the AZURETENANTID= system option. You must also specify the location of a file that contains authentication information that Microsoft Azure uses. You specify this location with the AZUREAUTHCACHELOC= system option. For more information, see the Requirements section in "Azure Data Lake Storage Data Source" on page 135, AZURETENANTID= system option, or AZUREAUTHCACHELOC system option.

Authentication to ADLS2

The first time that you access an ADLS caslib, SAS writes an error to the log.

```
ERROR: Message from TK: Pending end-user authorization.
ERROR: To sign in, use a web browser to open the page https://microsoft.com/
devicelogin
  and enter the code ###### to authenticate.
```

Log on to the Microsoft web page. A series of windows provide further instruction and result in the required authorization. Subsequent attempts to access data do not require this step. The authorization process creates a file that is accessible to the CAS session for the host account that is used to run the CAS session. This file is created in the location as specified in the AZUREAUTHCACHELOC= system option.

Bulk Loading and Bulk Unloading Data between Amazon S3 and CAS

You can bulk load Snowflake data from CAS to Amazon S3, and you can bulk unload Snowflake data that is staged in Amazon S3 into CAS. To enable bulk loading or bulk unloading, set bulkLoad=TRUE or bulkUnload=TRUE, respectively.

Here are additional options that you can use when you bulk load or bulk unload data between CAS and Amazon S3:

- awsConfig=

  Note: You do not need to specify awsConfig if your AWS configuration file is stored in the default location, `~/.aws/config`.

- bucket=

  Note: The bucket= option is required when you work with data in Amazon S3.

- compress=
- deleteDataFiles=
- delimiter=
- escape=

  Note: The escape= option applies to bulk loading data only.

- region=

  Note: The region= option is required when you work with data in Amazon S3.

- s3Config=

  Note: The s3Config= option is required when you work with data in Amazon S3.

- timeout=

To see an example that shows how to use these options, see "Example 4: Bulk Unload Snowflake Data from Amazon S3 to CAS" on page 350.

Supported Snowflake Data Types

The table below shows the data types that you can load from Snowflake into CAS. It also shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.55 *Supported Snowflake Data Types*

Snowflake Data Type	CAS Data Type
Character	
CHAR(*n*)	VARCHAR
VARCHAR(*n*)	VARCHAR
Numeric	
BIGINT	DOUBLE
BOOLEAN	DOUBLE
DECIMAL(*p*, *s*)	DOUBLE
DOUBLE	DOUBLE
FLOAT	DOUBLE
INTEGER	DOUBLE
NUMERIC(*p*, *s*)	DOUBLE
REAL	DOUBLE
SMALLINT	DOUBLE
TINYINT	DOUBLE
Date and Time	
DATE	DOUBLE (formatted as DATE*w.*)
DATETIME	DOUBLE (formatted as DATETIME*w.d*)
TIME	DOUBLE (formatted as TIME*w.*)

Snowflake Data Type	CAS Data Type
TIMESTAMP	DOUBLE (formatted as DATETIME*w.d*)

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Add a Snowflake Database as a Data Source for SAS Cloud Analytic Services

Use the CASLIB statement to establish a connection between your Snowflake source data and a caslib, SFcaslib. All of the options supplied in this example are required in the CASLIB statement, except the PASSWORD= option.

In this example, the Snowflake data is stored at the location designated by the SERVER= and DATABASE= options.

```
caslib SFcaslib desc='Snowflake Caslib'
     dataSource=(srctype='snowflake',
                 server='SFserver'
                 username='user1',
                 password='myPwd',
                 database='sfdatabase');
```

Example 2: Load Snowflake Data into SAS Cloud Analytic Services Using PROC CASUTIL

```
proc casutil;
   list files incaslib="SFcaslib";                    1
   load casdata="mySFdata" incaslib="SFcaslib" outcaslib="casuser"
        casout="class_from_SFcaslib";                 2
   list files incaslib="casuser";                     3
   contents casdata="%upcase(class_from_SFcaslib)" incaslib="casuser";
                                                      4
quit;
```

1 List the tables in SFcaslib before loading your data.

2 Load the table mySFdata from SFcaslib into caslib Casuser. Call the new table class_from_SFcaslib.

Note: You must specify Snowflake table names using the capitalization that is used in the database.

3 List the tables in Casuser to see the newly created table, class_from_SFcaslib, that you loaded.

4 List information about the newly loaded table, including column names, data types, and so on.

Example 3: Bulk Load to ADLS2 Storage

Here is an example that shows how you might bulk load data to ADLS2 storage. This example sets the AZURETENANTID= value in an OPTIONS statement, using the CASSESSOPS= option. The CASLIB statement specifies that you are using a Snowflake data connector to connect the CAS server to the myschema schema in the Datasource database on Azure. Refer to this connection with the `snow` casref. The Shared Access Signature is the authentication method for the COPY command, and the shared access signature is provided with the azureSAS= data connector option. Any temporary files are deleted from the CAS server and from Azure storage after a process completes.

The CASUTIL procedure takes input from the Casuser casref and outputs to the snow Azure location. The SAVE statement bulk loads the TEST1 CAS table to the Test1_out table in Azure. The defaultDir= option specifies where temporary files reside. These temporary files contain a log of any rejected rows and associated errors. The accountName= and applicationId= values specify the user ID and application ID, respectively, to use when writing to Azure storage. These values must be specified before you can write to Azure, because you did not specify them in the CASLIB statement. You must also provide the fileSystem= value. The data is

not being compressed, and the data that you are loading replaces any existing Test1_out table.

```
options cassessopts=(azuretenantid="user-tenant-id");

caslib snow datasource=(srctype='snowflake',
                        database='datasource',
                        username='myusr1',
                        password='mypwd1',
                        schema='myschema',
                        azureSAS='Shared-Access-Signature',
                        deleteDataFiles=true);

proc casutil incaslib="CASUSER" outcaslib="snow";
    save casdata="TEST1" casout="test1_out"
    options={bulkload=true,
             defaultdir='default-directory',
             accountName='account-name',
             applicationId='application-id',
             fileSystem='filesystem-name',
             compress=false} replace;
run;
```

Example 4: Bulk Unload Snowflake Data from Amazon S3 to CAS

Here is an example that shows how you might bulk unload Snowflake data that is staged in Amazon S3. The CASLIB statement specifies that you are using a Snowflake data connector to connect the CAS server to the `MYUSER` schema in the `Datasource` database on S3. Refer to this connection with the `snow` casref. The Snowflake data is staged in the `mybucket` location in S3. You are accessing the AWS server in the us-east-1 region, and your S3 configuration settings are stored in `config-file`. Data is retrieved from S3 via a bulk unload process. Any temporary files are deleted from the CAS server and from S3 after a process completes.

The LOAD statement uses the `snow` connection to retrieve data from a file called `TEST1` into an in-memory table called `test1_out`. If the `test1_out` table already exists, it is replaced. The data is not compressed.

```
caslib snow desc='Snowflake Caslib'
            datasource=(srctype='snowflake',
                        server='myServer.com',
                        database='Datasource',
                        username='myUser',
                        password='myPwd',
                        schema='MYUSER',
                        bulkUnload=true,
                        S3config='/myConfigDirectory/config-file',
                        bucket='mybucket',
                        region="us-east-1",
                        deleteDataFiles=true);

proc casutil incaslib="snow" outcaslib="CASUSER";
    load casdata="TEST1" casout="test1_out"
       options={compress=false} replace;
```

```
run;
```

Spark Data Connector

Enables you to transfer data between CAS and Spark or CAS and Databricks.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

Note: Starting in 2023.10, SAS redistributes Simba JDBC drivers from insightsoftware for connectivity to Databricks and Spark. These Simba drivers replace the JDBC drivers from CData. If you used CData properties in your connection, you must make changes. For more information, see "Troubleshooting Spark Connection Problems" in *SAS/ACCESS for Relational Databases: Reference*.

See: Spark LIBNAME statement connection

Examples: Define a connection between Databricks in Microsoft Azure and CAS. This example uses single sign-on authentication to Databricks in Azure.

```
caslib databrkscaslib
   datasource=(srctype='spark',
               auth=oauth2,
               schema='default',
               bulkload=no,
               server='server-name',
               port=443,
               useSsl=yes,
               httpPath='myHttpPath');
```

No connection is made to the data source until a statement that accesses the data is called. This statement loads Databricks data using PROC CASUTIL.

```
proc casutil
      incaslib="databrkscaslib"
      outcaslib="casuser" sessref=mysess;
      load incaslib="databrkscaslib" casdata="cars"
      casout="cars_CAS");
run;
quit;
```

Examples: "Publish a Model to a Spark Table and Run in Databricks on AWS" in *SAS In-Database Products: User's Guide*

"Publish a Model to a Spark Table and Run in Azure Databricks" in *SAS In-Database Products: User's Guide*

Details

Data Connector Options for Spark

Data connector options are used in the context of different statements that connect your data in Spark with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about

where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.56 Data Connector Options for Spark

Data Connector Option	Default Value
allowSelfSignedCerts= [1]	value configured in hive-site.xml file
auth=	password
authenticationDomain=	none
authScope=	2ff814a6-3304-4ab8-85cb-cd0e6f879c1d/.default
authType=	LDAP
bufferSize=	1048576
bulkLoad=	TRUE
charMultiplier=	1.0
classPath=	none
clusterId=	none
connCatalog=	none
dbCreateTableOpts=	none
dbMaxText=	1024
dispatchMethod=	STANDARD
driverClass=	"com.simba.databricks.jdbc.Driver" [Databricks] "com.simba.spark.jdbc.Driver" [Spark]
hadoopConfigDir=	none
hadoopJarPath=	none
hdfsTempDir=	"/TMP"
httpPath=	none
insertBuff=	calculated automatically based on row size
invalidDate=	WARN

Data Connector Option	Default Value
jobManagementURL=	none
login_timeout=	30
mapReduceJobName= [1]	value configured in hive-site.xml file
mapReduceQueueName= [1]	value configured in hive-site.xml file
maxLoadThreads=	24
maxSaveThreads=	24
numReadBuffers=	4
numReadNodes=	1
password=	none
platform=	none
port=	10016
properties=	none
readBuff=	Calculated automatically based on row size
schema=	'default'
scratchDb=	none
server=	none
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "spark" for Spark data.
sslTrustStore=.[1]	value configured in hive-site.xml file
statusInterval=	0 (no message)
tezQueueName= .[1]	value configured in hive-site.xml file
trustStorePassword= [1]	value configured in hive-site.xml file

Data Connector Option	Default Value
uri=	none
userName=	none
useSSL=	FALSE
webHdfsUrl=	none
zooKeeperNamespace= [1]	value configured in hive-site.xml file
zooKeeperPort= [1]	none
zooKeeperQuorum= [1]	value configured in hive-site.xml file

1 These options enable you to specify Hadoop configuration options on the CASLIB statement instead of in a configuration file. When a configuration file also exists, the CASLIB settings override the settings in the configuration file.

Spark Platforms

SAS Data Connector for Spark can be used for the following purposes:

- Transfer data between CAS and Spark that resides on a file system. The data connector supports transfer of data from HDFS.
- Transfer data between CAS and Databricks in a cloud-hosted environment, such as Microsoft Azure, Google Cloud Platform, and Amazon Web Services.
- Run scoring models in Databricks or Synapse, using SAS Data Accelerator (known as SAS In-Database Technologies prior to 2026.06). The SAS Embedded Process for Spark must be installed. For more information, see *SAS In-Database Products: User's Guide* .

For all uses except running scoring models, the SAS Viya platform constructs an appropriate JDBC URL connection string automatically from values that you specify in the server=, port=, database=, httpPath=, and properties= SAS data connector options. Starting in 2023.10, the data connector automatically determines the platform from the server= and uri= options. The platform= data connector option is now optional.

Note: Users of SAS Data Accelerator (known as SAS In-Database Technologies prior to 2026.06) must connect with the uri= option and specify the host environment in the platform= data connector option.

Spark Naming

The data connector can load Spark tables with names up to 128 characters or with column names that are up to 128 characters.

Authentication to Spark Data Sources

You authenticate to Spark data sources in the CASLIB statement.

Authenticating to Databricks in Microsoft Azure: Single sign-on can be used to authenticate connections to Databricks in Microsoft Azure. Single sign-on on the SAS Viya platform uses Microsoft Entra ID (formerly Azure Active Directory). This requires an administrator to configure Microsoft Entra ID as an OpenID Connect provider. For more information, see "Scenario: OIDC with Microsoft Entra ID" in *SAS Viya Platform: Authentication*. To use single sign-on, specify the auth=OAUTH2 data connector option in the CASLIB statement. The connection uses the default authentication scope that was configured for Databricks connections in Microsoft Entra ID. To use a different authentication scope, you can specify the authScope= data connector option. SAS/ACCESS then connects to Microsoft Azure using Microsoft Entra ID authentication.

When you authenticate using auth=OAUTH2, do not specify either the authenticationDomain= option or the userName= and password= options. These options disable the single sign-on feature.

Authenticating to Databricks on other cloud providers: Use the CASLIB statement and specify a value for either authenticationDomain= or both userName= and password=. Specify the word `token` in the userName= option and a personal access token for the target environment in the password= option.

Spark system access: Use the CASLIB statement and specify a value for either authenticationDomain= or both userName= and password=. For userName=, specify the username for connecting to Spark SQL. For password=, specify the password associated with the username.

Although not typical, it is possible to supply userName= and password= values in separate statements. For example, you might supply a userName= value in the CASLIB statement and then supply a password= value when you specify a LOAD statement that accesses the data. In this situation, you must supply the password= value for each statement that accesses the data.

Load a Connection to a Spark Data Source

When you load a connection to a data source, you specify the type of data source to which SAS Cloud Analytic Services is connecting. However, a connection is not made until you load data from the data source. To load the connection to the data source and to add a caslib that accesses that data source, use the CASLIB statement.

Supported Spark Data Types

The Spark data connector supports these data types for loading Spark data into SAS Cloud Analytic Services. This table shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.57 Supported Spark Data Types

Spark Data Type	CAS Data Type
Character	
CHAR	CHAR
VARCHAR	VARCHAR
STRING	VARCHAR
Numeric	
ARRAY	not supported
BIGINT	INT64
BINARY	VARCHAR
BOOLEAN	INT32
DECIMAL	DOUBLE
DOUBLE	DOUBLE
FLOAT	DOUBLE
	INT32
INTEGER	INT32
INTERVAL	DOUBLE
MAP	not supported
	DOUBLE
SMALLINT	INT32
STRUCT	not supported
TINYINT	INT32
UNIONTYPE	not supported
Date and Time	
DATE	DOUBLE

Spark Data Type	CAS Data Type
TIME	DOUBLE
TIMESTAMP	DOUBLE

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

The examples below provide a basic introduction to common SAS/ACCESS to Spark configurations.

Example 1: Connect to Databricks, Other Cloud Environments

Single sign-on authentication is not available for Amazon Web Services (AWS) or Google Cloud environments. A personal access token must be used. When using a personal access token, specify userName="token" and specify a personal access

token for the target environment in the password= option. Use of the authenticationDomain= option is preferred to avoid exposing sensitive credentials.

Note that no connection is made to the data source until a statement that accesses the data is called.

```
caslib sparkcaslib desc='Spark Caslib'
   dataSource=(srctype='spark',
               user=token,
               password='mytoken',
               server=Databricks-server,
               port=443,
               database=database,
               httppath='myHttpPath',
               useSsl=yes,
               bulkload=false);
```

Example 2: Connect to Spark on HDFS

This example illustrates what a CASLIB statement for making a Spark HDFS connection from CAS might look like. This example assumes the administrator has previously performed the tasks in "Configuring SAS/ACCESS Interface to Hadoop" in *SAS/ACCESS for Relational Databases: Reference* to configure access to HDFS. After HDFS access is configured, you enable HDFS access by specifying the hadoopJarPath= and hadoopConfigPath= options in the CASLIB statement.

For userName=, specify the username for connecting to Spark SQL. For password=, specify the password associated with the username.

Again, note that no connection is made to the data source until a statement that accesses the data is called.

```
caslib sparkcaslib desc='Spark Caslib'
   dataSource=(srctype='spark',
               username='myuserid',
               password=mypasswd,
               hadoopjarpath='/hadoop/jars:hadoop/spark/jars',
               hadoopconfigdir='/hadoop/conf',
               schema='default',
               server='Spark-server',
               port=10016,
               useSSL=yes,
               httpPath='myHttpPath');
```

Example 3: Load a Spark Table into SAS Cloud Analytic Services Using PROC CASUTIL

```
proc casutil;
   list files incaslib="sparkcaslib";                                    1
   load casdata="mySKdata" incaslib="sparkcaslib" outcaslib="casuser"
        casout="SKdata_from_sparkcaslib";                                2
   list tables incaslib="casuser";                                       3
   contents casdata="SKdata_from_sparkcaslib" incaslib="casuser";        4
quit;
```

1 List the tables in sparkcaslib before loading your data.

2 Load the table mySKdata from Spark into the caslib, casuser. Call the new table SKdata_from_sparkcaslib.

3 List the tables in caslib casuser to see the newly created SKdata_from_sparkcaslib table that you loaded.

4 List information about the newly loaded table, including column names, data types, and so on.

SPD Engine Files Data Connector

Enables you to transfer data between CAS and SPD Engine files that reside on a file system, including HDFS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

Restriction: When you use parallel data transfer mode, files cannot be encrypted or compressed, and tables cannot require a Read password.

Requirements: To access SPD Engine files in Hadoop, specify the hdfs=TRUE option.

Parallel data transfer is supported for SPD Engine files in Hadoop. You must license and deploy SAS In-Database Technologies for Hadoop. For the deployment, you might be required to use the Hadoop tracer script to identify and collect the full set of Hadoop JAR and configuration files. For those instructions, see "Configuring Hadoop JAR Files and Cluster Configuration Files" in *SAS SPD Engine: Storing Data in the Hadoop Distributed File System*.

For parallel data transfer, the webHdfsUrl= and jobManagementURL= options are required. You must use the hadoopConfigDir= option to specify the location of required client-side configuration files. The platform= option supports Spark only, not MapReduce. If you specify MapReduce, the data connector prints a warning to the SAS log and uses Spark instead.

Examples: Establish a connection between your SPD Engine files data source on a local path and SAS Cloud Analytic Services using serial mode.

```
caslib spdecaslib desc='SPDE Caslib'
   dataSource=(srctype='spde',
               dataTransferMode="serial",
               mdfpath="/user/MyMDFPath");
```

Establish a connection between your SPD Engine Files data source on HDFS and SAS Cloud Analytic Services using serial mode.

```
caslib spdecaslib desc='SPDE Caslib'
   dataSource=(srctype='spde',
               dataTransferMode="serial",
               hdfs=true,
               hadoopjarpath="/hadoop/jars",
               hadoopconfigdir="/hadoop/conf",
               mdfpath="/user/spdemdfs");
```

Establish a connection between your SPD Engine Files data source and SAS Cloud Analytic Services using parallel mode.

```
caslib spdecaslib desc='SPDE Caslib'
```

```
   dataSource=(srctype='spde',
               dataTransferMode="parallel",
               webHdfsUrl="url",
               jobManagementURL="Livy-rest-url",
               hadoopConfigDir="location-of-client-side-configuration-files",
               hadoopjarpath="/hadoop/jars",
               hadoopconfigdir="/hadoop/conf",
               mdfpath="/user/spdemdfs");
```

Load an SPD Engine Files data source using PROC CASUTIL.

```
proc casutil;
   incaslib="spdecaslib" sessref=mysess;
   load incaslib="spdecaslib" casdata="cars"
      casout="cars_CAS"
      options=(dataTransferMode="serial");
run;
quit;
```

Details

Data Connector Options for SPD Engine Files

Data connector options are used in the context of different statements that connect your data in SPD Engine files with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.58 *Data Connector Options for SPD Engine Files*

Data Connector Option	Default Value
alterPw=	none
authenticationDomain=	none
bufferSize=	1048576
charMultiplier=	1.0
compress=	NO
conOpts=	none
dataPath=	mdfPath= value
dataTransferMode=	SERIAL
encrypt=	NO

Data Connector Option	Default Value
encryptionPassword=	none
hadoopConfigDir=	none
hadoopJarPath=	none
hdfs=	FALSE (serial data transfer), TRUE (parallel data transfer)
hdfsTempDir=	"/tmp"
jobManagementURL=	none
label=	none
masterPw=	none
maxLoadThreads=	24
maxSaveThreads=	24
mdfPath=	none
numReadNodes=	0
numWriteBuffers=	10
padCompress=	0
partSize=	SPD Engine file default
password=	none
platform=	spark
readPw=	none
srcType=	none Use "spde" for SPD Engine file data.
statusInterval=	0 (no message)
userName=	none
webHdfsUrl=	none
writePW=	none

Load an SPD Engine Table

You can load an SPD Engine table into SAS Cloud Analytic Services from any Hadoop cluster to which you have access—. SAS Cloud Analytic Services does not have to be co-located with Hadoop. You can also run SAS Cloud Analytic Services on a subset of the Hadoop machines and still perform the load. Network performance can limit load times.

Load a Connection to an SPD Engine Files Data Source

When you load a connection to a data source, you specify the type of data source to which SAS Cloud Analytic Services is connecting. However, a connection is not made until you load data from the data source.

You can load a connection to a data source in two ways. To load only the connection, call the loadDataSource action. To load the connection to the data source and to define a caslib that accesses that data source, use the CASLIB statement. In addition, if you use the CASLIB statement, you do not need to call the loadDataSource action.

Supported SPD Engine Files Data Types

The SPD Engine Files data connector supports these data types for loading data from SPD Engine files into SAS Cloud Analytic Services. This table shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data. SAS formats are supported when loading and saving SPD Engine files data sets.

Table 7.59 *Supported SPD Engine Files Data Types*

SPD Engine Files Data Type	CAS Data Type
CHAR	CHAR
DOUBLE	DOUBLE

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see “VARCHAR Support for Implicit and Explicit Data Type Conversion” on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a

CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Specify SPD Engine Files as a Data Source for a Caslib in SAS Cloud Analytic Services

Use the CASLIB statement to initialize the data source and add the caslib for SPD Engine file data that is stored in Hadoop. No connection is made to the data source until a statement that accesses the data is called. Data is read serially into the caslib SPDEcaslib. SPD Engine file credentials are required to access the data. You can specify these in the CASLIB statement or when you use statements that access the data.

```
caslib spdecaslib sessref=mysess desc="SPDE caslib"
   datasource=(srctype="spde",
               username="user1",
               password="myPwd",
               mdfpath="hdfs_path_to_your_mdf_files",
               hadoopJarPath="<Hadoop_jar_path_directory>",
               hadoopConfigDir="<Hadoop_configuration_directory>",
               dtm="serial",
               hdfs="true"
);
```

Example 2: Load SPD Engine Files Data into SAS Cloud Analytic Services Using PROC CASUTIL

```
proc casutil;
   list files incaslib="SPDEcaslib";                                    1
   load casdata="mySPDEdata" incaslib="SPDEcaslib" outcaslib="casuser"
        casout="SPDEdata_from_SPDEcaslib";                              2
   list tables incaslib="casuser";                                      3
   contents casdata="SPDEdata_from_SPDEcaslib" incaslib="casuser";      4
quit;
```

1. List the tables in SPD Engine files before loading your data.
2. Load the table mySPDEdata from SPD Engine files into caslib Casuser. Call the new table SPDEdata_from_SPDEcaslib.
3. List the tables in caslib to see the newly created SPDEdata_from_SPDEcaslib table that you loaded.
4. List information about the newly loaded table, including column names, data types, and so on.

Teradata Data Connector

Enables you to transfer data between Teradata and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

Requirement: To transfer data in parallel, you must install the SAS Embedded Process in the data source. See *SAS In-Database Products: Deployment and Administration Guide*. The SAS Embedded Process is delivered with SAS Data Accelerator. For information about licensing SAS Data Accelerator (known as SAS In-Database Technologies for Teradata prior to 2026.08), contact your designated on-site SAS support personnel.

See: Teradata LIBNAME statement

Examples: Create a caslib that connects Teradata and CAS.

```
caslib tdlib
     datasource=(srctype='teradata'
                 dataTransferMode='parallel'
                 server='TDserver'
                 username='user1'
                 password='myPwd'
                 database='TDdatabase');
```

Load Teradata source data using PROC CASUTIL.

```
proc casutil;
   load incaslib='tdlib' casdata='cars'
        casout='cars_CAS'
        options=(dataTransferMode='parallel');
quit;
```

Details

Data Connector Options for Teradata

Data connector options are used in the context of different statements that connect your data in Teradata with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.60 *Data Connector Options for Teradata*

Data Connector Option	Default
authenticationDomain=	none
authScope=	none
authType=	AUTO
bufferSize=	1048576
catalog=	Active caslib
client_encoding=	UTF-8
database=	none
dataTransferMode=	"SERIAL"
dbCreateTableOpts=	none
dbMaxText=	32000
insertBuff=	Calculated automatically based on row size
maxLoadThreads=	24
maxSaveThreads=	24
numReadBuffers=	4
numReadNodes=	1
numWriteNodes=	1
password=	none
readBuff=	Calculated automatically based on row size
role=	none
scanStringColumns=	none
schema=	none
server=	none
sliceColumn=	none

Data Connector Option	Default
sliceExpressions=	none
srcType=	none Use "teradata" for Teradata data.
ssl_ca=	none
ssl_ca_path=	none
sslcrc=	ALLOW
sslMode=	none
sslProtocol=	none
statusInterval=	0 (no message)
userName=	none

Teradata Naming

The data connector and data connect accelerator can load Teradata tables with names up to 128 characters or with column names that are up to 128 characters.

Loading Data in Parallel

How Parallel Processing Works

To implement parallel data transfer, you must license SAS Data Accelerator (known as SAS In-Database Technologies for Teradata prior to 2026.08). You must install the SAS Embedded Process in the data source. Then, specify dataTransferMode="PARALLEL" to implement parallel data transfer.

To transfer data in parallel, the SAS Viya platform spreads the data across multiple connections for parallel data transfer between Teradata and CAS. The data distribution in the database is determined by the Teradata Primary Index (PI) for tables or by a hash that is calculated by the database for Teradata views. Talk to your Teradata administrator or review the Teradata user documentation to get more information about how Teradata distributes data across its units of parallelism (AMPs). The more evenly the data is divided, the more efficiently the data can be transferred between Teradata and CAS.

Case Sensitivity and Filtering Data

When using parallel data transfer, ensure case-insensitive filtering of data by using the UPCASE function. For example, suppose you have this Sample data set:

```
id col1      col2
3        ABCD1234    bcde2346
4        aBcD1234    BcDe2345
5        aacD1234    BcDe2345
1        abcd1234    BCDE2345
2        ABCD1234    bcde2345
```

Use the UPCASE function to ensure that you get a case-insensitive match for the values in Col1. In this example, you are loading data from the TDsample table in the Tddata caslib. The resulting table, ABCDdata, contains all observations where the Col1 value matches 'ABCD1234', regardless of the original case of the data.

Note: You are not changing the case of the data in the resulting table.

```
caslib tddata datasource=(srctype='teradata'
                          dataTransferMode='parallel'
                          server='TDserver'
                          username='user1'
                          password='myPwd'
                          database='TDdatabase');

proc casutil;
   load casdata="TDsample" incaslib="tddata" outcaslib="casuser"
      casout="ABCDdata"
      where="upcase(col1)='ABCD1234'";
   run;
quit;
```

The resulting ABCDdata table contains observations for ID values 1, 2, 3, and 4.

Security: Authentication with Teradata

Authentication to a Teradata Database

Credentials are required to access the data in a Teradata database. Typically, you supply credentials when you add a caslib. To do this, use the CASLIB statement and specify userName= and password= values. These credentials are then used for any statement that accesses the data using that caslib.

Alternatively, your system administrator might have defined an authentication domain that is associated with a pair of user ID and password values. The authentication domain and associated credentials are then available to you and other users who need to access data. In this case, you can specify an authentication domain when you add a caslib by using the authenticationDomain= option. Use the authenticationDomain= option instead of userName= and password= values.

You can specify different credentials when you issue a statement that accesses the data. If you originally specified an authentication domain when you added a caslib, then you must specify both a userName= value and a password= value to override the original credentials. You can also override authentication domain credentials with a different authenticationDomain= value.

Although it is not typical, it is possible to supply userName= and password= values in separate statements. For example, you might supply a userName= value in the CASLIB statement, and then you supply a password= value when you specify a

LOAD statement that accesses the data. In this situation, you must supply the password= value for each statement that accesses the data.

Authenticate Using Single Sign-On with Microsoft Azure with Entra ID

Single sign-on (SSO) to Teradata on Microsoft Azure uses Microsoft Entra ID to control authentication. This requires an administrator to configure Microsoft Entra ID as an OpenID Connect (OIDC) provider. For more information, see "Scenario: OIDC with Microsoft Entra ID" in *SAS Viya Platform: Authentication*.

SSO authentication to Teradata uses JSON web tokens (JWTs). This requires integration of JWTs with your OIDC provider. For more information, see "Configure JWT Client Authentication" in *SAS Viya Platform: Authentication*.

When authType=AUTO (the default), authentication uses SSO as long as you specify a valid value for your authentication scope. Set the authentication scope with the authScope= data connector option.

If you do not provide a value for authScope=, then use either authenticationDomain= (recommended) or the userName= and password= options.

Authentication Using TD2

Teradata 2 (TD2) is the default authentication method that is provided by Teradata. When TD2 is the default authentication method, use your standard user ID and password values. To use TD2 authentication when TD2 is not the default authentication method, append the "@TD2" token to the Teradata user name and provide the corresponding password. Appending the "@TD2" token to a user name overrides any default authentication that is configured.

This example shows how to connect using a TD2 user name:

```
libname x teradata user="janedoe@TD2" password="janedoeworld";
```

For more information about TD2 authentication, see the Teradata security documentation.

Authentication Using Single Sign-On with Kerberos

Kerberos authentication allows nodes to communicate over a network after providing credentials to the system in a secure manner. Typically, symmetric key encryption is used to mask credentials during the authentication process. Kerberos authentication provides mutual authentication between the nodes on a network. Once authenticated, nodes can communicate freely across the network.

To use single sign-on (SSO) to Teradata with Kerberos authentication, your system must have a valid Kerberos ticket (Ticket Granting Ticket or TGT) before you can initiate a connection to Teradata using Kerberos authentication. On Linux systems, one way to generate a TGT is to use the `kinit` command. Other methods are available to the security administrator at your site. However, methods to authenticate are specific to your site. The security administrator must properly set the TGT expiration time so that any long running jobs have time to complete.

Here are the ways to initiate Kerberos authentication to Teradata with your data connector:

- Remove userName= and password= options or the authenticationDomain= option from any statement that uses these options in existing code. Omit these options when you enter new code. This is the recommended way to specify Kerberos single sign-on as your authentication method.
- Indicate SSO by appending your user ID with "@KRB5", such as `userName="myname@KRB5"`. Alternatively, you can specify simply `userName="@KRB5"`. This syntax ignores any values that you supply for userName= and password= and sets the authentication method to Kerberos (KRB5).

  Note: When using Kerberos authentication, do not also specify the authenticationDomain= option.

Authentication Using LDAP

LDAP is an authentication method that uses an LDAP server to store user names and passwords. Peripheral applications pass user names and passwords to the LDAP server for user authentication. The LDAP authentication system at your organization might also use security realms. These realms define groups, such as administrators or managers, that have access to defined roles and subsets of information.

For LDAP authentication with either a NULL or single security realm, append only the "@LDAP" token to the Teradata user name. In this case, no realm name is needed. If you append a realm name, the LDAP authentication server ignores it and authentication proceeds.

This example shows how to connect to a single or NULL realm:

```
caslib tdlib
   datasource=(srctype='teradata'
               dataTransferMode='parallel'
               server='TDserver'
               username='johndoe@LDAP'
               password='johndoepwd'
               database='TDdatabase');
```

If your system is configured with multiple security realms, you must append the realm name to the "@LDAP" token. In this case, an LDAP server must already be configured to accept authentication requests from the Teradata server.

Here is an example of how to make the connection to a specific realm, jsrealm, where multiple realms are configured:

```
caslib tdlib
   datasource=(srctype='teradata'
               dataTransferMode='parallel'
               server='TDserver'
               username='johndoe@LDAPjsrealm'
               password='johndoepwd'
               database='TDdatabase');
```

Single sign-on is not supported with LDAP authentication.

TLS Support for CASLIB Connections

In 2026.03, support was added for several data connector options that enable you to control TLS encryption for CASLIB connections to Teradata. Here are the data connector options that were added for Teradata:

- ssl_ca=
- ssl_ca_path=

  Note: Specify either ssl_ca= or ssl_ca_path= but not both.

- sslcrc=
- sslMode=
- sslProtocol=

Supported Teradata Data Types

The following table shows the data types that can be read into CAS from a Teradata database. This table also shows the resulting data type and format when data is read into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.61 *Supported Teradata Data Types*

Teradata Data Type	CAS Data Type	Default Format in CAS
	Character	
CHAR	CHAR	$CHAR*w.*
VARCHAR	VARCHAR	$CHAR*w.*
	Numeric	
DECIMAL	DOUBLE	*w.d*
DOUBLE	DOUBLE	*w.d*
NUMBER	DOUBLE	*w.d*
INTEGER	INTEGER	*w.*
BYTEINT (I1)	INTEGER	*w.*
SMALLINT (I2)	INTEGER	*w.*
BIGINT (I8)	BIGINT	*w.*

Teradata Data Type	CAS Data Type	Default Format in CAS
	Date and Time	
DATE (DA)	DOUBLE	DATE*w.*
TIME (AT)	DOUBLE	TIME*w.d*
TIMESTAMP (TS)	DOUBLE	DATETIME*w.d*

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

Because data is stored in CAS with encoding UTF-8, which uses the UNICODE character set, the maximum length for CHAR and VARCHAR data in CAS is 32000 for Teradata. If you attempt to load data that is longer than this, you receive an error in the SAS log.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Example: Load Teradata Data into SAS Cloud Analytic Services Using PROC CASUTIL

```
caslib TDlib desc='Teradata Caslib' 1
     dataSource=(srctype='Teradata',
                 dataTransferMode='serial',
                 server='teradataServer',
                 username='user1',
                 password='******',
                 );

proc casutil;
   list files incaslib="Teradatacaslib";           2
   load casdata="myTDdata" incaslib="Teradatacaslib"
outcaslib="casuser"
        casout="TDdata_from_Teradatacaslib";       3
   list tables incaslib="casuser";                 4
   contents casdata="TDdata_from_Teradatacaslib" incaslib="casuser";
                                                   5
quit;
```

1 Use the CASLIB statement to initialize the data source and add the caslib for Teradata. No connection is made to the database until a statement that accesses the data is called. For this CASLIB statement, the data is read serially into the caslib Teradatacaslib.

2 List the tables in Teradatacaslib before loading your data.

3 Load the table myTDdata from Teradata into caslib Casuser. Call the new table TDdata_from_Teradatacaslib.

4 List the tables in caslib to see the newly created table, TDdata_from_Teradatacaslib, that you loaded.

5 List information about the newly loaded table, including column names, data types, and so on.

Vertica Data Connector

Enables you to transfer data between Vertica and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

See: Vertica LIBNAME statement

Examples: Establish a connection between your Vertica database and SAS Cloud Analytic Services.

```
caslib verticacaslib desc='Vertica Caslib'
     dataSource=(srctype='vertica'
                 server='VTserver'
                 username='user1'
                 password='myPwd'
                 database="VTdatabase-name"
```

```
                schema='myschema');
```

Override the user and password values.

```
proc casutil;
   load casdata="vtexamp" casout="myVTdata" incaslib="verticacaslib"
      outcaslib="casuser"
      dataSourceOptions=(
         username='user5'
         password='myPwd');
quit;
```

Details

Data Connector Options for Vertica

Data connector options are used in the context of different statements that connect your data in Vertica with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.62 *Data Connector Options for Vertica*

Data Connector Option	Default Value
authenticationDomain=	none
catalog=	Active caslib
client_encoding=	UTF-8
conOpts=	none
database=	none
dbCreateTableOpts=	none
insertBuff=	Calculated automatically based on row size
numReadBuffers=	4
numReadNodes=	1
numWriteNodes=	1
password=	none

Data Connector Option	Default Value
port=	5433
readBuff=	Calculated automatically based on row size
schema=	none
server=	none
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "vertica" for Vertica data.
useNarrowCharacterTypes=	FALSE
userName=	none
vertica_dsn=	none

Case Sensitivity with Vertica

All quoted values are passed to the Vertica database exactly as you type them. This means that you must specify all values, such as table names or ID values, using the same capitalization that is used in the Vertica database.

Supported Vertica Data Types

The Vertica data connector supports these data types for loading data from Vertica into CAS. This table shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Vertica Data Type	CAS Data Type
Character	
CHAR	CHAR
VARCHAR	VARCHAR
LONG VARCHAR	VARCHAR
Numeric	

Vertica Data Type	CAS Data Type
BIGINT	INT64
DECIMAL	DOUBLE
DOUBLE PRECISION	DOUBLE
FLOAT	DOUBLE
INTEGER or INT	INT64
NUMERIC	DOUBLE
REAL	DOUBLE
SMALLINT	INT64
TINYINT	INT64
Date and Time	
DATE	DOUBLE formatted as DATE
TIME without TIMEZONE	DOUBLE formatted as TIME
TIMESTAMP without TIMEZONE	DOUBLE formatted as DATETIME
INTERVAL DAY TO SECOND	CHAR
INTERVAL YEAR TO MONTH	CHAR
Other	
BINARY	CHAR (not fully supported)
BOOLEAN	DOUBLE
VARBINARY	VARCHAR (not fully supported)

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Add a Vertica Database as a Data Source for SAS Cloud Analytic Services

Use the CASLIB statement to establish a connection between your Vertica source data and a caslib, Verticacaslib. All options that are supplied in this example are required in the CASLIB statement except the SCHEMA= option.

In this example, the Vertica data is stored at the location designated by the SERVER= and DATABASE= options.

```
caslib Verticacaslib desc='Vertica Caslib'
     dataSource=(srctype='vertica',
                 server='VTserver'
                 username='user1',
                 password='myPwd',
                 database='VTdatabase',
                 schema='myschema');
```

Example 2: Load Vertica Data into SAS Cloud Analytic Services Using PROC CASUTIL

```
proc casutil;
   list files incaslib="Verticacaslib";                    1
   load casdata="myVTdata" incaslib="Verticacaslib" outcaslib="casuser"
        casout="class_from_Verticacaslib";                 2
   list files incaslib="casuser";                          3
   contents casdata="%upcase(class_from_Verticacaslib)"
      incaslib="casuser";                                  4
quit;
```

1 List the tables in Verticacaslib before loading your data.

2 Load the table myVTdata from Verticacaslib into caslib Casuser. Call the new table class_from_Verticacaslib.

Note: You must specify Vertica table names using the capitalization that is used in the database.

3 List the tables in Casuser to see the newly created table, class_from_Verticacaslib, that you loaded.

4 List information about the newly loaded table, including column names, data types, and so on.

Yellowbrick Data Connector

Enables you to transfer data between Yellowbrick and CAS.

Valid in: CASLIB statement, PROC CASUTIL statements (see options for details)

See: Yellowbrick LIBNAME engine

Examples: Establish a connection between your Yellowbrick database and SAS Cloud Analytic Services.

```
caslib ybrickcaslib desc='Yellowbrick Caslib'
     dataSource=(srctype='yellowbrick'
                 server='YBserver'
                 username='user1'
                 password='myPwd'
                 database="YBdatabase-name"
                 schema='myschema');
```

Override the user and password values.

```
proc casutil;
   load casdata="ybexamp" casout="myYBdata" incaslib="yellowbrickcaslib"
      outcaslib="casuser"
      dataSourceOptions=(
         username='user5'
         password='myPwd');
```

```
quit;
```

Details

Data Connector Options for Yellowbrick

Data connector options are used in the context of different statements that connect your data in Yellowbrick with CAS. For each described option, the applicable statements where you can use that option are indicated. For information about where to specify these options within statements, see "Where to Specify Data Connector Options" on page 175.

Table 7.63 *Data Connector Options for Yellowbrick*

Data Connector Option	Default Value
authenticationDomain=	none
catalog=	Active caslib
charMultiplier=	1.0
client_encoding=	Shell encoding
conOpts=	none
database=	none
dbType=	none
insertBuff=	Calculated automatically based on row size
numReadBuffers=	4
numReadNodes=	1
numWriteNodes=	1
password=	none
port=	none
readBuff=	Calculated automatically based on row size
schema=	none
server=	none

Data Connector Option	Default Value
sliceColumn=	none
sliceExpressions=	none
srcType=	none Use "yellowbrick" for Yellowbrick data.
sslMode=	prefer
useNarrowCharacterTypes=	FALSE
userName=	none
yellowbrick_dsn=	none

Case Sensitivity with Yellowbrick

All quoted values are passed to the Yellowbrick database exactly as you type them. This means that you must specify all values, such as table names or ID values, using the same capitalization that is used in the Yellowbrick database.

Supported Yellowbrick Data Types

The following table shows the data types that can be loaded from Yellowbrick into CAS. This table also shows the resulting data type for the data after it has been loaded into CAS. The length of the data format in CAS is based on the length of the source data.

Table 7.64 *Supported Yellowbrick Data Types*

Yellowbrick Data Type	CAS Data Type
Character	
CHAR(*n*)	CHAR
VARCHAR(*n*)	VARCHAR
Numeric	
DECIMAL DECIMAL(*m*) DECIMAL(*m.n*)	DOUBLE
DOUBLE PRECISION	DOUBLE

Yellowbrick Data Type	CAS Data Type
FLOAT	DOUBLE
BIGINT	INT64
INTEGER	INT32
SMALLINT	INT32
NUMERIC NUMERIC(*m*) NUMERIC(*m.n*)	DOUBLE
REAL	DOUBLE
Date and Time	
DATE	DOUBLE (formatted as DATE9.)
TIME(*p*) (without time zone)	DOUBLE (formatted as TIME*w.d*)
TIMESTAMP(*p*)	DOUBLE (formatted as DATETIME*w.d*)
Network Address	
IPV4	VARCHAR
IPV6	VARCHAR
MACADDR	VARCHAR
MACADDR8	VARCHAR
Other	
BOOLEAN	VARCHAR
UUID	CHAR

VARCHAR Data and the CAS Server

The CAS server supports loading, storing, and writing VARCHAR data. All tasks that can be completed using the CAS server use VARCHAR data. Any tasks that are

not completed by the CAS server convert VARCHAR data to fixed-length CHAR data before the data is printed or saved.

For more information, see "VARCHAR Support for Implicit and Explicit Data Type Conversion" on page 46.

Integer Data Types and Numeric Precision

The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved. A SAS DOUBLE value maintains approximately 15 digits of precision.

When you read values that contain more than 15 decimal digits of precision into a DATA step that is running on the Compute Server, the data is converted to a DOUBLE value. Therefore, it is recommended that you check whether a DATA step runs on the CAS server. Check the log to verify where your DATA step is running. For more information, see "DATA Step Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When you read values that contain more than 15 decimal digits of precision into a procedure that is not supported on CAS, the values are rounded and converted to a DOUBLE value with 15 digits of precision. Most procedures are supported on the CAS server. However, it is recommended that you verify that a procedure is supported before you use it for large numeric values. For more information, see *Base SAS Procedures Guide*.

Examples

Example 1: Add a Yellowbrick Database as a Data Source for SAS Cloud Analytic Services

Use the CASLIB statement to establish a connection between your Yellowbrick source data and a caslib, Yellowbrickcaslib. All of the options supplied in this example are required in the CASLIB statement, except the SCHEMA= option.

In this example, the Yellowbrick data is stored at the location designated by the SERVER= and DATABASE= options.

```
caslib Yellowbrickcaslib desc='Yellowbrick Caslib'
     dataSource=(srctype='yellowbrick',
                 server='YBserver'
                 username='user1',
                 password='myPwd',
                 database='YBdatabase',
                 schema='myschema');
```

Example 2: Load Yellowbrick Data into SAS Cloud Analytic Services Using PROC CASUTIL

```
proc casutil;
   list files incaslib="Yellowbrickcaslib";                    1
   load casdata="myYBdata" incaslib="Yellowbrickcaslib"
outcaslib="casuser"
        casout="class_from_Yellowbrickcaslib";                 2
   list files incaslib="casuser";                              3
   contents casdata="class_from_Yellowbrickcaslib" incaslib="casuser";
                                                               4
quit;
```

1 List the tables in Yellowbrickcaslib before loading your data.

2 Load the table myYBdata from Yellowbrickcaslib into caslib Casuser. Call the new table class_from_Yellowbrickcaslib.

3 List the tables in Casuser to see the newly created table, class_from_Yellowbrickcaslib, that you loaded.

4 List information about the newly loaded table, including column names, data types, and so on.

8

Data Connector Options

Dictionary ... **388**
accountName= Data Connector Option ... 388
aggregatePushdown= Data Connector Option ... 389
allow_unquoted_names= Data Connector Option ... 391
allowLargeResults= Data Connector Option ... 391
allowMovedPaths= Data Connector Option ... 392
allowSelfSignedCerts= Data Connector Option ... 393
alter= Data Connector Option ... 393
alterPW= Data Connector Option ... 394
append= Data Connector Option ... 394
applicationId= Data Connector Option ... 396
auth= Data Connector Option ... 396
authEndpoint= Data Connector Option ... 397
authenticationDomain= Data Connector Option ... 398
authenticationScope= Data Connector Option ... 399
authMethod= Data Connector Option ... 400
authScope= Data Connector Option ... 400
authType= Data Connector Option ... 401
awsAutoCreate= Data Connector Option ... 402
awsConfig= Data Connector Option ... 403
awsDurTime= Data Connector Option ... 403
awsIAMassumeRoleARN= Data Connector Option ... 404
azureAccountName= Data Connector Option ... 405
azureBlobEndpoint= Data Connector Option ... 405
azureClientId= Data Connector Option ... 406
azureClientSecret= Data Connector Option ... 406
azureDnsSuffix= Data Connector Option ... 407
azureResource= Data Connector Option ... 407
azureTenantId= Data Connector Option ... 408
azureSas= Data Connector Option ... 408
backingStore= Data Connector Option ... 409
baseDir= Data Connector Option ... 410
blAllowReadAccess= Data Connector Option ... 410
blAuthDomain= Data Connector Option ... 411
blCopyLocation= Data Connector Option ... 412
blCpuParallelism= Data Connector Option ... 413
blDataBufferSize= Data Connector Option ... 414

blDbaccessPath= Data Connector Option ... 415
blDefaultDir= Data Connector Option ... 415
blDelDatFile= Data Connector Option ... 416
blDelimiter= Data Connector Option ... 416
blDiskParallelism= Data Connector Option ... 417
blEnableLog= Data Connector Option ... 418
blEncKey= Data Connector Option ... 418
blErrors= Data Connector Option ... 419
blEscape= Data Connector Option ... 419
blExceptionTable= Data Connector Option ... 420
blFormat= Data Connector Option ... 421
blIndexingMode= Data Connector Option ... 422
blLoad= Data Connector Option ... 422
blLoadReplace= Data Connector Option ... 423
blLog= Data Connector Option ... 423
blLogDir= Data Connector Option ... 424
blNfxSdkPath= Data Connector Option ... 425
blNull= Data Connector Option ... 425
blNumDataFiles= Data Connector Option ... 426
blNumReadThreads= Data Connector Option ... 426
blobThreshold= Data Connector Option ... 427
blOptions= Data Connector Option ... 428
blParallel= Data Connector Option ... 429
blPSQLpath= Data Connector Option ... 429
blPortMax= Data Connector Option ... 430
blPortMin= Data Connector Option ... 430
blQuote= Data Connector Option ... 431
blRecoverable= Data Connector Option ... 431
blRemoteFile= Data Connector Option ... 432
blSkip= Data Connector Option ... 433
blSkipIdxMaint= Data Connector Option ... 433
blSkipUnusableIdx= Data Connector Option ... 434
blStreamSize= Data Connector Option ... 435
blWarningCount= Data Connector Option ... 435
blWriteBuff= Data Connector Option ... 436
branded_driver= Data Connector Option ... 437
bucket= Data Connector Option ... 437
bufferSize= Data Connector Option ... 438
bulk_api= Data Connector Option ... 439
bulk_refresh= Data Connector Option ... 439
bulk_timeout= Data Connector Option ... 440
bulkLoad= Data Connector Option ... 440
bulkUnload= Data Connector Option ... 441
catalog= Data Connector Option ... 442
charMultiplier= Data Connector Option ... 443
class= Data Connector Option ... 444
classPath= Data Connector Option ... 444
client_encoding= Data Connector Option ... 445
client_id= Data Connector Option ... 446
client_secret= Data Connector Option ... 447
clientDir= Data Connector Option ... 447
clientSecret= Data Connector Option ... 448
clusterId= Data Connector Option ... 449
compress= Data Connector Option ... 450

connCatalog= Data Connector Option ... 451
connectionTypeForCreate= Data Connector Option ... 452
conOpts= Data Connector Option ... 454
copyOptions= Data Connector Option ... 455
createMultipassColumn= Data Connector Option ... 455
createViewSnapshot= Data Connector Option ... 459
credentialsFile= Data Connector Option ... 464
credentialsProfile= Data Connector Option ... 465
credFile= Data Connector Option ... 466
csvOptions= Data Connector Option ... 466
database= Data Connector Option ... 467
dataPath= Data Connector Option ... 468
dataSource_dsn= Data Connector Option ... 468
dataTransferMode= Data Connector Option ... 470
dbCreateTableOpts= Data Connector Option ... 471
dbMaxText= Data Connector Option ... 472
dbType= Data Connector Option ... 473
default_auth_plugin= Data Connector Option ... 476
defaultDir= Data Connector Option ... 476
deleteDataFiles= Data Connector Option ... 477
delimiter= Data Connector Option ... 478
directoriesAsData= Data Connector Option ... 479
dispatchMethod= Data Connector Option ... 480
dm_unicode= Data Connector Option ... 481
dnsSuffix= Data Connector Option ... 481
driver_vendor= Data Connector Option ... 482
driverClass= Data Connector Option ... 483
duckdbThreads= Data Connector Option ... 483
dsn= Data Connector Option ... 484
enable_cleartextPlugin= Data Connector Option ... 485
encoding= Data Connector Option ... 485
encrypt= Data Connector Option ... 486
encryptionPassword= Data Connector Option ... 486
epDatabase= Data Connector Option ... 487
epProperties= Data Connector Option ... 488
escape= Data Connector Option ... 489
excelOptions= Data Connector Option ... 490
extensionDirectory= Data Connector Option ... 490
filePath= Data Connector Option ... 491
fileSystem= Data Connector Option ... 492
fileType= Data Connector Option ... 493
folder= Data Connector Option ... 495
getNames= Data Connector Option ... 496
getSheetNames= Data Connector Option ... 496
gcsAccessKey= Data Connector Option ... 497
gcsAuthDomain= Data Connector Option ... 497
gcsCredFile= Data Connector Option ... 498
gcsSecret= Data Connector Option ... 499
hadoopConfigDir= Data Connector Option ... 499
hadoopJarPath= Data Connector Option ... 501
hdfs= Data Connector Option ... 502
hdfsTempDir= Data Connector Option ... 502
host= Data Connector Option ... 503
httpPath= Data Connector Option ... 503

iamAssumeRoleARN= Data Connector Option ... 504
iamAssumeRoleName= Data Connector Option ... 505
icebergCatalog= Data Connector Option ... 505
icebergClientId= Data Connector Option ... 506
icebergClientSecret= Data Connector Option ... 507
icebergEndpoint= Data Connector Option ... 507
icebergEndpointType= Data Connector Option ... 508
icebergOauthUri= Data Connector Option ... 508
icebergSecretOptions= Data Connector Option ... 509
icebergToken= Data Connector Option ... 509
identity= Data Connector Option ... 510
ignoreFedSQLObjects= Data Connector Option ... 511
insertBuff= Data Connector Option ... 511
insertMultiValue= Data Connector Option ... 512
insertSQL= Data Connector Option ... 513
instance= Data Connector Option ... 514
invalidDate= Data Connector Option ... 514
jobManagementURL= Data Connector Option ... 515
jsonOptions= Data Connector Option ... 516
key= Data Connector Option ... 516
krbAuthType= Data Connector Option ... 517
krbHostFQDN= Data Connector Option ... 518
krbRealm= Data Connector Option ... 519
krbServiceName= Data Connector Option ... 520
label= Data Connector Option ... 521
largeResultsDataSet= Data Connector Option ... 521
largeResultsExpirationTime= Data Connector Option ... 522
largeResultsProject= Data Connector Option ... 522
ldPath= Data Connector Option ... 523
login_timeout= Data Connector Option ... 524
mapReduceJobName= Data Connector Option ... 524
mapReduceQueueName= Data Connector Option ... 525
masterPW= Data Connector Option ... 525
max_binary_len= Data Connector Option ... 526
max_char_len= Data Connector Option ... 526
max_string_size= Data Connector Option ... 527
max_array_len= Data Connector Option ... 528
maxErrors= Data Connector Option ... 529
maxLoadThreads= Data Connector Option ... 529
maxSaveThreads= Data Connector Option ... 530
mdfPath= Data Connector Option ... 530
memoryLimit= Data Connector Option ... 531
mode= Data Connector Option ... 531
multipassMemory= Data Connector Option ... 532
mutual_auth= Data Connector Option ... 535
numDataFiles= Data Connector Option ... 535
numReadBuffers= Data Connector Option ... 536
numReadNodes= Data Connector Option ... 537
numWriteBuffers= Data Connector Option ... 538
numWriteNodes= Data Connector Option ... 538
ora_encoding= Data Connector Option ... 539
padCompress= Data Connector Option ... 540
partSize= Data Connector Option ... 541
password= Data Connector Option ... 541

path= Data Connector Option ... 543
platform= Data Connector Option ... 544
pointObs= Data Connector Option ... 545
port= Data Connector Option ... 546
post_table_opts= Data Connector Option ... 547
pre_table_opts= Data Connector Option ... 547
profile= Data Connector Option ... 548
project= Data Connector Option ... 549
properties= Data Connector Option ... 549
proxy= Data Connector Option ... 550
proxy_host= Data Connector Option ... 551
proxy_pass= Data Connector Option ... 551
proxy_port= Data Connector Option ... 552
proxy_user= Data Connector Option ... 552
query_tag= Data Connector Option ... 553
query_timeout= Data Connector Option ... 553
range= Data Connector Option ... 554
read= Data Connector Option ... 555
readBuff= Data Connector Option ... 555
readMode= Data Connector Option ... 556
readOnly= Data Connector Option ... 556
readPW= Data Connector Option ... 557
refresh_token= Data Connector Option ... 557
region= Data Connector Option ... 558
remoteDriverOptions= Data Connector Option ... 558
replace= Data Connector Option ... 559
reuse= Data Connector Option ... 559
role= Data Connector Option ... 560
s3AccessKey= Data Connector Option ... 560
s3Config= Data Connector Option ... 561
s3Region= Data Connector Option ... 562
s3Secret= Data Connector Option ... 562
s3Token= Data Connector Option ... 563
saveMultipassColumn= Data Connector Option ... 563
scanStringColumns= Data Connector Option ... 565
schema= Data Connector Option ... 566
schema_auth= Data Connector Option ... 567
schema_collection= Data Connector Option ... 568
schema_db= Data Connector Option ... 569
schema_password= Data Connector Option ... 569
schema_platform= Data Connector Option ... 570
schema_port= Data Connector Option ... 570
schema_server= Data Connector Option ... 571
schema_ssl_ca= Data Connector Option ... 571
schema_ssl_cert= Data Connector Option ... 572
schema_uid= Data Connector Option ... 572
scratchDb= Data Connector Option ... 573
secret= Data Connector Option ... 573
server= Data Connector Option ... 574
serverDir= Data Connector Option ... 575
sheet= Data Connector Option ... 576
show_metadata= Data Connector Option ... 576
show_recycled= Data Connector Option ... 577
sliceColumn= Data Connector Option ... 577

sliceExpressions= Data Connector Option ... 578
socket= Data Connector Option ... 579
srcType= Data Connector Option ... 580
ssl_ca= Data Connector Option ... 582
ssl_ca_path= Data Connector Option ... 583
ssl_cert= Data Connector Option ... 583
ssl_cipher= Data Connector Option ... 584
ssl_key= Data Connector Option ... 584
sslcrc= Data Connector Option ... 585
sslKeyStore= Data Connector Option ... 585
sslKeyStorePwd= Data Connector Option ... 586
sslMode= Data Connector Option ... 587
sslProtocol= Data Connector Option ... 588
sslTrustStore= Data Connector Option ... 588
statusInterval= Data Connector Option ... 589
table_qualifier= Data Connector Option ... 589
tableType= Data Connector Option ... 590
tempDirectory= Data Connector Option ... 590
tezQueueName= Data Connector Option ... 591
threadedLoad= Data Connector Option ... 591
timeout= Data Connector Option ... 592
token= Data Connector Option ... 593
trustStorePassword= Data Connector Option ... 593
udfDatabase= Data Connector Option ... 594
uri= Data Connector Option ... 595
url= Data Connector Option ... 596
use_information_schema= Data Connector Option ... 597
use_native_names= Data Connector Option ... 598
useLog= Data Connector Option ... 598
useNarrowCharacterTypes= Data Connector Option ... 599
usePipe= Data Connector Option ... 600
userName= Data Connector Option ... 601
useSSL= Data Connector Option ... 602
varcharConversion= Data Connector Option ... 603
warehouse= Data Connector Option ... 603
webHdfsUrl= Data Connector Option ... 603
write= Data Connector Option ... 604
writePW= Data Connector Option ... 605
zooKeeperNamespace= Data Connector Option ... 605
zooKeeperPort= Data Connector Option ... 606
zooKeeperQuorum= Data Connector Option ... 606

Dictionary

accountName= Data Connector Option

specifies the account for accessing an external storage system.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: none

Restriction: This option is applicable when bulk loading data to or bulk unloading from Microsoft Azure or Microsoft Fabric Data Warehouse. For more information, see "Bulk Loading to Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse" on page 241.

Interaction: This value is used to specify the external data location, when combined with the values for dsnSuffix=, fileSystem=, and folder=. The data location is created in this way:

```
https://<accountname>.<dnssuffix>/<filesystem>/<folder>
```

Data source: Microsoft SQL Server, Snowflake

Syntax

ccountName=*"string"*

Required Argument

string
specifies the account for accessing an external storage system.

aggregatePushdown= Data Connector Option

specifies whether to push down aggregation requests to the database.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Default: false

Requirement: For aggregate pushdown that requests kurtosis or skewness statistics, the open-aggregates user-defined functions (UDFs) are required. See the udfDatabase= option for information about the requirement to download and store the UDFs.

Interaction: If the sql= parameter is specified in an action that supports aggregate pushdown, then that setting takes precedence over the aggregatePushdown= data connector option. If sql= is not set, then use aggregatePushdown=true to push down aggregation requests.

Data source: SAS SpeedyStore

Note: This option was added in 2024.11.

See: "udfDatabase= Data Connector Option" on page 594, "Simple Analytics Action Set" in *SAS Visual Analytics: Programming Guide*

Syntax

aggregatePushdown=true | false

Required Arguments

true
enables aggregation requests to be pushed down to the database.

false
does not enable aggregation requests to be pushed down to the database.

Details

Improving Performance Using SQL Pushdown

SQL pushdown is supported for these actions:

- Aggregate pushdown is supported by the simple.distinct action beginning in 2025.03.
- Aggregate pushdown is supported by the simple.summary action beginning in 2024.11.
- SELECT DISTINCT pushdown is supported by the simple.groupBy action beginning in 2024.09.

Aggregate pushdown enables CAS actions to push down aggregation requests to be processed in the database. The aggregated data is streamed to CAS. In general, performance improves when the aggregation can run in the database, instead of streaming the full data to CAS for aggregation. Here is how to process an aggregation request in the database:

- Either specify the aggregatePushdown=true data connector option or specify sql=true in an action that supports aggregate pushdown. If the sql= parameter is specified in the action, then that setting takes precedence over the aggregatePushdown= data connector option. If sql= is not set, then use aggregatePushdown=true to push down aggregation requests.
- Do not use the groupbyTable parameter in the simple.distinct action. This parameter is not supported for pushdown.
- Do not use any of the following parameters in the simple.summary action: freq=, groupByTable=, repeat=, weight=, whereTable=. These parameters are not supported for pushdown.

SELECT DISTINCT pushdown is not an aggregation request, but it does reduce the result set that is returned to CAS. Here is how to implement SELECT DISTINCT pushdown:

- Specify the aggregatePushdown=true data connector option. (The sql= parameter is not available in simple.groupBy.)

- Do not use any of the following parameters in the simple.groupBy action: aggregator=, freq=, scoreGt=, scoreLt=, weight=. These parameters are not supported for pushdown.

When an aggregation request is pushed down to the database, the following message is written in the SAS log:

```
NOTE: The aggregation is being calculated in the database.
```

If a SAS log message indicates that a user-defined function is required, then download the open-aggregates user-defined functions (UDFs). See "udfDatabase= Data Connector Option" on page 594 for more information.

allow_unquoted_names= Data Connector Option

specifies whether table and column names are passed to your data source without enclosing them in quotation marks.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: FALSE

Data source: PostgreSQL

Syntax

allow_unquoted_names=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether table and column names are passed to your data source without enclosing them in quotation marks.

Details

When quotation marks are removed from table and column names, the names are saved based on the default behavior for your data source. For PostgreSQL, unquoted names are saved in lowercase.

allowLargeResults= Data Connector Option

specifies whether to allow query results larger than the Google BigQuery maximum response size.

Valid in: CASLIB statement

Aliases: ALR=
largeResults=

Default: OFF

Interaction: This option is used in conjunction with the LargeResultsDataSet= and LargeResultsExpirationTime= options.

Data source: Google BigQuery

Syntax

allowLargeResults=ALWAYS | MUST | OFF

Required Arguments

ALWAYS
specifies that all query results are written to a temporary stored table, including query results smaller than the maximum response size.

MUST
specifies that query results larger than the maximum response size are written to a temporary stored table.

OFF
specifies that query results larger than the maximum response size trigger an error in the SAS log, and the query fails.

allowMovedPaths= Data Connector Option

specifies whether to allow scanning Iceberg tables that have been moved.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Default: TRUE

Data source: DuckDB

Note: Support for this option was added in 2026.09.

Syntax

allowMovedPaths=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether to allow scanning Iceberg tables that have been moved.

allowSelfSignedCerts= Data Connector Option

enables or disables wire encryption using self-signed certificates.

Valid in: CASLIB statement

Default: value configured in hive-site.xml file

Data source: Hadoop, Spark

See: sslTrustStore= data connector option, trustStorePassword= data connector option

Syntax

allowSelfSignedCerts="NO | YES"

Required Arguments

NO
disables Transport Layer Security wire encryption.

YES
enables Transport Layer Security wire encryption using self-signed certificates, when it is available.

alter= Data Connector Option

specifies the password to access and delete a password-protected data set.

Default: none

Requirement: This option is required only if the data set is password-protected.

Data source: SAS data sets

Syntax

alter=*"password"*

Required Argument

password
specifies the password to access and delete a password-protected data set.

alterPW= Data Connector Option

specifies the password to access, modify, or delete a password-protected data set.

Valid in: CASLIB statement, PROC CASUTIL: DELETESOURCE, LOAD, and SAVE statements

Default: none

Data source: SPD Engine Files

Syntax

alterPW=*"password"*

Required Argument

password
specifies the password to access, modify, or delete a password-protected data set.

Details

This option supports serial and parallel data transfer for the SAVE statement for PROC CASUTIL. This option supports serial data transfer only for the LOAD statement.

append= Data Connector Option

specifies whether to append data from CAS to an existing database table.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: FALSE

Interaction: Do not set replace=TRUE in the SAVE statement at the same time that you set append=TRUE. This causes the save action to end with an error.

Data source: MySQL, Oracle, PostgreSQL, SingleStore standard

Note: This option was added in 2024.09 for PostgreSQL. Support was added in 2024.11 for MySQL, Oracle, and SingleStore standard.

Syntax

append=TRUE | FALSE

Required Arguments

TRUE
specifies that SAS appends data from CAS into an existing database table.

FALSE
specifies that SAS does not append data to an existing database table. Instead, the table is deleted and replaced with the data from CAS.

Details

When append=TRUE, SAS attempts to append data to an existing database table. Here are some rules for appending data:

- The columns in the appended data must exist in the database table. The corresponding column names must be identical.
- A column can exist in the database table that does not exist in the appended data. In that case, missing values are assigned in the appended data.
- As of 2024.11, the columns are not required to be in the same order as in the database table.
- If the database table has a not-null column that you do not specify in the appended data, then the append fails.
- The data type of corresponding columns must be compatible. For example, a column with data type of CHAR(16) can write to a PostgreSQL column of type VARCHAR(20). DOUBLE values can be written to NUMERIC or DECIMAL columns, but the resulting values might be rounded. DATE or TIME values can be written only to columns with DATE, TIME, or TIMESTAMP values. Different databases can act differently, especially with DATE, TIME, and TIMESTAMP data types. It is best to use formats in those cases.

  Note: If necessary, you can use the dbType= data connector option to control the output data type when you write to the data source.

Here are more examples to illustrate the column rules:

- CAS table (a, b) can be appended to the database table (a, b, c) if column c has no not-null constraint.
- CAS table (a, b) can be appended to the database table (b, a).
- CAS table (a, b, d) cannot be appended to the database table (a, b) or database table (a, b, c) because column d does not exist in the database table.

applicationId= Data Connector Option

specifies the application ID (a GUID string) for accessing the external storage system.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: none

Restriction: This option is applicable only when bulk loading data to Microsoft Azure. For more information, see "Bulk Loading to Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse" on page 241.

Interaction: *Micorsoft SQL Server:* When bulk loading to Microsoft Azure or Microsoft Fabric Data Warehouse, if you do not set this option, then the Microsoft SQL Server data connector automatically attempts to obtain a Microsoft Entra ID token, if single sign-on has been configured.

Data source: Microsoft SQL Server, Snowflake

Example:
```
applicationId="b1fc955d5c-e0e2-45b3-a3cc-a1cf54120f"
```

Syntax

applicationId=*"string"*

Required Argument

string
specifies the application ID (a GUID string) for accessing the external storage system.

auth= Data Connector Option

specifies the authorization method that is used to authenticate to an external data source.

Valid in: CASLIB statement

Defaults: AUTO [MongoDB]
PASSWORD [Spark]

Interaction: It is possible to use one authentication method when you connect to the MongoDB data source and a different method when you connect to a MongoDB schema. To use a different authentication method for a schema, use the schema_auth= data connector option.

Data source: MongoDB, Spark

See: "authScope= Data Connector Option" [Spark]

Syntax

auth=AUTO | KERBEROS | LDAP | OAUTH2 | PASSWORD

Required Arguments

AUTO
specifies that SAS uses the default method of userName= and password= authentication to access MongoDB. Alternatively, you can specify an authentication domain with the authenticationDomain= data connector option, which resolves to user ID and password values when you connect to a data source.

Restriction This option applies to MongoDB only.

KERBEROS
specifies that Kerberos authentication is used to access MongoDB.

Restriction This option applies to MongoDB only.

LDAP
specifies that LDAP authentication is used to access MongoDB.

Restriction This option applies to MongoDB only.

OAUTH2
specifies that the SAS credentials service obtains a Microsoft Entra ID access token.

Restriction This option applies to Databricks connections through the Spark data connector in Azure only.

PASSWORD
specifies that userName= and password= values or authentication domain credentials are used to access the data source. Specify an authentication domain with the authenticationDomain= data connector option.

authEndpoint= Data Connector Option

specifies the authorization endpoint to use.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Default: none

Data source: Salesforce

Example: `authendpoint="http://wsware.com/myendpoint"`

Syntax

authEndpoint=*"value"*

Required Argument

value
specifies the authorization endpoint to use.

authenticationDomain= Data Connector Option

specifies the name of the authentication domain that contains credentials (userName= and password=) that are used to access a data source.

Valid in: CASLIB statement [see Requirement], PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Category: Authentication

Alias: authDomain=

Default: none

Restrictions: This option supports only authentication domains of type `Authentication (password)`.

SPD Engine files: The authenticationDomain= option is valid only when SPD Engine files are stored in Hadoop.

Requirements: If your database or data source requires authentication, you must specify valid credentials to access data. You can provide these credentials by specifying either an authenticationDomain= value or by specifying userName= and password= values.

Google BigQuery: This option is required in a CASLIB statement, unless you specify the credFile= option.

Data source: Amazon Redshift, DB2, Google BigQuery, Greenplum, Hadoop, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, Oracle, PostgreSQL, Salesforce, SAP HANA, SAP IQ, SAS SpeedyStore, SingleStore standard, Snowflake, Spark, SPD Engine files, Teradata, Vertica, Yellowbrick

Syntax

authenticationDomain=*"domain"*

Required Argument

domain
specifies the name of the authentication domain that contains credentials that are used to access a data source.

Details

An administrator creates definitions of authentication domains. Authentication domains are managed by a credentials service. Currently, the supported authentication domain type for CASLIB connections is `Authentication (password)`. The authentication domain is resolved when SAS/ACCESS accesses your data source and returns the authentication credentials. The authentication credentials that are returned are the user ID and password values. For more information, see "Credentials Domain: How to (CLI)" in *SAS Viya Platform: External Credentials*.

In general, you specify an authenticationDomain= value when you add a caslib. The associated credentials are then used for any statement that accesses the data source. If you specify an authenticationDomain= value in a statement other than the CASLIB statement, these credentials override any that were set when the caslib was added.

Google BigQuery: When you connect to Google Cloud Platform, the credentials service returns a JSON credentials file. This differs from other data sources, where user and password values are returned.

authenticationScope= Data Connector Option

specifies the Microsoft Azure resource and permissions that are required for authorizing access.

Valid in:	CASLIB statement
Alias:	authScope=
Defaults:	https://database.windows.net/user_impersonation [Microsoft SQL Server] https://ossrdbms-aad.database.windows.net/user_impersonation [MySQL, PostgreSQL] https://storage.azure.com/user_impersonation [SingleStore standard] none [Amazon Redshift, Snowflake] 2ff814a6-3304-4ab8-85cb-cd0e6f879c1d/.default [Spark]
Data source:	Amazon Redshift, Microsoft SQL Server, MySQL, PostgreSQL, Singlestore standard, Snowflake, Spark
Note:	Support for Amazon Redshift was added in 2025.08.
Example:	`authenticationScope="https://database.windows.net/user_impersonation"`

Syntax

authenticationScope=*"scope"*

Required Argument

scope
specifies the Microsoft Azure resource and permissions that are required for authorizing access.

authMethod= Data Connector Option

specifies the authentication method for single sign-on.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements

Default: Native

Requirement: This option is required for CASLIB connections. If omitted, the default value (Native) is used.

Data source: Amazon Redshift

Note: Support for this option was added in 2025.08.

Syntax

authMethod='IAM' | 'Native'

Required Arguments

IAM
specifies that you are using the IAM method for authentication for SSO.

Native
specifies that you are using the native authentication method for SSO.

authScope= Data Connector Option

specifies the Microsoft Azure resource and permissions that are required for authorizing access.

Valid in: CASLIB statement

Defaults: 2ff814a6-3304-4ab8-85cb-cd0e6f879c1d/.default [Spark]
https://ossrdbms-add.database.windows.net/user_impersonation [MySQL]
none [Teradata]

Data source: MySQL, Spark, Teradata

Note: Support for Teradata was added in 2025.06.

Syntax

authScope=*"scope"*

Required Argument

scope
specifies the Microsoft Azure resource and permissions that are required for authorizing access.

Details

When using the Spark data connector, this option applies to Databricks connections only.

The authentication scope is used when you connect to Microsoft Azure using single sign-on capabilities for the SAS Viya platform. The default values for this option are used when you do not specify authScope= and are intended for Microsoft Azure. Your SAS Viya platform administrator must configure Microsoft Entra ID (formerly Azure Active Directory) as an OpenID Connect provider. For more information, see "Scenario: OIDC with Microsoft Entra ID" in *SAS Viya Platform: Authentication*.

Alternative Microsoft Azure Clouds: If your SAS Viya platform environment is running in an alternative Microsoft Azure Cloud, such as the Microsoft Azure Government Cloud, then you must specify the authScope= option. Valid values for Microsoft Azure Government Cloud can be found at https://learn.microsoft.com/en-us/azure/azure-government/compare-azure-government-global-azure.

authType= Data Connector Option

specifies the authentication method to use when connecting to your data source.

Valid in:	CASLIB statement
Aliases:	auth= authenticationType=
Defaults:	LDAP [Hadoop, Spark] AUTO [Teradata]
Data source:	Hadoop, Spark, Teradata
Note:	Support for Teradata was added in 2025.06.
See:	authScope= data connector option

Syntax

Form 1: Hadoop, Spark:

authType=LDAP

Form 2: Teradata:

authenticationType=AUTO | OAUTH2 | PASSWORD

Required Arguments

AUTO
specifies that single sign-on is used if authScope= is provided. If not, then authentication uses user and password values or an authentication domain value.

Supports Teradata

LDAP
authenticates users by using standard Lightweight Directory Access Protocol (LDAP).

Supports Hadoop, Spark

OAUTH2
specifies that OAUTH2 authentication, which includes a derived user value and a token value, is used to connect to Microsoft Azure.

Supports Teradata

Note For Teradata, even when user and password values (or an authentication domain value) is specified, OAUTH2 authentication is used.

PASSWORD
specifies that user and password (or authentication domain) authentication is used to connect to your data source.

Supports Teradata

Note For Teradata, even when authScope= is specified, user and password authentication is used.

awsAutoCreate= Data Connector Option

specifies whether to create a database user with the name that you specify for userName= if it does not exist.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements

Default: NO

Data source: Amazon Redshift

Syntax

awsAutoCreate=YES | NO

Required Argument

value
specifies whether to create a database user with the name that you specify for userName= if it does not exist.

Details

You should use this option with caution. The user that is created must have permission to access your database. For this reason, the best practice is to use previously defined user credentials.

awsConfig= Data Connector Option

specifies the location of the Amazon Web Services (AWS) configuration file.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: `~/.aws/config`

Requirement: To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.

Data source: Amazon Redshift, Snowflake

Syntax

awsConfig=*"path-and-AWS-configuration-file-name"*

Required Argument

path-and-AWS-configuration-file-name
specifies the location of the Amazon Web Services (AWS) configuration file.

awsDurTime= Data Connector Option

specifies the number of seconds until the temporary password expires.

Valid in:	CASLIB statement, PROC CASUTIL: LOAD and SAVE statements
Alias:	duration_time=
Default:	900 (15 minutes)
Range:	900 - 3600 (15 minutes - 60 minutes)
Data source:	Amazon Redshift
Note:	Support for this option was added in 2025.08.

Syntax

awsDurTime=*value*

Required Argument

value
specifies the number of seconds until the temporary password expires.

Details

You must successfully connect to your Amazon Redshift data source before your temporary token expires. If you have fully established a connection and authenticated before the token expires, then you can continue to work after the temporary password expires.

awsIAMassumeRoleARN= Data Connector Option

specifies an Amazon Resource Name that identifies an IAM role.

Valid in:	CASLIB statement, PROC CASUTIL: LOAD and SAVE statements
Default:	none
Requirement:	This option is required for CASLIB connections.
Data source:	Amazon Redshift
Note:	Support for this option was added in 2025.08.

Syntax

awsIAMassumeRoleARN=*"role-name"*

Required Argument

role-name
specifies an Amazon Resource Name that identifies an IAM role. The associated IAM role enables you to get authentication credentials for the AssumeRole IAM operation.

azureAccountName= Data Connector Option

specifies the Microsoft Azure account name for authentication.

Valid in: CASLIB statement
Alias: azure_accountname=
Default: none
Data source: DuckDB
Note: Support for this option was added in 2026.09.

Syntax

azureAccountName=*"name"*

Required Argument

name
specifies the Microsoft Azure account name for authentication. Enclose the value in quotation marks.

azureBlobEndpoint= Data Connector Option

specifies the storage account under which files are located.

Valid in: CASLIB statement
Aliases: azure_blob_endpoint=
azure_blobendpoint=
Default: none
Data source: DuckDB
Note: Support for this option was added in 2026.09.
See: azureAccountName= on page 405, azureSas= on page 408
Example: `azureBlobEndpoint='storage-account.blob.core.windows.net'`

Syntax

azureBlobEndpoint=<'>*location-URL*<'>

Required Argument

location-URL
specifies the storage account under which files are located.

azureClientId= Data Connector Option

specifies the client ID for authentication.

Valid in: CASLIB statement

Aliases: azure_client_id=
azure_app_id=

Default: none

Data source: DuckDB

Note: Support for this option was added in 2026.09.

See: azureClientSecret= on page 406

Syntax

azureClientId=*client*

Required Argument

client
specifies the client ID for authentication. This value is associated with the Azure client secret value that you specify.

azureClientSecret= Data Connector Option

specifies the Microsoft Azure client secret for authentication.

Valid in: CASLIB statement

Alias: azure_client_secret=

Default: none

Data source: DuckDB

Note: Support for this option was added in 2026.09.

See: azureClientId= on page 406

Syntax

azureClientSecret=*'value'*

Required Argument

value
specifies the Microsoft Azure client secret for authentication. This value is associated with the Microsoft Azure client ID.

azureDnsSuffix= Data Connector Option

specifies the Azure DNS suffix to use for accessing Azure BLOB storage.

Valid in: CASLIB statement

Alias: azure_dns_suffix=

Default: none

Data source: DuckDB

Note: Support for this option was added in 2026.09.

Example: `azureDnsSuffix='blob.machine.net'`

Syntax

azureDnsSuffix=*"DNS-suffix"*

Required Argument

DNS-suffix
specifies the Azure DNS suffix to use for accessing Azure BLOB storage. This value is typically appended to the end of the storage account name.

azureResource= Data Connector Option

specifies the Azure service or API for which an access token is requested.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Default: none

Data source: DuckDB

Note: Support for this option was added in 2026.09.

Syntax

azureResource=*"value"*

Required Argument

value
specifies the Azure service or API for which an access token is requested.

azureTenantId= Data Connector Option

specifies the Microsoft Azure tenant ID for authentication.

Valid in: CASLIB statement

Alias: azure_tenant_id=

Default: none

Data source: DuckDB

Note: Support for this option was added in 2026.09.

Syntax

azureTenantId=*"value"*

Required Argument

value
specifies the Microsoft Azure tenant ID for authentication. Enclose the value in quotation marks.

azureSas= Data Connector Option

specifies the Azure shared access signature token for authentication.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements [Snowflake]

Default: none

Data source: DuckDB, Snowflake

Note: Support for DuckDB was added in 2026.09.

Example: `azureSas='IWUQERPIAWEJDKH...R8YAPHDFJKZSD'`

Syntax

azureSas=*"token"*

Required Argument

token
specifies the Azure shared access signature token for authentication.

backingStore= Data Connector Option

specifies to load the data into CAS or to keep the table in SingleStore.

Valid in: CASLIB statement

Alias: loadTableBackingStore=

Default: INPLACEPREFERRED

Data source: SAS SpeedyStore

Syntax

backingStore="CASDISKCACHE" | "INPLACEPREFERRED"

Required Arguments

CASDISKCACHE
specifies to load the data into the CAS disk cache.

INPLACEPREFERRED
specifies to keep the data in the SingleStore database, and stream to CAS.

Details

When you load a SingleStore table to the CAS server, under the default value backingStore="INPLACEPREFERRED", you avoid using the CAS disk cache. Instead, the server points to the table in SingleStore, and the table is streamed to the server. For very large tables, this use case can improve performance. However, if the table is moderately sized and would completely fit in CAS memory, then the performance

might be better if you set backingStore="CASDISKCACHE". Under that value, when you load a table, the server uses the CAS disk cache instead of streaming from SingleStore. Streaming performance is also negatively affected when multiple actions operate on the same table.

For details about the behavior with SingleStore views, see "createViewSnapshot= Data Connector Option" on page 459.

Comparisons

This syntax in this option is similar to the loadTableBackingStore= parameter in the table.addCaslib action. However, the primary use of loadTableBackingStore= is different, and loadTableBackingStore= is supported for Path or DNFS caslibs.

baseDir= Data Connector Option

specifies the path to the MySQL installation directory.

Valid in: CASLIB statement

Default: none

Data source: MySQL

Syntax

baseDir=*"path-to-installation-directory"*

Required Argument

path-to-installation-directory
specifies the path to the MySQL installation directory. This option sets the BASEDIR environment variable for MySQL.

blAllowReadAccess= Data Connector Option

specifies whether the original table data is still visible to readers during bulk loading.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Alias: bl_allow_read_access=

Default: FALSE

Requirement: To specify this option, you must also set bulkLoad=TRUE.

Data source: DB2

Syntax

blAllowReadAccess=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether the original table data is still visible to readers during bulk loading.

Details

For more information, see the SQLU_ALLOW_READ_ACCESS parameter in your Db2 documentation.

blAuthDomain= Data Connector Option

specifies an authentication domain that provides application ID and client secret values.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: none

Data source: Microsoft SQL Server

Note: Support for this option was added in 2025.08.

See: applicationId=, authenticationDomain=, clientSecret=

Syntax

blAuthDomain=*"authentication-domain"*

Required Argument

authentication-domain
specifies an authentication domain that provides application ID and client secret values for connection to Azure Data Lake Storage when bulk loading data to Microsoft Fabric Data Warehouse or to Azure Synapse Analytics (SQL DW).

Details

Use this option when you are bulk loading data from CAS to Microsoft Fabric Data Warehouse or to Azure Synapse Analytics (SQL DW). The authentication domain that you provide is defined in the credentials service for SAS Viya platform. SAS then passes the associated user and password values for the authentication domain. The user value contains the application (client) ID value and the password value contains the client secret.

In place of blAuthDomain=, you can specify these values by using the applicationId= and clientSecret= values. However, by using blAuthDomain=, you avoid the need to save these values in your SAS program.

For bulk loading, SAS attempts to authenticate to ADLS via these methods:

1 using blAuthDomain=

2 using applicationId= and clientSecret=

3 using authenticationDomain=

Note: The authenticationDomain= value might resolve to standard user and password values or it might resolve to application ID and client secret values. You can use either set of values, depending on your authentication setup.

Another method to authenticate to ADLS for bulk loading is to use single sign-on. For more information, see "Authentication to Azure for Bulk Loading" on page 242.

If you do not use any of the mentioned authentication methods, then SAS checks for userName= and password= values.

blCopyLocation= Data Connector Option

specifies the directory in which SAS saves a copy of the loaded data.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Alias: bl_copy_location=

Default: none

Requirement: To specify this option, you must also set bulkLoad=TRUE and blRecoverable=TRUE.

Data source: DB2

Syntax

blCopyLocation=<'>*path*<'>

Required Argument

path
specifies the directory in which SAS saves a copy of the loaded data.

blCpuParallelism= Data Connector Option

specifies the number of processes or threads to use when building table objects.

Valid in:	CASLIB statement, PROC CASUTIL: SAVE statement
Alias:	bl_cpu_parallelism=
Default:	none
Range:	0 – 30
Requirement:	To specify this option, you must also set bulkLoad=TRUE.
Data source:	DB2

Syntax

blCpuParallelism=*value*

Required Argument

value
specifies the number of processes or threads that the load utility uses to parse, convert, and format data records when building table objects.

Details

This option exploits intrapartition parallelism and can significantly improve load performance. It is particularly useful when loading presorted data, because record order in the source data is preserved.

If the value is 0 or has not been specified, the load utility selects an intelligent default. This default is based on the number of available CPUs on the system at run time. If there is insufficient memory to support the specified value, the utility adjusts the value.

When this value is greater than 1, the flushing operations are asynchronous, permitting the loader to exploit the CPU. If tables include either LOB or LONG VARCHAR data, parallelism is not supported. In this case, the value is set to 1 regardless of the number of system CPUs or the specified value.

Although use of this parameter is not restricted to symmetric multiprocessor hardware, you might not obtain any discernible performance benefit from using it in non-symmetric multiprocessor environments.

For more information, see the CPU_PARALLELISM Db2 parameter in your IBM Db2 documentation.

blDataBufferSize= Data Connector Option

specifies the amount of memory to allocate as a buffer for the bulk-load utility.

Valid in:	CASLIB statement, PROC CASUTIL: SAVE statement
Alias:	bl_data_buffer_size=, bl_databuffersize=
Default:	none
Requirement:	To specify this option, you must also set bulkLoad=TRUE.
Data source:	DB2

Syntax

blDataBufferSize=*"value"*

Required Argument

value
specifies the amount of memory to allocate as a buffered space for the bulk-load utility to use to transfer data. Memory is calculated in 4KB pages regardless of the degree of parallelism.

Details

If you specify a value that is less than the algorithm minimum, then the minimum required resource is used and no warning is returned in the log. This memory is allocated directly from the utility heap, the size of which you can modify through the util_heap_sz Db2 database configuration parameter. If you do not specify a value for this option, then the bulk-load utility calculates an intelligent default at run time. This calculated default is based on a percentage of the free space that is available in the utility heap at the time of instantiation of the loader, as well as on some characteristics of the table.

It is recommended that the buffer be several extents in size. An *extent* is the unit of movement for data within Db2, and the extent size can be one or more 4KB pages. The Data Buffer parameter in Db2 is useful when you are working with large objects (LOBs) because it reduces input and output waiting time. The data buffer is allocated from the utility heap. Depending on the amount of storage available on

your system, you should consider allocating more memory for use by the Db2 utilities. You can modify the Db2 configuration parameter util_heap_sz accordingly. The default value for the Utility Heap Size configuration parameter is 5000 4KB pages. Because load is only one of several utilities that use memory from the utility heap, it is recommended that no more than 50% of the pages specified by this parameter be made available for the load utility.

For more information about the Data Buffer and Utility Heap Size parameters, see your Db2 documentation.

blDbaccessPath= Data Connector Option

specifies the full path to the bulk-loading utility.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: `<INFORMIX_SDK_PATH>/bin/dbaccess`

Requirement: You must also set bulkLoad=TRUE or bulkUnload=TRUE when you use this option.

Data source: Informix

See: bulkLoad=, bulkUnload= on page 441

Syntax

blDbaccessPath=*'path'*

Required Argument

path
specifies the full path to the DBACCESS utility.

blDefaultDir= Data Connector Option

specifies the location where temporary data files are stored.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: current location for storing temporary data files

Data source: Informix, MySQL, SingleStore standard

Syntax

blDefaultDir=*'path'*

Required Argument

path
specifies the location where temporary data files are stored.

blDelDatFile= Data Connector Option

specifies whether to delete only the data file or all files that the data connector creates for the bulk-loading or bulk-unloading facility.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements

Default: TRUE

Requirement: To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.

Data source: Informix

Note: Support for PostgreSQL was removed in 2025.03.

Syntax

blDelDatFile=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether to delete only the data file (FALSE) or all files (TRUE) that the data connector creates with the bulk-loading or bulk-unloading facility.

Details

Informix: This option specifies whether to remove the data file only (FALSE) or the data file, configuration file, control file, and optional log file (TRUE).

PostgreSQL: Starting in 2025.03, bulk loading and bulk unloading uses the COPY API for PostgreSQL. For this reason, the blDelDatFile= option, which works with the PSQL utility, is no longer used.

blDelimiter= Data Connector Option

specifies an override for the default delimiter character during bulk loading or bulk data retrieval (unloading).

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements

Defaults: , (comma) [PostgreSQLl]
| (pipe) [Informix]

Requirement: To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.

Data source: Informix, PostgreSQL

Syntax

blDelimiter=*'single-character'*

Required Argument

single-character
specifies an override for the default delimiter character during bulk loading or bulk data retrieval (unloading).

blDiskParallelism= Data Connector Option

specifies the number of processes or threads to use when writing data to a data source.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Alias: bl_disk_parallelism=

Default: none

Requirement: To specify this option, you must also set bulkLoad=TRUE.

Data source: DB2

Syntax

blDiskParallelism=*processes*

Required Argument

processes
specifies the number of processes or threads to use when writing data to a data source.

Details

This option takes advantage of the number of available containers when loading data. Using this option can significantly improve performance.

The maximum number that is allowed is the greater of 50 or four times the blCpuParallelism= value. By default, this value is equal to the sum of the tablespace containers on all tablespaces that contain objects for the table that is being loaded. However, this value cannot exceed the maximum allowed value. If you do not specify a value, the utility selects an intelligent default that is based on the number of tablespace containers and the characteristics of the table.

For more information, see the DISK_PARALLELISM Db2 parameter in your IBM Db2 documentation.

blEnableLog= Data Connector Option

specifies whether to generate a log file for the bulk-load utility.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: FALSE

Data source: Informix

Syntax

blEnableLog=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether to generate a log file for the bulk-load utility.

Details

The bulk-loading utility for Informix is DBACCESS.

blEncKey= Data Connector Option

specifies the name of an encryption key for Amazon S3.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements

Default: none

Interaction: Specify this option with the bulkLoad= or bulkUnload= data connector option.

Data source: Amazon Redshift

Note: Support for this option was added in 2026.07.

See: bulkLoad=, bulkUnload=, S3 Procedure documentation

Syntax

blEncKey=*"encryption-key-name"*

Required Argument

encryption-key-name
specifies the name that is associated with an encryption key. The name matches a name that is assigned to an encryption key by using the S3 procedure.

blErrors= Data Connector Option

specifies that, after the indicated number of errors is received, the load should stop.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Category: Bulk Loading

Default: 1000000

Data source: Oracle

Syntax

blErrors=*number*

Required Argument

number
specifies the number of errors that should be received before the load stops. The default is 1000000 errors.

blEscape= Data Connector Option

specifies the single-byte character to use for escape sequences for CSV files.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements

Default: " (double quotation mark)

Requirement: To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE and set blFormat=CSV.

Data source: PostgreSQL

Syntax

blEscape=*'single-character'* | 'OFF'

Required Argument

single-character
specifies the single-byte character to use for escape sequences for CSV files.

blExceptionTable= Data Connector Option

specifies the exception table into which rows in error are copied.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: none

Requirement: To specify this option, you must also set bulkLoad=TRUE.

Data source: DB2

Syntax

blExceptionTable=*"table-name"*

Required Argument

table-name
specifies the exception table into which rows in error are copied.

Details

Any row that is in violation of a unique index or a primary key index is copied. Db2 DATALINK exceptions are also captured in the exception table. If you specify an unqualified table name, the table is qualified with the current Db2 schema. Information that is written to the exception table is not written to the dump file. In a partitioned database environment, you must specify an exception table for those partitions on which the loading table is stored. However, the dump file contains rows that cannot be loaded because they are not valid or they contain syntax errors.

For more information, see the FOR EXCEPTION parameter in your Db2 documentation.

blFormat= Data Connector Option

specifies the format of the external table data: plain text (TEXT) or comma-separated values (CSV).

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements [Amazon Redshift, PostgreSQL], PROC CASUTIL: SAVE statement [Microsoft SQL Server]

Defaults: "DELIMITED" [Amazon Redshift, Microsoft SQL Server]

"TEXT" [PostgreSQL]

Restrictions: To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.

Microsoft SQL Server: This option is applicable only when bulk loading data to Microsoft Azure. For more information, see "Bulk Loading to Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse" on page 241.

Data source: Amazon Redshift, Microsoft SQL Server, PostgreSQL

Syntax

Form 1: Amazon Redshift:

blFormat="DELIMITED" | "PARQUET"

Form 2: Microsoft SQL Server:

blFormat="DELIMITED"

Form 3: PostgreSQL:

blFormat="TEXT" | "CSV"

Required Arguments

"DELIMITED" | "PARQUET"

for Amazon Redshift, specifies the format for intermediate data when bulk loading or bulk unloading. For Microsoft SQL Server, specifies the format of files being loaded to Azure Synapse Analytics (SQL DW).

Restriction The PARQUET value is available for Amazon Redshift only.

"TEXT" | "CSV"

for PostgreSQL, specifies the format of the external table data: plain text (TEXT) or comma-separated values (CSV).

Details

Amazon Redshift: Selecting PARQUET can lead to better performance for bulk loading and bulk unloading.

blIndexingMode= Data Connector Option

specifies the indexing scheme that the bulk-loading utility uses for index maintenance.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Alias: bl_indexing_mode=

Default: AUTOSELECT

Requirement: To specify this option, you must also set bulkLoad=TRUE.

Data source: DB2

Syntax

blIndexingMode=AUTOSELECT | REBUILD | INCREMENTAL | DEFERRED

Required Arguments

AUTOSELECT
specifies that the load utility automatically decides between REBUILD or INCREMENTAL mode.

REBUILD
specifies that all indexes are rebuilt.

INCREMENTAL
specifies that indexes are extended with new data.

DEFERRED
specifies that the load utility does not attempt index creation. Indexes are marked as needing a refresh.

Details

For more information about these choices, see your IBM Db2 documentation.

blLoad= Data Connector Option

specifies that, after the indicated number of rows is loaded, the load should stop.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Category: Bulk Loading

Data source: Oracle

Syntax

blLoad=*number-of-rows*

Required Argument

number-of-rows
specifies the number of rows that should be loaded. The first rows from the table are loaded. The default is to load all rows.

blLoadReplace= Data Connector Option

specifies whether the bulk-loading process replaces all rows in a table during bulk loading.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Alias: bl_load_replace=

Default: FALSE

Requirement: To specify this option, you must also set bulkLoad=TRUE.

Data source: DB2

Syntax

blLoadReplace=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether the bulk-loading process replaces all table rows during bulk loading. If set to FALSE, bulk loading appends new rows to an external table.

blLog= Data Connector Option

specifies the name and location of a log file that contains information for bulk loading.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Alias: bl_log=

Default: none

Requirement: To specify this option, you must also set bulkLoad=TRUE.

Data source: DB2

Syntax

blLog=*"path-and-log-file-name"*

Required Argument

path-and-log-file-name
specifies the name and location of a log file that contains information for bulk loading.

Details

When the DBMS bulk-load facility is invoked, it creates a log file. The contents of the log file are DBMS-specific. If blLog= is specified with the same path and file name as an existing log, the new log replaces the existing log.

blLogDir= Data Connector Option

specifies the location of the log file for the bulk-loading utility.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: blDefaultDir= value, if specified, or the location where temporary files are stored

Data source: Informix

Syntax

blLogDir=*'path'*

Required Argument

path
specifies the location of the log file for the bulk-loading utility.

Details

If you enable blEnableLog=, then use this option to specify where to store the resulting log file.

blNfxSdkPath= Data Connector Option

specifies the installation path of the Informix client SDK.

Valid in: CASLIB statement

Default: none

Requirement: This option is required when you set bulkLoad=TRUE or bulkUnload=TRUE.

Data source: Informix

See: bulkLoad=, bulkUnload= on page 441

Syntax

blNfxSdkPath=*"value"*

Required Argument

value
specifies the installation path of the Informix client SDK.

blNull= Data Connector Option

specifies the string that represents a null value.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements

Default: '\N' for TEXT files or unquoted empty value for CSV files

Restriction: To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.

Data source: PostgreSQL

Syntax

blNull='\N' | *empty-value*

Required Argument

'\N' | *empty-value*
specifies the string that represents a null value.

blNumDataFiles= Data Connector Option

specifies the number of data files to create to contain the source data.

Valid in: CASLIB statement, PROC CASUTIL: DELETESOURCE, LOAD, SAVE statements

Default: 2

Interaction: To specify this option, you must also set bulkLoad=TRUE.

Data source: Snowflake

See: bulkLoad= data connector option

Syntax

blNumDataFiles=*value*

Required Argument

value
specifies the number of data files to create to contain the source data. This value can be an integer between 1 and 100, inclusively.

blNumReadThreads= Data Connector Option

specifies the number of threads to use for bulk loading data from the data source into the SAS Viya platform.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Alias: bl_num_read_threads=

Default: none

Range: 0 – 10

Data source: Snowflake

Syntax

blNumReadThreads=*value*

Required Argument

value
specifies the number of threads to use for bulk loading data from the data source into the SAS Viya platform.

blobThreshold= Data Connector Option

specifies whether to allocate memory for large-object (LOB) columns based on the maximum length of those columns.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Default: 0 (do not perform a preliminary scan)

Range: 0 – 2,147,483,647

Data source: DB2

Syntax

blobThreshold=*value*

Required Argument

value
specifies whether to allocate memory for large-object columns based on the maximum length of those columns. A positive integer means that the data is scanned to determine the maximum length of large-object (LOB) columns, such as BLOB or CLOB columns. This value also functions as an upper limit to the allowed size of the LOB values. See Details for more information about how this value is used.

If you specify 0, then an alternative method is used to load the large-object data. This method ultimately allocates the same amount of data as when you specify a positive value. However, loading the data with this alternate method is not as rapid as the method that is used after a preliminary scan.

Details

This option can improve performance for data with large objects, such as BLOBs, CLOBs, LONGVARCHAR, LONGVARBINARY, and so on.

When you specify an integer larger than 0, the all LOB-type columns are scanned, and the maximum length is noted. If that maximum value is less than or equal to the value that you set for blobThreshold=, then the LOB-type columns are loaded as direct binds, and the size for those columns is set to the maximum value that was determined during the preliminary scan of the data.

blOptions= Data Connector Option

specifies options to pass to the bulk-load or bulk-unload facility.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements [Amazon Redshift, Snowflake], PROC CASUTIL: SAVE statement [DB2, Microsoft SQL Server]

Aliases: bl_options= [DB2]
blModifiedBy=, bl_modified_by= [DB2]
bulkLoadOptions= [Amazon Redshift, Microsoft SQL Server, Snowflake]

Default: none

Restriction: *Microsoft SQL Server:* This option is applicable only when bulk loading data to Microsoft Azure or Microsoft Fabric Data Warehouse. For more information, see "Bulk Loading to Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse" on page 241.

Requirements: *Amazon Redshift, Snowflake:* To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.
DB2, Microsoft SQL Server: To specify this option, you must also set bulkLoad=TRUE.

Data source: Amazon Redshift, DB2, Microsoft SQL Server, Snowflake

Example: blOptions="ROUNDEC"

Syntax

blOptions=*"option-1<, option-2<, ...>>"*

Required Argument

option-N
specifies an option to pass to the bulk-load and bulk-unload facility.

Details

These options affect how data is transferred and processed. These options are passed exactly as you enter them in the blOptions= string.

Amazon Redshift, Snowflake: For bulk loading, any options that you specify are appended to the COPY INTO command that is executed by the bulk-loading tool.

Microsoft SQL Server: This option specifies the options that you pass to the bulk-load facility. These options affect how data is transferred and processed in Azure or in Microsoft Fabric Data Warehouse. Any options that you specify are appended to

a COPY command as a WITH clause. You must specify option names as they are expected for the COPY command. For example, for a blOptions= string, you would specify FIELDTERMINATOR='*field_terminator*' rather than the delimiter= option.

blParallel= Data Connector Option

specifies whether to perform a parallel bulk load.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Category: Bulk Loading

Default: FALSE

Data source: Oracle

Syntax

blParallel=TRUE | FALSE

Required Arguments

TRUE
specifies that a parallel load should be performed.

FALSE
specifies that a parallel load is not performed. That is the default behavior.

blPSQLpath= Data Connector Option

specifies the full path to the PSQL executable file.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Default: psql

Restriction: To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.

Data source: PostgreSQL

Note: Support for this option was removed in 2025.03.

Syntax

blPSQLpath=<'>*path-name*<'>

Required Argument

path-name
specifies the full path to the PSQL executable file.

This option enables the data connector to call the PSQL utility.

Details

Starting in 2025.03, bulk loading and bulk unloading uses the COPY API for PostgreSQL. For this reason, this option, which works only with the PSQL utility, is no longer used.

blPortMax= Data Connector Option

specifies the highest available port number for concurrent uploads.

Valid in:	CASLIB statement, PROC CASUTIL: SAVE statement
Alias:	bl_port_max=
Default:	none
Requirement:	To specify this option, you must also set bulkLoad=TRUE.
Interaction:	To reserve a port range, specify values for this option and for blPortMin=.
Data source:	DB2
See:	blPortMin= on page 430, bulkLoad= on page 440

Syntax

blPortMax=*integer*

Required Argument

integer
specifies the highest available port number for concurrent uploads.

blPortMin= Data Connector Option

specifies the lowest available port number for concurrent uploads.

Valid in:	CASLIB statement, PROC CASUTIL: SAVE statement
Alias:	bl_port_min=

Default: none

Requirement: To specify this option, you must also set bulkLoad=TRUE.

Interaction: To reserve a port range, specify values for this option and for blPortMax=.

Data source: DB2

See: blPortMax= on page 430, bulkLoad= on page 440

Syntax

blPortMin=*integer*

Required Argument

integer
specifies the lowest available port number for concurrent uploads.

blQuote= Data Connector Option

specifies the quotation character for CSV files.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements

Default: " (double quotation mark)

Restriction: To specify this option, you must set bulkLoad=TRUE or bulkUnload=TRUE and set blFormat=CSV.

Data source: PostgreSQL

Syntax

blQuote=*'single-character'*

Required Argument

single-character
specifies the quotation character for CSV files.

blRecoverable= Data Connector Option

specifies whether a bulk-loading process is recoverable.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Alias: bl_recoverable= [DB2]

Default: FALSE

Requirement: To specify this option, you must also set bulkLoad=TRUE.

Interaction: When you enable this option, you must also provide a value for blCopyLocation= [DB2].

Data source: DB2, Oracle

See: blCopyLocation= [DB2] on page 412, bulkLoad= on page 440

Syntax

blRecoverable=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether a bulk-loading process is recoverable.

Details

DB2: When blRecoverable=FALSE, the UNRECOVERABLE keyword is added to the load command for the bulk-loading utility.

blRemoteFile= Data Connector Option

specifies the location of temporary files that are created by the bulk-loading utility.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Alias: bl_remote_file=

Default: none

Data source: DB2

Example: `blRemoteFile='/db2-server-location/subfolder'`

Syntax

blRemoteFile=<'>*path*<'>

Required Argument

path
specifies the full path to a server location that the bulk-loading utility uses to store temporary files that are created during bulk loading.

Details

Specify a location on the DB2 server that is accessed exclusively by a single DB2 server instance, and for which the instance owner has Read and Write permissions. Bulk loading to different databases must use different blRemoteFile= values.

The temporary files that are created in this location during bulk loading are deleted if the process completes without errors.

blSkip= Data Connector Option

specifies to skip the indicated number of rows before starting the bulk load.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Category: Bulk Loading

Default: 0

Data source: Oracle

Syntax

blSkip=*number-of-rows*

Required Argument

number-of-rows
specifies the number of rows to skip before beginning the load. The default is 0, which means that no records are skipped.

blSkipIdxMaint= Data Connector Option

specifies whether to perform index maintenance on the bulk load.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Category: Bulk Loading

Default: FALSE

Data source: Oracle

Syntax

blSkipIdxMaint=TRUE | FALSE

Required Arguments

TRUE
specifies to stop index maintenance on the load. This causes the index partitions that would have had index keys added to them to be marked Index Unusable. The index segment is inconsistent with the data it indexes. Index segments that are not affected by the load retain the Index Unusable state that they had prior to the load.

FALSE
specifies that index maintenance is performed on the load. This is the default value.

blSkipUnusableIdx= Data Connector Option

specifies whether to skip index entries that are in an unusable state and continue with the bulk load.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Category: Bulk Loading

Data source: Oracle

Syntax

blSkipUnusableIdx=TRUE | FALSE

Required Arguments

TRUE
specifies that the unusable index entry should be skipped. This is the default value.

FALSE
specifies that the unusable index entry should not be skipped.

Details

If an index in an Index Unusable state is encountered, by default, it is skipped and the load operation continues. This allows the SQL*Loader to load a table with indexes that are in an unusable state before beginning the load. Indexes that are not in an unusable state at load time are maintained by the SQL*Loader. Indexes that are in an unusable state at load time are not maintained and remain in an unusable state at load completion.

If this bulk load option is not specified, the default value is specified in the Oracle database configuration parameter, SKIP_UNUSABLE_INDEXES. This value is specified in the initialization parameter file. The BL_SKIP_UNUSABLE_INDEXES bulk load table option overrides the value of the SKIP_UNUSABLE_INDEXES configuration parameter in the initialization parameter file.

blStreamSize= Data Connector Option

specifies the size of the stream transfer buffer.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Category: Bulk Loading

Default: 256000

Data source: Oracle

Syntax

blStreamSize=*value*

Required Argument

value
specifies the size of the stream transfer buffer.

blWarningCount= Data Connector Option

specifies the maximum number of row warnings to allow before a load fails.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: 2147483646

Requirement: To specify this option, you must also set bulkLoad=TRUE and specify a value for blRemoteFile=.

Data source: DB2

See: blRemoteFile= on page 432, bulkLoad= on page 440

Syntax

blWarningCount=*value*

Required Argument

value
specifies the maximum number of row warnings to allow before a load fails.

Details

See the log file for information about why the rows generated warnings.

blWriteBuff= Data Connector Option

specifies the number of rows to buffer in memory before writing to your data source.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: 10000

Requirement: To use this option, you must also set bulkLoad=TRUE.

Data source: PostgreSQL

Syntax

blWriteBuff=*value*

Required Argument

value
specifies the number of rows to buffer in memory before writing to your data source.

Details

Use this option to improve performance for bulk loading to your data source. SAS/ACCESS adjusts this value automatically based on row size. Specifically, the number of rows * row size must not exceed 500MB.

branded_driver= Data Connector Option

specifies the type of driver that you are using with the Greenplum data connector.

Valid in: CASLIB statement

Default: none

Data source: Greenplum

Syntax

branded_driver=*"driver-name"*

Required Argument

driver-name
specifies the type of driver that you are using with the Greenplum data connector.

bucket= Data Connector Option

specifies the name of the cloud platform bucket to use when transferring data in bulk.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement, PROC CASUTIL: SAVE statement [Amazon Redshift, Snowflake]

Default: none

Requirement: The bucket name that you specify must already exist.

Interactions: *Amazon Redshift, Snowflake:* To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.

Google BigQuery: This option is required when you specify bulkUnload=TRUE.

Data source: Amazon Redshift, Google BigQuery, Snowflake

Syntax

bucket=*"bucket-name"*

Required Argument

bucket-name
specifies the name of the cloud platform bucket to use when writing or retrieving data in bulk.

Details

Amazon Redshift, Snowflake: The bucket= value specifies an Amazon S3 bucket.

Google BigQuery: The bucket= value specifies a Google Cloud Storage bucket.

bufferSize= Data Connector Option

specifies the buffer size that is used to send data to or receive data from SAS embedded processes.

Valid in:	CASLIB statement, PROC CASUTIL: LOAD and SAVE statements
Default:	1048576
Restriction:	This option applies only with parallel data transfer (dataTransferMode="parallel").
Requirement:	You must specify a value that is greater than 0.
Data source:	Hadoop, Spark, SPD Engine files, Teradata
Tip:	If you experience memory errors during data transfer, reduce the value of bufferSize=.

Syntax

bufferSize=*bytes* | "*n*B" | "*n*K" | "*n*M" | "*n*G" | "*n*T"

Details

If you set this value in PROC CASUTIL, then this value overrides a value that you set in a CASLIB statement. Increasing the size might result in better performance with a trade-off of increased memory usage.

You can specify this value as an integer (bytes) or with a suffix, such as "4M" to indicate four megabytes. Supported suffixes are "B" (bytes), "K" (kilobytes), "M" (megabytes), "G" (gigabytes), and "T" (terabytes). Suffixes are case insensitive.

bulk_api= Data Connector Option

specifies whether to use bulk loading to insert a large number of rows from CAS into an external table.

Valid in: CASLIB statement, PROC CASUTIL: LOAD, SAVE statements

Default: NO

Data source: Salesforce

Syntax

bulk_api=YES | NO

Required Argument

YES | NO
specifies whether to use bulk loading to insert a large number of rows from CAS into an external table.

bulk_refresh= Data Connector Option

specifies how often, in milliseconds, bulk load requests are checked for completion.

Valid in: CASLIB statement, PROC CASUTIL: LOAD, SAVE statements

Default: 1000 (1 second)

Interaction: The bulk_refresh= and bulk_timeout= values reset to their default values if bulk_refresh= is greater than bulk_timeout=.

Data source: Salesforce

Syntax

bulk_refresh=value

Required Argument

value
specifies how often, in milliseconds, bulk load requests are checked for completion.

bulk_timeout= Data Connector Option

specifies the timeout period for bulk-load operations in milliseconds.

Valid in: CASLIB statement, PROC CASUTIL: LOAD, SAVE statements

Default: 90000 (90 seconds)

Interaction: The bulk_refresh= and bulk_timeout= values reset to their default values if bulk_refresh= is greater than bulk_timeout=.

Data source: Salesforce

Syntax

bulk_timout=value

Required Argument

value
specifies the timeout period for bulk-load operations in milliseconds.

bulkLoad= Data Connector Option

specifies whether to use bulk loading to insert a large number of rows from CAS into an external table.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Defaults: FALSE [Amazon Redshift, DB2, Informix, Microsoft SQL Server, MySQL, Oracle, PostgreSQL, SAP IQ, SingleStore standard]

TRUE [Google BigQuery, Hadoop, Impala, ORC, Spark]

none [Snowflake]

Restrictions: *Hadoop:* Support for this option applies only when dataTransferMode='serial'.

Microsoft SQL Server: This option is applicable only when bulk loading data to Microsoft Azure or Microsoft Fabric Data Warehouse.

Requirement: *Spark:* When the Spark data connector is used to connect to Databricks, set bulkLoad=false.

Interactions: *Informix:* When you set bulkLoad=TRUE, you must also specify a value for blNfxSdkPath=.

Snowflake: When you set bulkLoad=TRUE, you must also set numWriteNodes=0 (use all nodes).

Data source: Amazon Redshift, DB2, Google BigQuery, Hadoop, Impala, Informix, Microsoft SQL Server, MySQL, Oracle, ORC, PostgreSQL, SAP IQ, SingleStore standard, Snowflake, Spark

Tips: *Hadoop:* You can use bulkLoad=false where HDFS does not exist, is not easily accessible, or is not allowed within your configuration, such as with certain security configurations.

ORC: Use bulkLoad=FALSE where HDFS does not exist, is not easily accessible, or is not allowed within your configuration, such as with delimited, ORC, or Parquet files.

See: blNfxSdkPath=, dataTransferMode=, numWriteNodes=

Syntax

bulkLoad=TRUE | FALSE

Details

DB2: Bulk loading to DB2 uses the CLI LOAD interface.

Informix: Bulk loading to Informix uses the DBACCESS utility. All available CAS nodes are used by default.

Microsoft SQL Server: Data is moved from CAS into Azure Synapse Analytics (SQL DW) or Microsoft Fabric Data Warehouse. Data is loaded using an Azure Data Lake Storage Gen2 account and the COPY INTO command. For more information, see "Bulk Loading to Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse" on page 241.

Spark: Bulk loading is enabled by default for Spark connections that have HDFS access. When HDFS access is not available, you must set bulkLoad=false. Bulk loading is not available for Databricks in the Spark data connector; therefore, you must set bulkLoad=false for Databricks connections.

bulkUnload= Data Connector Option

specifies whether to retrieve (fetch) a large number of rows from an external table into CAS.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Default: FALSE

Interactions: *Informix:* When you set bulkUnload=TRUE, you must also specify a vale for blNfxSdkPath=.

Amazon Redshift: Bulk unloading works only when bulkUnload= is enabled and when numReadNodes= is not set to 1.

Google BigQuery: When bulkUnload=TRUE, the bucket= option is required.

SAP IQ: When you set bulkUnload=TRUE, you must also specify values for clientDir= and serverDir= options. In addition, the numReadNodes=, sliceExpressions=, and sliceColumn= options are ignored when bulkUnload=TRUE.

Data source: Amazon Redshift, Google BigQuery, Informix, PostgreSQL, SAP IQ, Snowflake

See: blNfxSdkPath=, bucket=, clientDir=, numReadNodes=, serverDir=, sliceColumn= on page 577, sliceExpressions=

Syntax

bulkUnload=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether to retrieve (fetch) a large number of rows from an external table into CAS.

catalog= Data Connector Option

specifies a logical catalog name.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements [Amazon Redshift, DB2, Greenplum, Impala, Microsoft SQL Server, MongoDB, Netezza, ODBC, PostgreSQL, SAS Federation Server]

Default: Active caslib

Requirements: *Microsoft SQL Server:* This option is required for Microsoft SQL Server connections, specified as `catalog="*"`.

Netezza: This option is required for Netezza connections, specified as `catalog="*"`.

Data source: Amazon Redshift, DB2, DuckDB, Greenplum, Hadoop, Impala, Informix, JDBC, Microsoft SQL Server, MongoDB, Netezza, ODBC, PostgreSQL, SAP IQ, SAS Federation Server, Teradata, Vertica

Note: Support for DuckDB was added in 2026.09.

Syntax

Form 1: Most data sources

catalog=*"catalog-name"*

Form 2: Microsoft SQL Server and Netezza, and optionally, ODBC and SAS Federation Server

catalog=*"*"*

Required Arguments

catalog-name
specifies a logical catalog name.

*
specifies that data source supports multiple native databases.

Note This value is available to Microsoft SQL Server, Netezza, ODBC, and SAS Federation Server.

Details

The logical name can be any user-defined name. This name is displayed in the Catalog column for all tables in the results from the CONTENTS statement in PROC CASUTIL.

ODBC, SAS Federation Server: When you use the data connectors for ODBC or SAS Federation Server to connect to data sources that natively support multiple catalogs, such as Microsoft SQL Server or Netezza, specify `catalog="*"`. Specifying `catalog="*"` lets all catalogs be referenced.

charMultiplier= Data Connector Option

specifies the number of bytes per character of fixed-byte-width character columns.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement [SPD Engine files], PROC CASUTIL: LOAD and SAVE statements [all data sources except SPD Engine files]

Default: 1

Range: 1–5

Restrictions: This option applies to data that is encoded using a single-byte character set.

Hadoop, Spark: Because all Hive tables are UTF-8 tables, this value does not affect Hive.

Data source: Amazon Redshift, DB2, DuckDB, Greenplum, Hadoop, Impala, Informix, JDBC, Microsoft SQL Server, MongoDB, MySQL, Netezza, ODBC, Oracle, PostgreSQL, Salesforce, SAP HANA, SAP IQ, SAS data sets, SAS Federation Server, SingleStore standard, Snowflake, Spark, SPD Engine files, Vertica, Yellowbrick

Note: Support for DuckDB was added in 2026.09.

Syntax

charMultiplier=*value*

Required Argument

value
specifies the number of bytes per character of fixed-byte-width character columns.

Details

The number of bytes that are needed for multibyte characters depends on the characters in a string. For double-byte character sets, set charMultiplier=2.0.

When you specify this value for a PROC CASUTIL statement, the value overrides a value of charMultiplier= that was set in a CASLIB statement.

class= Data Connector Option

specifies the JDBC driver class name.

Valid in:	CASLIB statement
Alias:	driverClass=
Default:	none
Requirement:	This option is required in the CASLIB statement.
Data source:	JDBC

Syntax

class=*"class-name"*

Required Argument

class-name
specifies the JDBC driver class name.

classPath= Data Connector Option

specifies additional resources, including JDBC drivers or XML files, to add to the JDBC class path.

Valid in:	CASLIB statement
Default:	none
Data source:	Hadoop, JDBC, Spark

Syntax

classPath=*"directory-path"*

Required Argument

directory-path
specifies a path beneath the **/data-drivers/** folder path.

Details

The classPath= data connector option can be used to specify an alternate class path if you need a different class path for some CASLIB statements.

Here is an example that shows how this option might be used:

```
caslib cdatalib desc='CData caslib'
   dataSource=(srctype='hadoop',
               username='myuser1',
               password='mypwd',
               classpath='/data-drivers/cdata'
               driverclass='cdata.jdbc.apachehive.ApacheHiveDriver'
               server='hiveServer'
               schema='<Hive-schema-name>'
               hadoopJarPath=<Hadoop_jar_path_directory>',
               hadoopConfigPath='<Hadoop_config_path_directory>')
;
```

This example specifies a class path of **/data-drivers/cdata**. This folder contains the CData JDBC driver for Apache Hive. The driverClass= option specifies to load the CData JDBC driver.

Most users will not need this option. The location **/data-drivers/jdbc** is the reserved location for JDBC drivers that are obtained by users. In the SAS Viya Kubernetes environment, it is common to make the **/data-drivers/jdbc** mount point available to CAS via a Patch Transformer.

client_encoding= Data Connector Option

specifies the encoding that the data source client uses to transmit CHAR and VARCHAR data.

Valid in: CASLIB statement

Defaults: UTF-8 [Amazon Redshift, Impala, Teradata]
none [DB2, Informix, ODBC, Snowflake, Vertica]
current SAS session encoding [DuckDB]
Shell encoding [Microsoft SQL Server, PostgreSQL, Yellowbrick]

Restriction: Teradata: This option applies only to serial data transfer (dataTransferMode="serial").

Data source: Amazon Redshift, DB2, DuckDB, Impala, Informix, Microsoft SQL Server, ODBC, PostgreSQL, Snowflake, Teradata, Vertica, Yellowbrick

Note: Support for DuckDB was added in 2026.09.

Syntax

client_encoding=*"encoding"*

Required Argument

encoding
specifies the encoding that the data source client uses to transmit CHAR and VARCHAR data.

Details

Typically, the client encoding value corresponds to the LANG environment variable. However, the client_encoding= value is independent of the NLS_LANG environment variable. For encodings other than UTF-8, the data is transcoded into UTF-8 when the data is loaded into CAS.

Amazon Redshift, Impala, Informix, Microsoft SQL Server, ODBC, PostgreSQL: If you have configured the underlying ODBC driver to use a different encoding than the LANG environment variable, set client_encoding= to the corresponding encoding value.

DB2: If you have configured the DB2 client to use a different encoding than the LANG environment variable, set client_encoding= to the corresponding encoding value.

Teradata: If you have configured the Teradata client to use a different encoding than the LANG environment variable, set client_encoding= to the corresponding encoding value.

client_id= Data Connector Option

specifies the client ID value for the OAuth client that is used to connect to Google BigQuery.

Valid in: CASLIB statement

Category: Authentication

Default: none

Interaction: Use this option in conjunction with the client_secret= and refresh_token= options.

Data source: Google BigQuery

See: client_secret=, refresh_token=

Syntax

client_id=*"ID-value"*

Required Argument

ID-value
specifies the client ID value for the OAuth client that is used to connect to Google BigQuery using the OAuth authentication method.

client_secret= Data Connector Option

specifies the client secret value for the OAuth client that is used to connect to Google BigQuery.

Valid in: CASLIB statement

Category: Authentication

Default: none

Interaction: Use this option in conjunction with the client_id= and refresh_token= options.

Data source: Google BigQuery

See: client_id=, refresh_token=

Syntax

client_secret=*authentication-secret*

Required Argument

authentication-secret
specifies the client secret value for the OAuth client that is used to connect to Google BigQuery using the OAuth authentication method.

clientDir= Data Connector Option

specifies the location that contains the data files for bulk loading.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: none

Requirement: The location that you specify must be one that all CAS nodes have permission to write to.

Data source: SAP IQ

See: serverDir= data connector option

Syntax

clientDir=*"directory"*

Required Argument

directory
specifies the directory path that contains the data files for bulk loading to the data source.

Details

This option specifies the location of the data files that contain the table rows to load to the data source. This location is typically an NFS directory, and it must be Write accessible to all CAS nodes.

The location that you specify for clientDir= is the same location as that specified for serverDir=. However, clientDir= specifies the path that is used by the CAS nodes to access the location. The serverDir= option specifies the path that the data source server uses to access the data file location. The values for clientDir= and serverDir= might look different, depending on how the file systems are mounted.

clientSecret= Data Connector Option

specifies the client secret for authentication to Azure ADLS Gen2 storage.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: none

Requirement: To specify this option, also set bulkLoad=TRUE.

Interaction: When you use this option, also set the applicationID= option.

Data source: Microsoft SQL Server

Note: Support for this option was added in 2025.08.

See: applicationId=, bulkLoad=

Example:
```
clientSecret='aaa0B~c~d1efGHiJklm2N3o4pQRs5TUVw6-7xyzA'
```

Syntax

clientSecret=*'value'*

Required Argument

value
specifies the client secret for authentication to Azure ADLS Gen2 storage.

Details

Use this option to authentication to Azure ADLS Gen2 storage for bulk loading data to Fabric Data Warehouse or Azure Synapse. When you specify this option, authentication uses your client ID and the client secret that you provide with this option. Provide the client ID with the applicationID= data connector option.

The value that you specify is blotted in the SAS log.

To obtain the client secret value in the Azure portal:

1 Search for your application ID value.

Note: This value should be added to the Storage Blob Data Contributor role to ensure access to the storage account.

2 Click on **Certificates & secrets** and create a new secret. Save this value and provide it as your client secret.

clusterId= Data Connector Option

specifies the cluster ID for a connection.

Valid in:	CASLIB statement
Category:	Data Access
Alias:	cluster_id [Amazon Redshift]
Default:	none
Requirements:	*Amazon Redshift:* This option is required for CASLIB connections. *Spark:* The platform= data connector option is specified.
Data source:	Amazon Redshift, Spark
Notes:	*Spark:* This option is supported for Databricks only. Support for Amazon Redshift was added in 2025.08.

Syntax

clusterId="*cluster-ID*"

Required Argument

cluster-ID
specifies the cluster ID for your connection.

Details

Amazon Redshift: This value specifies the Amazon Redshift cluster ID.

Spark: This value specifies the Databricks cluster ID.

compress= Data Connector Option

specifies whether to compress data or specifies the compression method to use when you save an in-memory table.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement [Amazon Redshift], PROC CASUTIL: SAVE statement, saveTable action: importOptions= [ORC]

Defaults: FALSE [Amazon Redshift, Microsoft SQL Server, SAP IQ, Snowflake]
NO [SAS data sets, SPD Engine Files]
ZLIB [ORC]

Restrictions: *Microsoft SQL Server:* This option applies only when bulk loading data to Microsoft Azure. For more information, see "Bulk Loading to Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse" on page 241.
SPD Engine files: This option applies to serial data transfer only.

Requirements: *Amazon Redshift:* To specify this option, you must set bulkLoad=TRUE or bulkUnload=TRUE.
Microsoft SQL Server: To specify this option, you must set bulkLoad=TRUE and blFormat=DELIMITED.

Interaction: *SAS data sets:* Use this option with the reuse= option.

Data source: Amazon Redshift, Microsoft SQL Server, ORC, SAP IQ, SAS data sets, Snowflake, SPD Engine Files

Note: Support for SAP IQ was added in 2023.06.

Syntax

Form 1: Amazon Redshift, Microsoft SQL Server, SAP IQ, Snowflake:

compress=TRUE | FALSE

Form 2: ORC:

compress="YES" | "NO" | "ZLIB"

Form 3: SAS data sets, SPD Engine Files:

compress="YES" | "NO" | "BINARY" | "CHAR"

Required Arguments

TRUE | FALSE
for Amazon Redshift, Microsoft SQL Server, SAP IQ, and Snowflake, specifies whether to compress data using the GZIP format.

"YES" | "NO" | "ZLIB"
for ORC, specifies whether to compress or the compression method to use for the data set when you save an in-memory table as a SAS data set.

"YES" | "NO" | "BINARY" | "CHAR"
for SAS data sets or SPD Engine Files, specifies whether to compress or the compression method to use for the data set when you save an in-memory table as a SAS data set.

Details

Microsoft SQL Server: If set to TRUE, data is compressed by the COPY INTO command using the GZIP compression method.

connCatalog= Data Connector Option

specifies the Unity Catalog name to use to qualify names when listing database objects.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Default: none

Data source: Spark

Note: Support for this option was added in 2026.02.

Syntax

connCatalog=*"catalog-name"*

Required Argument

catalog-name
specifies the Unity Catalog name to use to qualify names when listing database objects. If your organization has enabled Databricks Unity Catalog, you can use this option to specify the Unity Catalog.

Example: Specify the Databricks Unity Catalog

If your organization has enabled Databricks Unity Catalog, you can specify the catalog in the connCatalog= option in the Spark CASLIB statement.

```
caslib x datasource=(srctype="spark"
       platform="databricks"
       uid=token
       pwd="my-password"
       connCatalog="main"
       schema="default"
       bulkload=no
       server="my-server.azuredatabricks.net"
       port=443
       useSsl=yes
       httpPath="my-http-path"
);
```

connectionTypeForCreate= Data Connector Option

specifies whether to use data types from PROC FEDSQL or from those that are native to your data source.

Valid in:	CASLIB statement, PROC CASUTIL: SAVE statement
Default:	fedsql
Interaction:	Use this data connector option to determine the available data types that you can specify for the dbType= data connector option.
Data source:	Microsoft SQL Server
See:	dbType= data connector option

Syntax

connectionTypeForCreate="fedsql" | "native"

Required Argument

fedsql | native

specifies whether to use FedSQL data types or data types that are native to your data source. When you select a native connection type, you can specify any data type that is supported by your data source. However, it is recommended that you specify data types that can be read back into CAS whenever possible.

Details

Use This Option with Care

The connection type that you specify applies to all columns that are saved to your data source, including any that are not specified for the dbType= option. To determine the list of default data type conversions, see the data types that are supported for your data source for either FedSQL or for the LIBNAME engine (native) for your data source.

When you specify connectionTypeForCreate="native", you can save data to any data type that is valid in your data source. Assign a data type for all data types in your CAS data that are not compatible with your database data types. It is recommended that you specify only data types that can be loaded back into SAS. This enables you to load data into SAS for additional analyses.

You should use caution when using the dbType= and connectionTypeForCreate= data connector options. With these options, there is the potential for data conversion errors (due to values being out of range for your selected data type) or for loss of precision when working with numeric values. For more information, see "Use This Option with Caution" on page 474.

Additional Data Type Support for Microsoft SQL Server

When you specify connectionTypeForCreate="native", all data types that you assign to variables with the dbType= data connector option must be of a native type. In addition, all columns that you write, including those that you do not specify for dbType=, are assigned the default data type that is used for a native connection. You can see the list of default data type conversions for a native connection in "Microsoft SQL Server: LIBNAME Statement Data Conversions" in *SAS/ACCESS for Relational Databases: Reference*.

When you set connectionTypeForCreate="native", you can save very large character values to your Microsoft SQL Server data source. Character values are saved to the external data source using the following criteria.

Character data that is stored as a WCHAR in CAS is written to Microsoft SQL Server as follows:

- values that have 4000 characters or more are written as NVARCHAR(MAX).
- values with fewer than 4000 characters are written as NCHAR(*n*).

Character data that is stored as a CHAR in CAS is written to Microsoft SQL Server as follows:

- values that have 8000 characters or more are written as VARCHAR(MAX). The maximum length for VARCHAR(MAX) is 2GB.
- values with fewer than 8000 characters are written as VARCHAR(*n*).

The data conversions for characters differ from the default data conversions that occur with the LIBNAME engine, because the data connector enables you to save larger character values. All character variables are saved to Microsoft SQL Server using the rules above, including character values that are not explicitly listed for the dbType= data connector option.

conOpts= Data Connector Option

specifies optional connection options that you pass to the DBMS.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Default: none

Data source: Amazon Redshift, DB2, DuckDB, Greenplum, Hadoop, Impala, Informix, Microsoft SQL Server, MongoDB, MySQL, Netezza, ODBC, Oracle, PostgreSQL, Salesforce, SAP HANA, SAP IQ, SAS Federation Server, SingleStore standard, Snowflake, Spark, SPD Engine files, Vertica, Yellowbrick

Note: Support for DuckDB was added in 2026.09.

Examples: For a PostgreSQL (or other DBMS) data source:

```
conOpts="QueryTimeout=0;UseDeclareFetch=1;FetchSize=100000"
```

For a SAS Federation Server data source:

```
conopts="(dsn=ora_dsn)"
```

Syntax

conOpts=*"connection-options"*

Required Argument

connection-options
specifies optional connection options that you pass to the DBMS. Separate multiple values with a semi-colon (;) within the outer quotation marks. For SAS Federation Server, this option specifies the data source name (DSN) to connect to, enclosed in parentheses.

Details

Connection options that you specify in statements other than the CASLIB statement override any connection options that were set when a caslib was added.

Microsoft SQL Server, PostgreSQL, Snowflake: For details about using this option with single sign-on to Microsoft Azure, see "Authenticate to Microsoft Azure with Azure AD" for Microsoft SQL Server, PostgreSQL, or Snowflake.

copyOptions= Data Connector Option

specifies options to pass to the COPY command.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Alias: copy_options=

Default: none

Data source: DuckDB

Note: Support for this option was added in 2026.09.

Syntax

copyOptions="*option1*<, *option2*<, ...>>"

Required Argument

option
specifies options to pass to the COPY command. Separate multiple options by commas, and enclose the entire string in quotation marks.

createMultipassColumn= Data Connector Option

specifies when to create the multipass column for multipass processing.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Default: If CAS added the multipass column (with the option saveMultipassColumn=TRUE), then the default is "onLoad". Otherwise, the default is "onDemand".

Interaction: Use the related option saveMultipassColumn=TRUE in order to permanently add a multipass column.

Data source: SAS SpeedyStore

Syntax

createMultipassColumn="onDemand" | "onLoad"

Required Arguments

onDemand
specifies to create the multipass column on demand.

onLoad
specifies to create the multipass column at table load time.

Details

The setting createMultipassColumn="onLoad" can improve performance if your analysis uses multiple passes against a table, especially for a promoted (global) table.

The multipass column is a columnstore sort key that uses the SingleStore AUTO_INCREMENT feature. The multipass column improves the performance of parallel Read operations by reducing the memory footprint. The feature enables CAS to maintain a sort key across multiple reads.

Here are some details about the behavior:

- Use the related option saveMultipassColumn=TRUE in order to permanently add a multipass column. CAS also adds an indicator in the column comments to signify that CAS added the column and it can be trusted. In this case, the createMultipassColumn= default is "onLoad" for that table. Whether createMultipassColumn= is set to "onLoad" or "onDemand", the existing multipass column is trusted and is used.
- If a multipass column exists in a table, but it was not added by CAS, here is the behavior:
  - If createMultipassColumn="onLoad", then the existing multipass column is trusted and is used.
  - If createMultipassColumn="onDemand", then processing is blocked with an error message when an action requires multiple passes.
- If no multipass column exists in the table, the default is "onDemand". Here is the behavior:
  - If createMultipassColumn="onLoad", CAS creates a temporary copy of the table as it is loaded, in order to add a multipass column. For a session table, the temporary copy with the multipass column exists until the session ends. For a promoted table, the temporary copy with the multipass column exists until you drop the table or until the server shuts down.
  - createMultipassColumn="onDemand", CAS does not create the temporary copy at load time. Instead, CAS creates the temporary copy and adds the multipass column when an action requires multiple passes. The temporary copy becomes the current loaded copy of the table and exists as long as the original table is loaded. For a promoted table, more than one session can access the same table concurrently. Therefore, several sessions could each create a temporary copy.
- When you edit a table, a temporary *snapshot* copy of the table is created. In this case, a multipass column is added in the snapshot copy. Examples of editing are

removing or appending rows. For details about the behavior with SingleStore views, see “createViewSnapshot= Data Connector Option” on page 459.

Avoid programming practices outside CAS that add or modify an AUTO_INCREMENT column, due to the following details:

- When you set createMultipassColumn="onLoad", you specify to trust and use an existing AUTO_INCREMENT column as the multipass column. The column is trusted even if you did not create it by specifying saveMultipassColumn=TRUE.
- If the AUTO_INCREMENT column is modified outside CAS, then CAS cannot guarantee that the column contains unique values.
- If CAS did not create the column, or if the column is modified outside CAS, then you must take steps to ensure that the column contains unique values. If duplicate values are present, then multipass analytics might not return correct results.

Here is some guidance to optimize performance:

- Determine how often multipass processing is needed for the tables in a caslib. Multipass processing is common in analytics. If many tables in a caslib need multipass processing, then set createMultipassColumn="onLoad" for the caslib.
- Note that the "onLoad" setting can reduce performance for tables that do not need multipass processing. You can choose to set createMultipassColumn= in the LOAD statement of the CASUTIL procedure (or in the table.loadTable action) for a table. Setting the option for a table overrides the setting in the caslib.
- Some examples of single-pass processing include basic DATA steps, the summary action, and the fetch action. However, your single-pass processing step could be followed by multipass processing on the same table.

Comparisons

The data connector option saveMultipassColumn= affects a table when it is saved. The option createMultipassColumn= affects a table when it is loaded or when an action requires multiple passes, depending on the setting.

Example: Saving Multipass Columns to a Table in SingleStore

This example demonstrates how to save a multipass column, also known as an auto_increment column, to a table in SingleStore.

```
/* Specify a host and port that are valid for your site.*/
options cashost="cloud.example.com" casport=5570;

/* If not already done, start session Casauto. */
cas casauto;
```

```
proc casutil;                                          /*1*/
   load data=sashelp.iris
   casout="iris"
   outcaslib="casuser"
   replace;
run;

proc cas;                                              /*2*/
table.save /
table={name="iris,
      caslib="casuser"},
      name="irisCopy.sashdat",
      caslib="singlestorelib",
      replace=True;
      datasourceoptions={srctype='singlestore',
                         saveMultipassColumn=true}     /*3*/
;
quit;
```

1 The CASUTIL procedure loads the iris data file from the SASHELP library and saves it into memory as a table named iris.

2 The table.save action saves the table as a file named irisCopy.sashdat.

3 Specify the data source options:

srctype=singlestore
: specifies SingleStore as the data source.

saveMultipassColumn=true
: specifies that CAS saves a multipass column in the table, and CAS adds an indicator in the column comments to signify that CAS added the column, and it can be trusted.

  In order to perform multipass CAS actions in SingleStore, a shadow copy with an additional multipass/auto-Increment column of target input tables is made either during the multipass action (createMultipassColumn=onDemand), or during the load of that table (createMultipassColumn=Onload). Creating the shadow copy might take some time. Depending on your workflow, it might be appropriate to generate the table with the multipass/auto-increment column already added as a pre-step one time, prior to your analytical workflow.

  Upon completion in the SingleStore database, you will see the table with an additional column: `rowId_unique_id`. However, in CAS this added auto-increment is a hidden column that is utilized to preserve stable ordering during a multi-pass action.

See Also

“saveMultipassColumn= Data Connector Option” on page 563

createViewSnapshot= Data Connector Option

specifies whether to stream a view or create a snapshot copy of a view result set.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Default: never

Interactions: If you load a view with backingStore="CASDISKCACHE", the view result set is copied to CAS, regardless of the value of the createViewSnapshot= option. If createViewSnapshot="onLoad", a temporary snapshot copy of the view result set is created in the database first, and it is deleted when the contents of the database snapshot are copied to CAS.

If you run a multipass action with multipassMemory="cache", and the input data is a view loaded with createViewSnapshot="never", the CAS Disk Cache must have enough space to hold the data to be analyzed.

If you run a multipass action with multipassMemory="stream", and the input data is a view loaded with createViewSnapshot="never", the code does not run. An error is printed in the log.

Data source: SAS SpeedyStore

Syntax

createViewSnapshot="onLoad" | "never"

Required Arguments

onLoad
specifies to create a snapshot copy in the SingleStore database of a view result set when a view is loaded.

never
specifies to not create a snapshot copy of a view result set, and to stream views instead.

Details

Overview

By default the data connector streams a view. Under the default value of createViewSnapshot="never", the data connector does not create a snapshot copy of the view result set.

Here are some details about views, whether you stream a view or create a snapshot copy of a view result set:

- If the view or the snapshot copy is saved to SingleStore by using the table.save action, the view or copy is saved as a table and not as a view. In addition, the table.save action does not allow you to specify a target that is an existing SingleStore view and overwrite the view. Even if you specify replace=true in the invocation of table.save, an error is returned.
- CAS stores column attributes such as SAS formats and labels and certain data type attributes in the column comments when you save a table to SingleStore. The SingleStore definition of a view does not include column comments. Therefore, if the table that underlies a view was created by CAS, then any column attributes that are stored in column comments are not part of the view definition when the view is loaded into CAS.

Streaming Views with createViewSnapshot="never"

When createViewSnapshot="never", a view is streamed with parallel Read or serial Read. CAS determines at load time whether parallel Read will be possible for subsequent actions that Read the view. If CAS determines that subsequent actions will perform a serial Read, a note is printed in the SAS log. Below are some examples of views that are not suitable for parallel Read. (This is not a complete list.)

- SELECT * FROM <*table*> where <*table*> is a REFERENCE table
- SELECT * FROM <*table*> where <*table*> is an INFORMATION_SCHEMA system view
- A view in database A that references a table in database B
- SELECT with no FROM clause (or the FROM clause is "FROM DUAL")

If createViewSnapshot="never" and you submit code that requires a snapshot copy, the code does not run and an error is printed in the log. Here are some examples of code that requires a snapshot copy:

- multipass actions when multipassMemory="stream" (See "Example 2: Understanding an Error Caused by a Multipass Action" on page 463.)
- multipass actions when multipassMemory="cache" and the CAS Disk Cache does not have enough space to hold the data to be analyzed
- table edit actions such as table.deleteRows, table.update, and table.append

Creating Snapshot Copies with createViewSnapshot="onLoad"

If createViewSnapshot="onLoad" and you submit code that requests to load a view, a snapshot copy is made of the view result set, and an informational note is printed in the SAS log.

Here are some details about the snapshot copy of the view result set:

- The copy is a SingleStore table that is created in the same database as the view. For a session table, the copy exists until the table is dropped, or until its containing caslib is dropped, or until the session ends. For a promoted table, the copy exists until the table is dropped, or until its containing caslib is dropped, or until the server shuts down. For a promoted table, the copy is deleted by a background process, at a time when any actions currently using the table have completed.

- The database user who is associated with the view's caslib must have permission to create tables in the database that is associated with the caslib. Therefore, when createViewSnapshot="onLoad", CAS users cannot call loadTable on a SingleStore INFORMATION_SCHEMA view, because SingleStore users do not have permission to create new tables in INFORMATION_SCHEMA.
- The copy contains a multipass column for multipass processing.
- The copy can be modified with actions such as table.append, table.alterTable, table.deleteRows, and table.update.

You can refresh the copy of the view result set in the following ways:

- You can issue a dropTable action followed by another loadTable action. The table is unavailable until the loadTable action completes (and a new copy of the view result set has been created).
- If an action's table= parameter references an unloaded SingleStore view (a transient-scope view), and if createViewSnapshot="onLoad", the copy is refreshed. At the start of the action, CAS loads the view and creates a copy of the result set. At the end of the action, the copy is deleted.

Examples

Example 1: Reading a SingleStore View

This example demonstrates the streaming Read of a SingleStore view.

The example also highlights an important issue: if the view references a table that was created by CAS, the view does not retain certain column attributes from the table. As noted in the details above, CAS stores column attributes such as SAS formats and labels and certain data type attributes in the column comments when you save a table to SingleStore. The SingleStore definition of a view does not include column comments. Therefore, if the table that underlies a view was created by CAS, then any column attributes that are stored in column comments are not part of the view definition when the view is loaded into CAS.

In the first portion of the code, create a table in SingleStore named cars_out.

```
/*Specify a host and port that are valid for your site.*/
/*options cashost="cloud.example.com" casport=5570;*/
/*If not already done, create session Casauto.*/
/*cas casauto;*/

caslib singlestorelib                                          /*1*/
   datasource=(
      srctype="singlestore"
      database="my_database"
      host="svc-sas-singlestore-cluster-dml"
      pass="my_password"
      port="my_port_number"
      user="my_username"
      createViewSnapshot="never"
   )
```

```
;

proc casutil;                                                    /*2*/
   load data=sashelp.cars
   casout="cars_cas"
   replace;

proc casutil;                                                    /*3*/
   save casdata="cars_cas"
   casout="cars_out"
   outcaslib="singlestorelib"
   replace;
quit;

proc cas;                                                        /*4*/
   table.columninfo /
      table={name="cars_out", caslib="singlestorelib"};
run; quit;
```

1 Assign a caslib to your SingleStore database.

2 Load the table cars from Sashelp as an in-memory table named cars_cas.

3 Save the table to SingleStore with the name cars_out.

4 List information about the columns of cars_out.

Column Information for cars_out in Caslib SINGLESTORELIB

Column	Label	Id	Type	Length	Formatted Length	Format	Format Width	Format Decimal
Make		1	char	13	13		0	0
Model		2	char	40	40		0	0
Type		3	char	8	8		0	0
Origin		4	char	6	6		0	0
DriveTrain		5	char	5	5		0	0
MSRP		6	double	8	8	DOLLAR	8	0
Invoice		7	double	8	8	DOLLAR	8	0
EngineSize	Engine Size (L)	8	double	8	12		0	0
Cylinders		9	double	8	12		0	0
Horsepower		10	double	8	12		0	0
MPG_City	MPG (City)	11	double	8	12		0	0
MPG_Highway	MPG (Highway)	12	double	8	12		0	0
Weight	Weight (LBS)	13	double	8	12		0	0

In an external tool, use SQL such as the following code to create a view named cars_view in SingleStore. The view selects some of the columns from the table, and the WHERE clause filters the rows.

```
create view cars_view as
select Make, Model, MSRP, MPG_City, MPG_Highway from cars_out
where MSRP < (select avg(MSRP) from cars_out);
```

The following code shows that CAS can read the SingleStore view. However, the column labels and attributes are not displayed.

```
proc cas;
   table.columninfo /
      table={name="cars_view", caslib="singlestorelib"};
run;
quit;
```

Column Information for cars_view in Caslib SINGLESTORELIB

Column	Id	Type	Length	Formatted Length	Format Width	Format Decimal
Make	1	char	52	52	0	0
Model	2	char	160	160	0	0
MSRP	3	double	8	12	0	0
MPG_City	4	double	8	12	0	0
MPG_Highway	5	double	8	12	0	0

Example 2: Understanding an Error Caused by a Multipass Action

This example demonstrates an error caused when a multipass action cannot create a snapshot copy of the view result set. You cannot use multipass actions when createViewSnapshot="never" and multipassMemory="stream".

> **TIP** You can use multipass actions when createViewSnapshot="never". To avoid the error, you must set multipassMemory="cache", and the CAS Disk Cache must have enough space to hold the data to be analyzed.

Before you run this example, use the method in "Example 1: Reading a SingleStore View" on page 461 to save a table in SingleStore and create a view externally. Then reassign the caslib in order to set the `multipassMemory="stream"` option, which causes the error when you run the percentile.boxplot action.

```
caslib singlestorelib drop;

caslib singlestorelib
   datasource=(
      srctype="singlestore"
      database="my_database"
      host="svc-sas-singlestore-cluster-dml"
      pass="my_password"
      port="my_port_number"
      user="my_username"
```

```
        createViewSnapshot="never"
        multipassMemory="stream"
    )
;
proc cas;
   percentile.boxplot table='cars_view' inputs={'msrp' 'mpg_city'};
run;
quit;
```

The SAS log output below shows the error. To avoid this error, set multipassMemory="cache".

Example Code 8.1 *Log Output*

```
ERROR: A multipass operation could not be completed. To avoid this error, specify
       the multipassMemory="cache" data connector option and ensure that
       the CAS Disk Cache has sufficient size to hold the data to be analyzed.
```

Example 3: Specifying the Option in the loadTable Action

If you specify the createViewSnapshot= option when you load a view, the value takes precedence over the caslib value.

The following code references the same caslib and the same view as in "Example 1: Reading a SingleStore View" on page 461. However, the option createViewSnapshot="onLoad" prevents the data connector from streaming the view. Instead, the view is accessed by creating a snapshot copy of the view result set.

```
proc cas;
     table.loadTable /
         path="cars_view",
         caslib="singlestorelib",
         casout={name="cars_newview"},
         datasourceoptions={createViewSnapshot='onLoad'};
quit;
```

In the following SAS log output, the messages indicate the view is successfully read.

Example Code 8.2 *Log Output*

```
NOTE: Cloud Analytic Services made a copy of the result set from SingleStore
view cars_view.
NOTE: Cloud Analytic Services made the external data from cars_view available
as table CARS_NEWVIEW in caslib SINGLESTORELIB.
```

credentialsFile= Data Connector Option

specifies the path and name of the AWS credentials file to use for bulk loading data.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement [Amazon Redshift], PROC CASUTIL: SAVE statement

Category: Authentication

Default: none

Requirements: *Amazon Redshift:* To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.

Snowflake: To specify this option, you must also set bulkLoad=TRUE.

Data source: Amazon Redshift, Snowflake

Syntax

credentialsFile=*"path-and-filename"*

Required Argument

path-and-filename
specifies the path and name of the AWS credentials file to use for bulk loading data.

credentialsProfile= Data Connector Option

specifies the profile name to use when there is more than one profile in the AWS credentials file.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement [Amazon Redshift], PROC CASUTIL: SAVE statement

Default: none

Requirements: *Amazon Redshift:* To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.

Snowflake: To specify this option, you must also set bulkLoad=TRUE.

Data source: Amazon Redshift, Snowflake

Syntax

credentialsProfile=*"profile-name"*

Required Argument

profile-name
specifies the profile name to use when there is more than one profile in the AWS credentials file.

credFile= Data Connector Option

specifies the path and name of the Google Cloud Platform credentials file.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: none

Interactions: This option is required in the CASLIB statement, unless you specify the autheticationDomain= option.

You can use the authenticationDomain= option instead of the credFile= option. The autheticationDomain= option enables you to connect to the Google Cloud platform via a defined authentication domain that is managed in credentials services.

Data source: Google BigQuery

See: authenticationDomain=

Example: credfile='/u/authfiles/BigQuery/xxx-yyy-8e99c10a22537.json'w

Syntax

credFile=*'path-and-file-name'*

Required Argument

path-and-file-name
specifies the path and name of the Google Cloud Platform credentials file.

csvOptions= Data Connector Option

specifies options to pass to the read_csv function.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Alias: csv_options=

Default: none

Data source: DuckDB

Note: Support for this option was added in 2026.09.

Syntax

csvOptions="*option1*<, *option2*<, ...>>"

Required Argument

option
specifies options to pass to the read_csv function. Separate multiple options by commas, and enclose the entire string in quotation marks.

database= Data Connector Option

specifies the name of the database.

Valid in:	CASLIB statement, PROC CASUTIL: LOAD statement [all except DB2]
Alias:	DB= [DB2, DuckDB, MongoDB, Teradata]
Default:	none
Requirement:	This option is required for the CASLIB statement.
Interaction:	If you supply values for schema= and database=, the value for schema= is used to qualify a table name.
Data source:	Amazon Redshift, DB2, DuckDB, Greenplum, Informix, MongoDB, MySQL, Netezza, PostgreSQL, SAP IQ, SAS SpeedyStore, SingleStore standard, Snowflake, Teradata, Vertica, Yellowbrick
Note:	Support for DuckDB was added in 2026.09.

Syntax

database="*database-name*"

Details

When you specify a value for database=, the table name is qualified with the database name. For example, if you set database="myDB" and you want to access table Studydata, the table myDB.Studydata is accessed.

DuckDB: The value that you specify for DuckDB includes the path and name of a DuckDB database file (for example, **`database="/mymount/sample.db"`**).

dataPath= Data Connector Option

specifies the location in which to store the DPF files for the SPD Engine table.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: mdfPath= value

Data source: SPD Engine Files

Syntax

dataPath=*"DPF-file-path"*

Required Argument

DPF-file-path
specifies the location in which to store the DPF files for the SPD Engine table.

dataSource_dsn= Data Connector Option

specifies the data source name that you are accessing with the SAS Viya platform. The name of this option is specific to your data source.

Valid in: CASLIB statement

Default: none

Requirements: This option is required in the CASLIB statement.

All data sources except Netezza: Set the ODBCSYSINI or ODBCINI environment variable. If the ODBCSYSINI environment variable is set, it must point to the full path of the directory that contains the configured odbc.ini and odbcinst.ini files. If ODBCSYSINI is not set, set the ODBCINI environment variable to the full path of the directory that contains the configured odbc.ini file. The appropriate environment variable is typically set during SAS/ACCESS configuration.

Netezza: Set the NZ_ODBC_INI_PATH environment variable to the full path of the directory that contains the configured odbc.ini file. This environment variable is typically set during SAS/ACCESS configuration.

Data source: Amazon Redshift, Greenplum, Impala, Microsoft SQL Server, Netezza, ODBC, PostgreSQL, SAP IQ, Snowflake, Vertica, Yellowbrick

Example: redshift_dsn="*Amazon-Redshift-data-source-name*"

Syntax

***dataSource*_dsn="*datasource-name*"**

Required Arguments

dataSource
specifies the data source type that corresponds to the data source name that you are accessing. For example, specify this option as impala_dsn= for an Impala data source. For the full list of options names, see Details.

datasource-name
specifies the data source name (DSN).

Details

The option name that you use is determined by your data source. See the table below for the option name that you should use for your data connector. Data sources with more than one option name can use any of the names that are listed.

An administrator configures a data source for use with the SAS Viya platform, typically in an odbi.ini file.

Table 8.1 *Option Names for each Type of Data Source*

Data Source	Option Name
Amazon Redshift	redshift_dsn=
Greenplum	greenplum_dsn= gpl_dsn= [alias]
Impala	impala_dsn=
Microsoft SQL Server	sqlserver_dsn= database= [alias] db= [alias]
Netezza	netezza_dsn=
ODBC	odbc_dsn= database= [alias] db= [alias]
PostgreSQL	ptg_dsn=
SAP IQ	siq_dsn=

Data Source	Option Name
Snowflake	snowflake_dsn=
Vertica	vertica_dsn=
Yellowbrick	yellowbrick_dsn=

dataTransferMode= Data Connector Option

specifies the mode of data transfer.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements, loadTable action: importOptions= [SAS data sets]

Aliases: dataTransfer=
dtm=

Defaults: SERIAL [Hadoop, SPD Engine files, Teradata]
AUTO [SAS data sets]

Restrictions: Multinode Read is not allowed with the FedSQL execDirect action. The load proceeds with numReadNodes=1.
[Hadoop] Parallel data transfer does not support national characters in SAS names such as table names or column names.

Requirements: *Hadoop:* Parallel data transfer is available only when SAS In-Database Technologies for Hadoop is enabled for the SAS Viya platform. For more information, see *SAS In-Database Products: Deployment and Administration Guide*.
Teradata: Parallel data transfer is available only when SAS Data Accelerator (known as SAS In-Database Technologies for Teradata prior to 2026.08) is enabled for the SAS Viya platform. For more information, see *SAS In-Database Products: Deployment and Administration Guide*.

Interaction: For details of the interaction with numReadNodes= and numWriteNodes=, see "Specifics of the dataTransferMode= Option".

Data source: Hadoop, SAS data sets, SPD Engine files, Teradata

See: [Hadoop] "Parallel Data Transfer for Cloudera" on page 216

Example:
```
dataTransferMode="PARALLEL"
```

Syntax

dataTransferMode="AUTO" | "PARALLEL" | "SERIAL"

Required Arguments

AUTO
specifies to first try to load or save the data in parallel using embedded processing. If this fails, a note is written to the SAS log and serial processing is then attempted.

PARALLEL
specifies to load or save the data in parallel to your database or data source.

SERIAL
specifies to load or save the data serially to your database or data source.

Details

Specifics of the dataTransferMode= Option

If you specify this option outside of a CASLIB statement, this value overrides the value in a CASLIB statement.

If numReadNodes= or numWriteNodes= is a value other than 1 and dataTransferMode="AUTO", SAS attempts first to transfer the data in parallel. If parallel transfer fails, SAS attempts to transfer data with the data connector using the specified number of CAS worker nodes. If multiple nodes are not available, SAS transfers the data serially.

SAS data sets: Data is loaded serially if only the controller node has access to the SAS data set. To load data in parallel, the controller and all worker nodes must have access to the SAS data set.

Comparison with dataTransferMode= for File Import

The dataTransferMode= option that is part of a caslib definition is not the same as the dataTransferMode= file import option. Although they share the same name and values, they are functionally different and therefore are not interchangeable. See "Where to Specify Data Transfer Modes" for details.

dbCreateTableOpts= Data Connector Option

specifies any table options that should be appended to a CREATE TABLE statement that is passed to the data source.

Valid in:	PROC CASUTIL: SAVE statement
Default:	none
Restriction:	*Teradata:* This option applies to serial data transfer only.
Data source:	Amazon Redshift, DB2, DuckDB, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MongoDB, MySQL, Netezza, ODBC,

Oracle, PC Files, PostgreSQL, Salesforce, SAP HANA, SAP IQ, SAS Federation Server, Snowflake, Spark, Teradata, Vertica, Yellowbrick

Note: Support for DuckDB was added in 2026.09.

Syntax

dbCreateTableOpts=*"text"*

Required Argument

text
specifies table options that should be appended to a CREATE TABLE statement that is passed to the data source.

Details

Teradata: By default for Teradata, no primary indexes are assigned when you save a table. To specify a primary index, you can use this option:

```
options={dbCreateTableOpts="unique primary index(idnum)"}
```

dbMaxText= Data Connector Option

specifies the maximum length for all string data types.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Alias: dbMax_text=

Defaults: 1024 [Haddop, Spark]
32767 [JDBC]
32000 [Teradata]

Ranges: *Hadoop, Spark:* 1 (minimum), *Teradata:* 1–32767

Restriction: *Teradata:* This option applies to serial data transfer only.

Data source: Hadoop, JDBC, Spark, Teradata

Syntax

dbMaxText=*maximum-string-column-length*

Required Argument

maximum-string-column-length
specifies the maximum length for all string data types.

Details

String data is loaded as VARCHAR data in CAS. If you specify `dbMaxText=100`, all string data types are loaded as VARCHAR(100).

dbType= Data Connector Option

specifies alternative data types for specified columns.

Valid in:	CASLIB statement, PROC CASUTIL: SAVE statement
Default:	none
Restriction:	You must specify a valid column name for this option. If you specify a column name that is not part of the table that you are writing to your data source, an error is written in the log.
Interaction:	*Microsoft SQL Server:* Use the connectionTypeForCreate= data connector option to determine the list of valid data types to specify for your data source.
Data source:	DuckDB, Greenplum, Impala, JDBC, Microsoft SQL Server, MongoDB, MySQL, Oracle, PostgreSQL, Salesforce, SAP HANA, SingleStore standard, Snowflake, Yellowbrick
Note:	Support for DuckDB was added in 2026.09.
See:	connectionTypeForCreate= data connector option

Syntax

dbType=*"column1=<'>data-type1<'> <column2=<'>data-type2<'>...>"*

Required Argument

column=data-type
specifies one or more column names and corresponding data types that should be used when writing to a table in the data source. Separate multiple column and data type pairs with a space.

Details

Use This Option with Caution

Use caution when you use the dbType= data connector option. Using alternative data types instead of default data type conversions can result in loss of numeric precision or data conversion errors. Check your log carefully to ensure that all data was written successfully to your external data source.

Be careful if you use the dbType= option to specify an alternative numeric value when you save data to an external table. If a numeric variable is normally saved to a BIGINT data type, but you save it to an INTEGER type, precision for very large values is at risk. For more information, see the information about "Integer Data Types and Numeric Precision" for your data connector.

In addition, by selecting a data type conversion that differs from the default data conversion, it is possible that some values lie outside of the acceptable range for the data type that you select. For example, suppose that you have a variable that includes a 2000-character string, and you want to save that variable to a SMALLINT data type. Writing the rows to an external table is likely to fail and generate a data conversion error.

Further, how a data conversion failure is handled varies based on your data source. For some data sources, if an invalid value is encountered, all updated rows are removed from the external table (a rollback). Other data sources insert rows until an invalid value is encountered and stop updating the external table at that point. Neither case is desirable, since either no data or a subset of your data is written to the external table.

Note: If your data source does not roll back updates due to errors, the subset of data that is written to your external table can vary each time you attempt to write data with conversion errors to your data source. This is because writing to external tables typically uses multiple CAS nodes. This means that the number of rows written and the order of rows can differ each time.

Overview of Available Data Types

The available data types for your data source are based on those that are used for PROC FEDSQL. Use the links below to determine the data types that are available to your data source.

Table 8.2 *Links to Available Data Types*

Data Source	FedSQL Data Types
Greenplum	Data Types for Greenplum
Impala	Data Types for Impala
JDBC	Data Types for JDBC

Data Source	FedSQL Data Types
Microsoft SQL Server	Data Types for Microsoft SQL Server See also: "Additional Data Type Support for Microsoft SQL Server"
MongoDB	Supports:[1] ■ BIGINT ■ CHAR(*n*) ■ DOUBLE ■ INTEGER ■ TIMESTAMP ■ VARCHAR
MySQL	Data Types for MySQL
Oracle	Data Types for Oracle
PostgreSQL	Data Types for PostgreSQL
Salesforce	Data Types for Writing to Salesforce
SAP HANA	Data Types for SAP HANA
SingleStore standard	Data Types for SingleStore
Snowflake	Data Types for Snowflake
Yellowbrick	Data Types for Yellowbrick

1 The list of supported data types for dbType= is a subset of all FedSQL data types.

Additional Data Type Support for Microsoft SQL Server

There are additional data types that are supported for Microsoft SQL Server, including support for large character values. Support for Microsoft SQL Server includes the related connectionTypeForCreate= data connector option. When you set connectionTypeForCreate="native", you can specify any data type that is supported for Microsoft SQL Server. However, it is recommended that you specify data types that can be read back into SAS so that you can load data for additional calculations. For more information, see "connectionTypeForCreate= Data Connector Option" on page 452.

Example

In this example, you have a table called Cars_toyota in caslib Mycaslib. You want to save the table back out to your PostgreSQL data source, but you want to save the MPG_City and MPG_Highway columns as SMALLINT values instead of DOUBLES.

Use the dbType= option to change the default data type for these columns when you save the file to PostgreSQL. Use the output from the CONTENTS statement to verify that the data types were assigned correctly.

```
proc casutil incaslib=mycaslib;
  save casdata='cars_toyota' outcasib='mycaslib'
      casout='cars_subset'
      options=(dbtype="MPG_City='SMALLINT' MPG_Highway='SMALLINT'");
  contents casdata='cars_subset';
quit;
```

default_auth_plugin= Data Connector Option

specifies the authentication plugin to use when you connect to a MySQL compatible data source.

Valid in: CASLIB statement

Default: caching_sha2_password (for MySQL 8 client and higher)

Requirements: Specify the value for this option in lower case.

To connect to a SingleStore server in a MySQL CASLIB statement, set default_auth_plugin="mysql_native_password".

Data source: MySQL

Example:

```
caslib x datasource=(srctype="mysql" username="myuser" password="mypwd"
                     server="myserver.example.com" db="myDB"
                     default_auth_plugin="mysql_native_password");
```

Syntax

default_auth_plugin="caching_sha2_password" | "mysql_native_password"

Required Argument

"caching_sha2_password" | "mysql_native_password"
specifies the authentication plugin to use when you connect to a MySQL compatible data source.

defaultDir= Data Connector Option

specifies where all temporary files are created during bulk loading or bulk unloading.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements [Amazon Redshift, Google BigQuery], PROC CASUTIL: SAVE statement [Microsoft SQL Server, Snowflake]

Category: Bulk Loading

Alias: defaultDirectory=

Default: *temporary-file-directory* that is specified by the UTILLOC= system option

Requirements: *Amazon Redshift, Google BigQuery:* To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.

Microsoft SQL Server, Snowflake: To specify this option, you must also set bulkLoad=TRUE.

Interactions: *Microsoft SQL Server:* You use this option when you specify blFormat=DELIMITED.

Microsoft SQL Server: If you specify useLog=TRUE, then this location is used for the error log of rejected rows.

Data source: Amazon Redshift, Google BigQuery, Microsoft SQL Server, Snowflake

Syntax

defaultDir=*"directory-path"*

Required Argument

directory-path
specifies where all temporary files are created during bulk loading or bulk unloading.

Details

The value that you specify is prepended onto the file name. Provide the complete directory path, including the directory separator character.

Microsoft SQL Server: This option is applicable only when bulk loading data to Microsoft Azure. For more information, see "Bulk Loading to Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse" on page 241.

deleteDataFiles= Data Connector Option

specifies whether to delete only the data file or all files that are created during bulk loading or bulk unloading.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements [Amazon Redshift, Google BigQuery, Snowflake], PROC CASUTIL: SAVE statement [Microsoft SQL Serve, SAP IQ]

Default: TRUE

Requirements: *Amazon Redshift, Google BigQuery, SAP IQ, Snowflake:* To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.

Microsoft SQL Server: To specify this option, you must also set bulkLoad=TRUE.

Data source: Amazon Redshift, Google BigQuery, Microsoft SQL Server, SAP IQ, Snowflake

Syntax

deleteDataFiles=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether to delete only the data file (FALSE) or all files (TRUE) that are created during bulk loading (saving to the DBMS) or bulk unloading (retrieval from the DBMS).

Details

The files are deleted from both the local machine and from the cloud platform location.

Microsoft SQL Server: This option applies to the files that are created during bulk loading to Azure or Microsoft Fabric Data Warehouse. For more information, see "Bulk Loading to Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse" on page 241.

Snowflake: This option applies to the files that are created during bulk loading to Azure. For more information, see "Bulk Loading and Bulk Unloading with Microsoft Azure Synapse Analytics" on page 344.

delimiter= Data Connector Option

specifies a character that separates columns of data during bulk loading to or bulk unloading from your data source.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements [Amazon Redshift, Snowflake], PROC CASUTIL: SAVE statement [Google BigQuery, Microsoft SQL Server]

Category: Bulk Loading

Default: bell character (ASCII 0x07)

Requirements: *Amazon Redshift, Snowflake:* To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.

Google BigQuery, Microsoft SQL Server: To specify this option, you must also set bulkLoad=TRUE.

Data source: Amazon Redshift, Google BigQuery, Microsoft SQL Server, Snowflake

Syntax

delimiter=*"character"*

Required Argument

character
specifies a character that separates columns of data during bulk loading.

Details

You might specify a value for delimiter= if your character data contains the default delimiter character.

Microsoft SQL Server: This option is applicable only when bulk loading data to Microsoft Azure. For more information, see "Bulk Loading to Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse" on page 241.

directoriesAsData= Data Connector Option

specifies whether SAS treats a directory of Parquet data files as a single table.

Valid in:	CASLIB statement
Default:	FALSE
Requirement:	To enable this option, you must also set fileType="PARQUET".
Data source:	DuckDB
Note:	Support for this option was added in 2026.09.
See:	fileType=, "SAS_DUCKDB_DIRECTORIES_AS_DATA Environment Variable" in *SAS Global Statements: Reference*

Syntax

directoriesAsData=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether SAS treats a directory of Parquet data files as a single table.

Details

When you set directoriesAsData=TRUE, SAS treats a directory of Parquet data files as a single table. In the code that SAS passes to DuckDB, the 'globbing' syntax is used to access files. Here is an example:

```
caslib parts desc='DuckDB Caslib'
             dataSource=(srcType='duckdb' fileType=parquet filePath='/
mydata'
                         directoriesAsData=yes);

proc sql;
select * from parts.mytable;
quit;
```

SAS passes a SELECT statement that uses globbing syntax:

```
SELECT * FROM read_parquet('/mydata/mytable/**/*.parquet')
```

This data connector option overrides the value for the SAS_DUCKDB_DIRECTORIES_AS_DATA environment variable.

dispatchMethod= Data Connector Option

specifies how to submit a SAS Embedded Process job.

Valid in:	CASLIB statement, PROC CASUTIL: LOAD and SAVE statements
Default:	STANDARD
Restriction:	This option applies to parallel data transfer only.
Data source:	Hadoop, Spark

Syntax

dispatchMethod=STANDARD | OOZIE

Required Arguments

STANDARD
specifies to use the existing YARN process.

OOZIE
specifies to use the Oozie workflow process.

dm_unicode= Data Connector Option

specifies the Unicode encoding for the driver manager.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statements

Defaults: UCS-2 / UTF-16 [Impala]
UTF-8 [ODBC]

Data source: Impala, ODBC

Syntax

dm_unicode=*"Unicode-setting"*

Required Argument

Unicode-setting
specifies the Unicode encoding for the driver manager. Possible values include UTF-8, UCS-2, and so on. This setting applies to Linux platforms when using third-party ODBC driver managers, such as unixODBC.

dnsSuffix= Data Connector Option

specifies the network host name where the storage system resides.

Valid in: CASLIB statement

Default: dfs.core.windows.net

Interaction: The value for dnsSuffix= combines with the values for accountName=, fileSystem=, and folder= to define the external data location. (See Details.)

Data source: Microsoft SQL Server, Snowflake

Example: dnsSuffix="dfs.core.windows.net"

Syntax

dnsSuffix=*"network-storage-host-name"*

Required Argument

network-storage-host-name
specifies the network host name where the storage system resides. It is referred to as a suffix because the value is typically appended to the end of the storage account name.

Details

This value is used to specify the external data location, when combined with the values for accountName=, fileSystem=, and folder=. The data location is created in this way:

```
https://<accountname>.<dnssuffix>/<filesystem>/<folder>
```

Microsoft SQL Server: This option is applicable only when bulk loading data to Microsoft Azure or Microsoft Fabric Data Warehouse. For more information, see "Bulk Loading to Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse" on page 241.

Snowflake: This option is applicable only when bulk loading data to Microsoft Azure. For more information, see "Bulk Loading to Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse" on page 241.

driver_vendor= Data Connector Option

specifies the name of the third-party vendor that supports the Amazon Redshift driver.

Valid in: CASLIB statement

Default: SAS

Data source: Amazon Redshift

Syntax

driver_vendor="SAS" | "DATADIRECT" | "AMAZON"

Required Argument

"SAS" | "DATADIRECT" | "AMAZON"
specifies the name of the third-party vendor that supports the Amazon Redshift driver.

Details

By default, the Amazon Redshift data connector uses the SAS version of the DataDirect Amazon Redshift ODBC driver that is included with SAS/ACCESS Interface to Amazon Redshift. This driver corresponds to the value SAS.

driverClass= Data Connector Option

specifies the fully-qualified class name of a JDBC driver.

Defaults: none [JDBC]
"com.simba.hive.jdbc.HS2driver" [Hadoop]
"com.simba.databricks.jdbc.Driver" [Databricks]
"com.simba.spark.jdbc.Driver" [Spark]

Data source: Hadoop, JDBC, Spark

Note: This option is required when using a JDBC driver that is not redistributed with the SAS Viya platform. The default drivers changed in 2023.10.

Syntax

driverClass=*'driver'*

Details

This option specifies the JDBC driver for your database connection. If you use the classPath=option, the JDBC driver manager might not register your driver. By specifying driverClass=, you can ensure the JDBC driver is loaded.

For the Hadoop and Spark data connectors, the driverClass= option is typically not needed, as SAS automatically loads the appropriate JDBC driver.

duckdbThreads= Data Connector Option

specifies the number of threads for parallel query execution.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Default: none

Data source: DuckDB

Note: Support for this option was added in 2026.09.

See: "SAS_DUCKDB_THREADS Environment Variable" in *SAS Global Statements: Reference*

Example:
```
duckdbThreads=16
```

Syntax

duckdbThreads=*value*

Required Argument

value
specifies the number of threads for parallel query execution. SAS sets this value in the DuckDB configuration file for the current session.

Details

This data connector option overrides any value that you specify for the SAS_DUCKDB_THREADS environment variable.

dsn= Data Connector Option

specifies an ODBC data source name (DSN) that is defined by the SAS Federation Server administrator.

Valid in: CASLIB statement

Default: none

Data source: SAS Federation Server

Syntax

dsn=*"data-source-name"*

Required Argument

data-source-name
specifies the name of the data source.

enable_cleartextPlugin= Data Connector Option

specifies whether to enable the Cleartext plugin for authentication.

Valid in: CASLIB statement

Default: FALSE

Requirement: You must enable this option to connect to Microsoft Azure when you specify the userName= and password= options.

Data source: MySQL, SingleStore standard

Syntax

enable_cleartextPlugin=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether to enable the Cleartext plugin for authentication.

encoding= Data Connector Option

specifies the encoding for a data set.

Valid in: PROC CASUTIL: SAVE statement

Default: UTF-8

Data source: SAS data sets

Syntax

encoding=*"encoding"*

Required Argument

encoding
specifies the encoding for a data set.

encrypt= Data Connector Option

specifies the form of encryption to use to access a table.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement [SAS data sets], PROC CASUTIL: CONTENTS, LOAD, and SAVE statements [SPD Engine Files]

Default: NO

Restriction: *SPD Engine files:* This option applies only for serial data transfer.

Requirements: This option is required if your SAS data set or SPD Engine table uses AES encryption.

SAS data sets: When you create a table, the encryption key must be specified in uppercase letters.

Data source: SAS data sets, SPD Engine Files

Syntax

encrypt="YES" | "NO" | "AES" | "AES2"

Required Argument

"YES" | "NO" | "AES" | "AES2"
specifies the form of encryption to use to access a table.

Note: AES2 is not a valid value for SPD Engine Files.

encryptionPassword= Data Connector Option

specifies a passphrase to use for a data set or SPD Engine table that is encrypted using Advanced Encryption Standard (AES).

Valid in: CASLIB statement, PROC CASUTIL: DELETESOURCE, LOAD, SAVE statement [SAS data sets], PROC CASUTIL: CONTENTS, LOAD, and SAVE statements [SPD Engine Files]

Default: none

Restriction: *SPD Engine Files:* This option applies to serial data transfer only.

Requirements: This option is required only if the SAS data set or SPD Engine table uses AES encryption.

SAS data sets: When you create a table, the encryption key must be in uppercase letters.

Interaction: *SPD Engine Files:* You can specify this option only with the dataSource= option.

Data source: SAS data sets, SPD Engine Files

Syntax

encryptionPassword=*"passphrase"*

Required Argument

passphrase
specifies a passphrase to use for a data set or SPD Engine table that is encrypted using the Advanced Encryption Standard (AES).

epDatabase= Data Connector Option

specifies the name of the metadata database for SAS Embedded Process.

Valid in: CASLIB statement

Default: If you do not specify epDatabase= then the database specified in the database= option is used.

Data source: SAS SpeedyStore

Syntax

epDatabase=*"database-name"*

Details

When you assign a SAS SpeedyStore caslib, you can use the epDatabase= option to specify a SingleStore database for SAS Embedded Process to store metadata. A dedicated database is recommended. The metadata is not encrypted. If you must keep users from seeing each others' metadata, then specify a separate epDatabase= for each user. In this case, you must also protect the databases by using role-based access control in SingleStore.

The database contains tables and functions that are used during processing by SAS Embedded Process. The metadata is automatically deleted after processing completes. Examples of in-database processing are WHERE expressions or computed columns that are automatically processed in the SingleStore data source.

For details about supported processing, see "Restrictions for WHERE Processing and Computed Columns" on page 322.

One benefit of using epDatabase= is to separate the permissions granted for metadata from the permissions granted for data. Regardless of whether you use epDatabase=, you must have permissions to the SingleStore database where SAS Embedded Process stores temporary metadata. If you do not specify epDatabase=, then the database specified in the database= option is used. To grant the permissions, an administrator uses role-based access control in SingleStore.

See Also

- *SAS SpeedyStore: Administration and Configuration Guide*

epProperties= Data Connector Option

passes properties to the SAS Embedded Process.

Valid in: CASLIB statement

Default: none

Data source: SAS SpeedyStore

Note: This option was added in 2022.12.

Example:

```
caslib x datasource=(srctype="singlestore"
                     epProperties={"logger.level=trace","cmp.optimizations=all"}
                     database="my_db"
                     host="svc-sas-singlestore-cluster-dml" port="3306"
                     username="my_user" password="my_pwd"
                     );
```

Syntax

epProperties={*properties*}

Required Argument

properties

lists one or more properties, as "*name=value*" pairs, to pass to the SAS Embedded Process.

Enclose the list in braces. Enclose each *name=value* pair in single or double quotation marks, and use a comma to separate them in the list. The following properties are supported:

cmp.threads=*number*
specifies the maximum number of CMP threads that are used to run the CMP program. The maximum number of CMP threads cannot exceed the number of cores that are available in the system.

logger.level=*value*
specifies the logger level. The following levels are supported: allon, trace, debug, info, warn, error, fatal, off. This property can also be specified in the EPOPTIONS= argument of the RUNMODEL statement in the SCOREACCEL procedure.

ds2.dbMaxText=*number*
specifies the maximum length of character data types that are read into the SAS Embedded Process for in-database scoring. Specify this property for long character types such as LONGTEXT or LONGBLOB in order to avoid memory being exceeded. Note that this property is useful only for tables that were not created by CAS. When CAS creates a SingleStore table, the length attribute of these long character types is stored with the table. In that case, the length is honored by in-database scoring, and the ds2.dbMaxText= property is ignored. This property can also be specified in the EPOPTIONS= argument of the RUNMODEL statement in the SCOREACCEL procedure.

escape= Data Connector Option

specifies whether to escape a predefined set of special characters when you bulk load data to your cloud data source.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: FALSE

Requirements: *Amazon Redshift, Snowflake:* To specify this option, you must also set bulkLoad=TRUE.

Microsoft SQL Server: To specify this option you must set bulkLoad=TRUE and blFormat=DELIMITED.

Data source: Amazon Redshift, Microsoft SQL Server, Snowflake

Syntax

escape=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether to escape a predefined set of special characters when you bulk load data to your cloud data source.

Details

When set to TRUE, string delimiters, field (column) delimiters, and row delimiters are escaped. The double quotation mark (") is not escaped.

Microsoft SQL Server: This option is applicable only when bulk loading data to Microsoft Azure. For more information, see "Bulk Loading to Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse" on page 241.

excelOptions= Data Connector Option

specifies options that are passed to the read_xlsx function.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Alias: excel_options=

Default: none

Data source: DuckDB

Note: Support for this option was added in 2026.09.

Syntax

excelOptions=*"option1 value1<, option2 value2<, ...>>"*

Required Argument

option value
specifies options that are passed to the read_xlsx function. Specify the option name and the value to assign to it. Separate multiple name-value pairs by commas, and enclose the entire string in quotation marks.

extensionDirectory= Data Connector Option

specifies a temporary location where DuckDB can install extensions that facilitate access to non-native file types.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Alias: extension_directory=

Default: location that is specified by the UTILLOC system option

Data source: DuckDB

Note: Support for this option was added in 2026.09.

See: fileType=

Syntax

extensionDirectory=<"*path*">

Required Argument

path
specifies a temporary location where DuckDB can install extensions that facilitate access to non-native file types.

Details

There are DuckDB extensions that enable access to file types other than DUCKDB files. For example, an Iceberg extension is needed to access Iceberg files. Use this option if you need to install the extension in a directory that differs from the SAS Viya platform default location.

filePath= Data Connector Option

specifies the folder or file path for your data.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Alias: file_path=

Default: none

Interaction: If fileType=DUCKDB (the default), do not specify a value for filePath=. Instead, specify the DuckDB database location with the database= option.

Data source: DuckDB

Note: Support for this option was added in 2026.09.

See: database=, fileType=

Syntax

filePath=*"path"*

Required Argument

path
specifies the folder or file path for your data.

Details

When you connect to DuckDB data in a cloud environment, the path should include the appropriate prefix:

az://location/data_path	Microsoft Azure location
gs://bucket/data_path	Google Cloud Storage location
s3://bucket/data_path	Amazon S3 location

fileSystem= Data Connector Option

specifies the name of the file system within the external storage system that files are written to.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: none

Requirement: This option is required in the CASLIB statement when connecting to Microsoft Azure.

Interaction: The value for dnsSuffix= combines with the values for accountName=, fileSystem=, and folder= to define the external data location. (See Details.)

Data source: Microsoft SQL Server, Snowflake

Syntax

fileSystem=*"file-system-name"*

Required Argument

file-system-name
specifies the name of the file system within the external storage system that files are written to.

Details

This value is used to specify the external data location, when combined with the values for accountName=, dnsSuffix=, and folder=. The data location is created in this way:

```
https://<accountname>.<dnssuffix>/<filesystem>/<folder>
```

This option is applicable only when bulk loading data to Microsoft Azure or Microsoft Fabric Data Warehouse. For more information, see "Bulk Loading to Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse" on page 241.

fileType= Data Connector Option

specifies the file type.

Valid in: CASLIB statement [ORC, PC Files], PROC CASUTIL: DELETESOURCE, LOAD, SAVE statement

Alias: file_type=

Defaults: DUCKDB [DuckDB]

XLS [PC Files]

BASESAS, if you specify the SAS7BDAT extention in the file name. [SAS data sets]

Requirements: *ORC:* This option is required for the ORC data connector.

SAS data sets: This option is required for SAS7BDAT files only if you do not include the SAS7BDAT extension in the file name.

Data source: DuckDB, ORC, PC Files, SAS data sets

Note: Support for DuckDB was added in 2026.09.

Syntax

Form 1: DuckDB:

fileType=AVRO | CSV | DELTA | DUCKDB | EXCEL | ICEBERG | JSON | PARQUET

Form 2: ORC:

fileType="ORC"

Form 3: PARQUET:

fileType="PARQUET"

Form 4: PC Files:

fileType="DTA" | "EXCEL" | "JMP" | "SPSS" | "XLS"

Form 5: SAS data sets:

fileType="BASESAS"

Required Arguments

AVRO
specifies files with the row-based Avro file format.

Restrictions This value applies only to the DuckDB data connector.

You can read these files, but you cannot write to them.

BASESAS
specifies the file type for SAS data sets.

Restriction This value applies only to the data connector for SAS data sets.

CSV
specifies comma-separated value files.

Restriction This value applies only to the DuckDB data connector.

DELTA
specifies files with a Delta Lake format.

Restrictions This value applies only to the DuckDB data connector.

You can read these files, but you cannot write to them.

DTA
specifies the DTA file type.

Restriction This value applies only to the PC Files data connector.

DUCKDB
specifies the native DuckDB file format.

Restriction This value applies only to the DuckDB data connector.

EXCEL
specifies the XLSX file type.

Restriction This value applies only to the DuckDB and PC Files data connectors.

Requirement Use this type for files that contain national language characters. These characters are not displayed correctly for XLS files.

ICEBERG
specifies a file with an Apache Iceberg format.

Restriction This value applies only to the DuckDB data connector.

JMP
specifies a JMP file type.

Restriction This value applies only to the PC Files data connector.

JSON
specifies a file with the JSON format.

Restriction This value applies only to the DuckDB data connector.

ORC
specifies an ORC file type.

Restriction This value applies only to the ORC data connector.

PARQUET
specifies a PARQUET file type.

Restriction This value applies only to the DuckDB data connector or to the PARQUET file type.

SPSS
specifies the SPSS file type.

Restriction This value applies only to the PC Files data connector.

XLS
specifies the XLS file type.

Restriction This value applies only to the PC Files data connector.

folder= Data Connector Option

specifies the folder or file path for the data in the external storage system.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: none

Interaction: This value is used to specify the external data location, when combined with the values for accountName=, dnsSuffix=, and fileSystem=. The data location is created in this way:

```
https://<accountname>.<dnssuffix>/<filesystem>/<folder>
```

Data source: Microsoft SQL Server

Syntax

folder=*"file-path-and-folder"*

Required Argument

file-path-and-folder
specifies the folder or file path for the data in the external storage system. The location path begins with the container name.

Details

This option is applicable only when bulk loading data to Microsoft Azure or Microsoft Fabric Data Warehouse. For more information, see "Bulk Loading to Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse" on page 241.

getNames= Data Connector Option

specifies whether to retrieve variable names from the first line of an input file.

Valid in: CASLIB statement

Default: TRUE

Data source: PC Files

Syntax

getNames=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether to retrieve variable names from the first line of an input file.

getSheetNames= Data Connector Option

specifies whether to create a CAS table with a list of all sheet names in an Excel workbook.

Valid in: CASLIB statement

Default: FALSE

Interaction: When set to TRUE, this option overrides the sheet= option.

Data source: PC Files

Syntax

getSheetNames=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether to create a CAS table with a list of all sheet names in an Excel workbook.

gcsAccessKey= Data Connector Option

specifies your GCS HMAC access ID.

Valid in: CASLIB statement

Aliases: gcs_access_key=
gcs_key_id=

Default: none

Interaction: The value that you provide for this option corresponds to the HMAC secret that you provide for gcsSecret=.

Data source: DuckDB

Note: Support for this option was added in 2026.09.

See: gcsSecret= on page 499

Example:
```
gcsAccessKey="GOOG12LASKDJFAOWEIRNSLDKFHASDFIALSDKFJALSDKFJGEHHW"
```

Syntax

gcsAccessKey=*"access-ID"*

Required Argument

access-ID
specifies your Google Cloud Storage HMAC access ID.

gcsAuthDomain= Data Connector Option

specifies the authentication domain to use to authenticate to Google Cloud Storage.

Valid in: CASLIB statement

Aliases: gcs_authdomain=

gcsdomain=

Default: none

Data source: DuckDB

Note: Support for this option was added in 2026.09.

Syntax

gcsAuthDomain=*"value"*

Required Argument

value
specifies the authentication domain to use to authenticate to Google Cloud Storage.

gcsCredFile= Data Connector Option

specifies the path to a credentials file that enables authentication to Google Cloud Storage.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Aliases: gcs_cred_file=
gcs_auth_file=

Default: none

Data source: DuckDB

Note: Support for this option was added in 2026.09.

Syntax

gcsCredFile=*'path-and-file-name'*

Required Argument

path-and-file-name
specifies the location of a JSON credentials file that enables authentication to Google Cloud Storage. Specify this value within single quotation marks only.

Details

This value facilitates functionality like PROC CONTENTS, PROC DATASETS, and PROC DELETE for DuckDB files.

The value for this option is masked in the SAS log. This option accepts values that have been encoded using PROC PWENCODE. SAS/ACCESS recognizes encoded values as those that begin with a SAS encoding tag. For more information, see "PWENCODE Procedure" in *Base SAS Procedures Guide*.

For more information about the credentials file, see the Getting Started information in your Google Cloud documentation.

gcsSecret= Data Connector Option

specifies the HMAC secret that is linked to your HMAC access ID.

Valid in: CASLIB statement

Alias: gcs_secret=

Default: none

Interaction: The value for this option corresponds to the HMAC access ID that you specify for gcsAccessKey=.

Data source: DuckDB

Note: Support for this option was added in 2026.09.

See: gcsAccessKey= on page 497

Example:
```
gcsSecret="40-byte-secret"
```

Syntax

gcsSecret=*"value"*

Required Argument

value
specifies the HMAC secret that is associated with your HMAC access ID.

hadoopConfigDir= Data Connector Option

specifies a location for Hadoop configuration files.

Valid in: CASLIB statement

Default: none

Interaction: For more requirements that are not included in the details below, see the documentation for each data connector.

Data source: Hadoop, Impala, Spark, SPD Engine Files

Syntax

hadoopConfigDir=*"configuration-files-directory"*

Required Argument

configuration-files-directory
specifies a location for Hadoop configuration files.

Details

For parallel data transfer and in-database model scoring in Hadoop, this location includes the client-side spark-defaults.conf and sparkep-defaults.conf files. These files must be copied to the client side during the deployment of SAS Embedded Process. For information about this configuration task, see *SAS In-Database Products: Deployment and Administration Guide*.

For serial data transfer, prior to 2025.08, this option was used to specify the location of configuration files obtained by running the Hadoop tracer script on the target cluster. Starting in 2025.08, when using WebHDFS, you can use the webHdfsUrl= option. In that case, running the Hadoop tracer script is no longer required. Running the tracer script was previously a requirement for using bulkLoad=TRUE (which is the default) with serial data transfer. For more information, see "Configuring SAS/ACCESS Interface to Hadoop" in *SAS/ACCESS for Relational Databases: Reference*.

For the Hadoop and Spark data connectors, several properties that were previously set in hive-site.xml can now be specified in the caslib assignment. These options include allowSelfSignedCerts=, httpPath=, mapReduceJobName=, mapReduceQueueName=, sslTrustStore=, tezQueueName=, trustStorePassword=, zooKeeperNamespace=, and zooKeeperQuorum=. If the hive-site.xml file exists in the specified location, it is used. If any of these options are specified in the data connector, they override the properties in hive-site.xml.

SPD Engine Files: For parallel data transfer, this location must contain the client-side spark-defaults.conf and sparkep-defaults.conf files as well as the files that are obtained by running the Hadoop tracer script. For serial data transfer, only the files from the Hadoop tracer script are required.

hadoopJarPath= Data Connector Option

specifies one or more paths to the JAR files.

Valid in: CASLIB statement

Default: none

Restriction: *Hadoop*: Do not specify this option in Hive connections for in-database processing or parallel load.

Requirements: *Hadoop, Spark, Impala:* This option is required to use HDFS.
All data sources: Files must be delimited by colons (:).

Interaction: *SPD Engine Files:* To use this option, you must also set hdfs=TRUE.

Data source: Hadoop, Impala, Spark, SPD Engine Files

Syntax

hadoopJarPath=*"JAR-files-path"*

Required Argument

JAR-files-path
specifies one or more paths to the JAR files. These files are delimited by colons.

Details

If you are accessing Hive tables that require a customized Hive SerDe, you must include the SerDe JAR files in the hadoopJarPath.

Hadoop, Spark, SPD Engine Files: These are the JAR files that you can obtain by running the Hadoop tracer tool on the target cluster. For Spark, the tracer tool obtains the Spark configuration file and Spark JAR files along with the Hadoop configuration and JAR files. The Spark JAR files are stored in a subfolder called **`spark`** under the Hadoop JAR folder.

Spark: If you specify Spark as the data source (`PLATFORM="spark"`), you must include the Spark JAR files in the hadoopJarPath. On Hortonworks Hadoop, the Spark JAR folder must be the last one on the JAR path. On Cloudera Hadoop, the Spark JAR folder must the first one on the JAR path.

hdfs= Data Connector Option

specifies whether SPD Engine file data is in HDFS.

Valid in: CASLIB statement

Default: FALSE (serial data transfer), TRUE (parallel data transfer)

Restriction: This option is used only for serial data transfer.

Data source: SPD Engine Files

Syntax

hdfs=TRUE | FALSE

Required Arguments

TRUE
specifies that SPD Engine file data is stored in HDFS.

FALSE
specifies that SPD Engine file data is stored on a traditional disk system.

hdfsTempDir= Data Connector Option

specifies the Hadoop HDFS directory to use to store temporary data.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements

Defaults: /TMP [Hadoop, Impala, Spark]
/tmp [SPD Engine Files]

Restriction: *SPD Engine Files:* This option applies to parallel data transfer only.

Interaction: *SPD Engine Files:* This option requires that you set hdfs=TRUE.

Data source: Hadoop, Impala, Spark, SPD Engine Files

Syntax

hdfsTempDir=*"Hadoop-HDFS-temporary-directory"*

Required Argument

Hadoop-HDFS-temporary-directory
specifies the Hadoop HDFS directory to use to store temporary data.

host= Data Connector Option

specifies the host name.

Valid in: CASLIB statement

Alias: server= [MySQL, SAS SpeedyStore, SingleStore standard]

Default: none

Requirement: This option is required in the CASLIB statement.

Data source: Informix, MySQL, SAP IQ, SAS SpeedyStore, SingleStore standard

Syntax

Form 1: All data sources, including SAP IQ:

host=*"host-name"*

Required Argument

host-name
specifies the host name.

Details

MySQL: This value is the MySQL host name.

SAP IQ: This value specifies a host name that tells the client where to find the database server.

SingleStore: This value is the child aggregator endpoint.

httpPath= Data Connector Option

specifies the HTTP or HTTPS endpoint for a connection.

Valid in: CASLIB statement

Category: Data Access

Default: None

Requirements: [Spark] The platform= data connector option is specified.
Server connection properties are specified using data connector options.

Interaction: httpPath= is ignored when uri= is specified.

Data source: Hadoop, Spark

Syntax

httpPath="*http-path*"

Required Argument

http-path
specifies the path of the HTTP or HTTPS endpoint used for connections when HTTP mode is enabled.

iamAssumeRoleARN= Data Connector Option

specifies an Amazon Resource Name (ARN) that identifies an IAM role.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements

Default: none

Interactions: This option is used only when you specify bulkLoad=TRUE or bulkUnload=TRUE.
If you specify both iamAssumeRoleARN= and iamAssumeRoleName=, then the iamAssumeRoleName= value is used.

Data source: Amazon Redshift

Example: iamAssumeRoleARN='arn:aws:iam::111122223333:role/eard'

Syntax

iamAssumeRoleARN=*"ARN-value"*

Required Argument

ARN-value
specifies an Amazon Resource Name (ARN) that identifies an IAM role.

Details

This IAM role enables you to get access credentials for S3 via the AssumeRole IAM operation.

iamAssumeRoleName= Data Connector Option

specifies an AWS IAM role name that enables you to get access credentials for S3 via the AssumeRole IAM operation.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements

Default: none

Interactions: This option is used only when you specify bulkLoad=TRUE or bulkUnload=TRUE.

If you specify both iamAssumeRoleARN= and iamAssumeRoleName=, then the iamAssumeRoleName= value is used.

Data source: Amazon Redshift

Example: iamAssumeRoleName='eard'

Syntax

iamAssumeRoleName=*"IAM-role-name"*

Required Argument

IAM-role-name
specifies an AWS IAM role name that enables you to get access credentials for S3 via the AssumeRole IAM operation.

icebergCatalog= Data Connector Option

specifies the identifier of an Iceberg catalog.

Valid in: CASLIB statement

Default: none

Requirement: This option is required when you set fileType=ICEBERG and you want to write to an Iceberg data source.

Data source: DuckDB

Note: Support for this option was added in 2026.09.

Example: To connect to an S3 bucket:

```
icebergCatalog='arn:aws:s3tables:us-east-1:012345678901:bucket/my-table-bucket'
```

Syntax

icebergCatalog=*"value"*

Required Argument

value
specifies the identifier for an Iceberg catalog.

The identifier that you specify varies based on the catalog provider. For example, provide the Amazon Resource Name (ARN) for an S3 Tables bucket. (You can obtain this value from the AWS Management Console.)

Details

The identifier that you provide is used for the ATTACH command to bind your REST catalog with the Iceberg catalog in DuckDB.

icebergClientId= Data Connector Option

specifies the client ID for authentication to an Iceberg catalog.

Valid in:	CASLIB statement
Default:	none
Data source:	DuckDB
Note:	Support for this option was added in 2026.09.
See:	icebergClientSecret= on page 507

Syntax

icebergClientId=*"identifier"*

Required Argument

identifier
specifies the client ID for authentication to an Iceberg catalog.

icebergClientSecret= Data Connector Option

specifies the client secret for authentication to an Iceberg catalog.

Valid in:	CASLIB statement
Alias:	iceberg_client_secret=
Default:	none
Interaction:	Specify this option with icebergClientId=.
Data source:	DuckDB
Note:	Support for this option was added in 2026.09.
See:	icebergClientId= on page 506

Syntax

icebergClientSecret=*"secret"*

Required Argument

secret
specifies the client secret for authentication to an Iceberg catalog.

icebergEndpoint= Data Connector Option

specifies the Iceberg endpoint URL to connect to an Iceberg REST catalog.

Valid in:	CASLIB statement
Default:	none
Restriction:	Specify either icebergEndpoint= or icebergEndpointType= to connect to an Iceberg catalog. Do not specify both of these options.
Data source:	DuckDB
Note:	Support for this option was added in 2026.09.
See:	icebergEndpointType= on page 508

Syntax

icebergEndpoint=*"URL"*

Required Argument

URL
specifies the Iceberg endpoint URL for connecting to an Iceberg REST catalog.

icebergEndpointType= Data Connector Option

specifies the Iceberg endpoint type for connecting to an Iceberg catalog.

Valid in: CASLIB statement

Default: none

Restriction: Specify either icebergEndpoint= or icebergEndpointType= to connect to an Iceberg catalog. Do not specify both of these options.

Data source: DuckDB

Note: Support for this option was added in 2026.09.

See: icebergEndpoint= on page 507

Syntax

icebergEndpointType=GLUE | S3_TABLES

Required Arguments

GLUE
specifies a connection to an Amazon Glue catalog.

S3_TABLES
specifies a connection to an S3 bucket.

icebergOauthUri= Data Connector Option

specifies the OAuth2 server URI to authenticate to for an Iceberg catalog.

Valid in: CASLIB statement

Alias: icebergOauth2ServerUri=

Default: none

Interaction: When you use this option, also specify icebergToken=.

Data source: DuckDB

Note: Support for this option was added in 2026.09.

See: icebergToken= on page 509

Syntax

icebergOauthUri=*'server-URI'*

Required Argument

server-URI
specifies the OAuth2 server URI to authenticate to for an Iceberg catalog.

icebergSecretOptions= Data Connector Option

specifies the client secret for authentication to an Iceberg catalog.

Valid in: CASLIB statement

Default: none

Interaction: Specify this option with icebergClientId= and icebergClientSecret=.

Data source: DuckDB

Note: Support for this option was added in 2026.09.

See: icebergClientId= on page 506, icebergClientSecret= on page 507

Example:
```
icebergSecretOptions="DEFAULT_REGION 'us-east-1', AUTHORIZATION_TYPE 'SigV4'"
```

Syntax

icebergSecretOptions=*"option-value-pairs"*

Required Argument

option-value-pairs
specifies additional secret options for authenticating to an Iceberg catalog. Enclose one or more option-value pairs in double quotation marks. Specify option-value pairs with a space between the option name and the value, and enclose the value in single quotation marks. Separate multiple option-value pairs with a comma. These options are used for the DuckDB CREATE SECRET command.

icebergToken= Data Connector Option

specifies the OAuth2 token to authenticate to an Iceberg catalog.

Valid in: CASLIB statement

Default: none

Requirement: When you use this option, also specify icebergOauthUri=.

Data source: DuckDB

Note: Support for this option was added in 2026.09.

See: icebergOauthUri= on page 508

Syntax

icebergToken=*"OAuth2-token"*

Required Argument

OAuth2-token
specifies the OAuth2 token to authenticate to an Iceberg catalog.

identity= Data Connector Option

specifies the authentication method for the COPY INTO command.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: none

Interaction: The identity= value and the secret= value are supplied to the CREDENTIAL argument in the COPY command for Azure Synapse Analytics (SQL DW) or Microsoft Fabric Data Warehouse.

Data source: Microsoft SQL Server

Syntax

identity=*"identity-value"*

Required Argument

identity-value
specifies the authentication method for the COPY INTO command. Possible values are “Shared Access Signature”, “Storage Account Key”, “Managed Identity”, or the OAuth 2.0 token endpoint.

Details

This option is applicable only when bulk loading data to Microsoft Azure or Microsoft Fabric Data Warehouse. For more information, see “Bulk Loading to

Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse" on page 241.

ignoreFedSQLObjects= Data Connector Option

specifies whether to ignore FedSQL objects in a library query.

Valid in: CASLIB statement, PROC CASUTIL: LIST statements

Alias: ignore_fedsql_objects=

Default: FALSE

Restriction: This option applies only when working with PROC DS2 and PROC FEDSQL.

Data source: DuckDB, Google BigQuery

Note: Support for DuckDB was added in 2026.09.

Syntax

ignoreFedSQLObjects=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether to ignore FedSQL objects in a library query.

Details

When you enable the IGNORE_FEDSQL_OBJECTS= option, a library query ignores FedSQL objects such as FedSQL views, DS2 packages, or DS2 threads. These objects are typically created by calling PROC FEDSQL or PROC DS2. When you specify that FedSQL objects should be ignored, you often improve the performance of library queries or requests to view data.

insertBuff= Data Connector Option

specifies the number of rows per block of data to save to a table in the data source.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: Calculated automatically based on row size

Restriction: *Hadoop, Teradata, SPD Engine files:* This option applies to serial data transfer only.

Data source: Amazon Redshift, DB2, DuckDB, Greenplum, Hadoop, Impala, Informix, JDBC, Microsoft SQL Server, MongoDB, Netezza, ODBC, Oracle. PostgreSQL, SAP HANA, SAP IQ, SAS Federation Server, Snowflake, Spark, Teradata, Vertica, Yellowbrick

Note: Support for DuckDB was added in 2026.09.

Syntax

insertBuff=*number-of-rows*

Required Argument

number-of-rows
specifies the number of rows per block of data to save to a table in the data source.

insertMultiValue= Data Connector Option

specifies whether to use multi-value INSERT syntax to load data.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Alias: insert_multi_value=

Defaults: FALSE (most Microsoft SQL Server data sources)
TRUE (Microsoft Fabric Data Warehouse)

Interaction: Do not set bulkLoad=TRUE when you set insertMultiValue=TRUE.

Data source: Microsoft SQL Server

Note: Support for this option was added in 2026.02.

See: bulkLoad=

Syntax

insertMultiValue=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether to use multi-value INSERT syntax to load data.

Details

When you set this value to TRUE, SAS uses multi-value INSERT syntax to load data to your data source. This syntax is similar to this sample that loads data to three columns in a table:

```
INSERT INTO [table-name] (Column1,Column2,Column3)
VALUES
('Value1', 'Value2', 'Value3'),
('Value4', 'Value5', 'Value6'),
('Value7', 'Value8', 'Value9');
```

insertSQL= Data Connector Option

specifies whether to use a parameterized INSERT SQL statement to write data.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Alias: insert_sql=

Defaults: TRUE [DB2, DuckDB, Microsoft SQL Server]
FALSE [ODBC]

Data source: DB2, DuckDB, Microsoft SQL Server, ODBC

Note: Support for DuckDB was added in 2026.09.

Syntax

insertSQL=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether to use a parameterized INSERT SQL statement to write data to the data source.

Details

Use this option to reduce the time to write data to your data source.

ODBC: When you use this data connector option for an ODBC data source, it is recommended that you test the performance for writing data to the data source. For some data sources, bulk-loading options improve write performance more than using a parameterized INSERT statement. However, for other data sources, using the parameterized INSERT statement results in improved write performance.

instance= Data Connector Option

specifies the SAP HANA port instance.

Valid in: CASLIB statement

Default: 00

Restriction: Specify a value for only instance= or port=, not both. (As an alternative to port=, you can instead specify server="*server-identifier:port*.) If you specify a value for instance and port= (or a server value that includes a port value), then a connection error results.

Data source: SAP HANA

Syntax

instance=*"number"*

Required Argument

number
specifies the SAP HANA port instance.

invalidDate= Data Connector Option

specifies how to treat invalid date values when loading DATE or DATETIME values.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Defaults: WARN when loading multiple records. ERROR when loading records individually. [DB2, Microsoft SQL Server, ODBC, PostgreSQL]

WARN [Hadoop, JDBC, Oracle, Spark]

Interaction: Use the readBuff= option to control how many records are retrieved in a Load operation.

Data source: DB2, Hadoop, JDBC, Microsoft SQL Server, ODBC, Oracle, PostgreSQL, Spark

See: readBuff= data connector option

Syntax

invalidDate="ERROR" | "SILENT" | "WARN"

Required Arguments

ERROR
specifies that when an invalid DATE or DATETIME value is detected, the load process ends, and an error is printed in the SAS log.

SILENT
specifies that SAS converts an invalid DATE or DATETIME value to a missing value, and no message is printed in the SAS log.

WARN
specifies that SAS converts an invalid DATE or DATETIME value to a missing value and prints a warning in the SAS log.

Details

For more information about working with dates with the SAS Viya platform, see "SAS Date Values" in *SAS Formats and Informats: Reference*.

jobManagementURL= Data Connector Option

specifies the URL to use to submit execution requests over a REST interface.

Valid in:	CASLIB statement
Alias:	restURL=
Default:	none
Requirement:	*SPD Engine files:* This option requires that you set hdfs=TRUE and dataTransferMode="PARALLEL".
Data source:	Hadoop, Spark, SPD Engine files

Syntax

jobManagementURL=*"rest-URL"*

Required Argument

rest-URL
specifies the URL to use to submit execution requests over a REST interface.

Details

This option is used for services such as Apache Livy or Databricks notebooks.

Spark: For more information, see "Running Scoring Models in Databricks Using PROC SCOREACCEL or CAS Actions" in *SAS In-Database Products: User's Guide*.

jsonOptions= Data Connector Option

specifies JSON-related options to pass to the database.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Alias: json_options=

Default: none

Data source: DuckDB

Note: Support for this option was added in 2026.09.

Syntax

jsonOptions=*"option1*<, *option2*<, ...>>*"*

Required Argument

option
specifies options to pass to the read_json function. Separate multiple options by commas, and enclose the entire string in quotation marks.

key= Data Connector Option

specifies the Amazon Web Services access key that is used with key-based access control.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements [Amazon Redshift], PROC CASUTIL: SAVE statement [Snowflake]

Alias: blEncKey= [Amazon Redshift]

Default: none

Requirements: *Amazon Redshift:* To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.

Snowflake: To specify this option, you must also set bulkLoad=TRUE.

Data source: Amazon Redshift, Snowflake

See: bucket= data connector option, bulkLoad= data connector option, bulkUnload= data connector option

Syntax

key=*"value"*

Required Argument

value
specifies the Amazon Web Services access key that is used with key-based access control.

Details

Overview of key= Usage

If you are using temporary token credentials, this is the temporary access key ID.

Amazon Redshift: Encryption in S3 Environments

All S3 buckets have encryption enabled by default for data that is stored in S3. All new objects that are uploaded are automatically encrypted by using server-side encryption with S3-managed keys (SSE-S3). This encryption uses the AES-256 algorithm and is managed by AWS, which generates and rotates encryption keys. Data is encrypted and decrypted transparently for users.

A *MinIO environment* is an S3-compatible object storage platform that can run on AWS or on other platforms. MinIO supports server-side encryption using AES-256-GCM. The default encryption behavior is specific to your MinIO environment.

In cases when you want to specify an encryption key, use the blEncKey= (or key=) option, you provide the name of the encryption key that you are using to bulk load, bulk unload, and access your data. You use the S3 procedure to register an encryption key and assign a corresponding name. This is the name that you provide as the value for the blEncKey= option. For more information, see the ENCKEY statement of the "S3 Procedure" in *Base SAS Procedures Guide*.

When you use the blEncKey= option with the Amazon Redshift data connector, also provide the name of the S3 bucket that you are accessing. Here is an example:

```
caslib "myRSlib" datasource=(srctype="redshift" username='myUser' password="myPwd"
                server="myserver.us-east-1.redshift.amazonaws.com" database='myDatabase'
                bulkload=true bulkunload=true region='us-east-1'
                bucket='myS3bucket' blEncKey="<encryption-key>" secret="<aws-secret>");
```

krbAuthType= Data Connector Option

Specifies how to obtain the subject for Kerberos authentication.

Valid in: CASLIB statement

Default: none

Requirement: This option is required when using the default Simba JDBC driver in an environment that uses Kerberos.

Data source: Hadoop

Note: Support for this option was added in 2025.07.

Example: `krbAuthType=2`

Example: "Example 3: Connecting to Hive with Kerberos Authentication" on page 220

Syntax

krbAuthType=*integer*

Required Argument

integer
specifies how to obtain the subject for Kerberos authentication. Valid values are 0, 1, 2, or 3. For more information, see "Using Kerberos" and "Connector Configuration Options" on the insightsoftware website.

krbHostFQDN= Data Connector Option

Specifies the fully qualified domain name of the service principal host.

Valid in: CASLIB statement

Default: none

Interaction: When you connect to an environment that uses Kerberos, specify krbHostFQDN=, krbRealm=, and krbServicename=.

Data source: Hadoop, Impala

Notes: Support for Impala was added in 2025.09.
Support for this option was added in 2025.07.

See: krbRealm= data connector option, krbServiceName= data connector

Example: `krbHostFQDN=myhost.company.com`

Examples: "Example 3: Connecting to Hive with Kerberos Authentication" on page 220,
"Example 4: Connect to Impala with Kerberos Authentication"

Syntax

krbHostFQDN=*domain-name*

Required Argument

domain-name
specifies the fully qualified domain name of the service principal host.

Details

SAS uses the values that you provide for krbServiceName=, krbHostFQDN=, and krbRealm= to build the Kerberos service principal that is passed to the data source:

`<krbServiceName>/<krbHostFQDN>@<krbRealm>`

For more information, see "Using Kerberos" and "Connector Configuration Options" on the insightsoftware website.

krbRealm= Data Connector Option

Specifies the Kerberos realm for authentication.

Valid in:	CASLIB statement
Default:	none
Requirement:	When connecting to an environment that uses Kerberos, specify krbHostFQDN=, krbRealm=, and krbServiceName=.
Data source:	Hadoop, Impala
Notes:	Support for Impala was added in 2025.09. Support for this option was added in 2025.07.
See:	krbHostFQDN= data connector option, krbServiceName= data connector
Example:	`krbRealm=example.com`
Examples:	“Example 3: Connecting to Hive with Kerberos Authentication” on page 220, “Example 4: Connect to Impala with Kerberos Authentication”

Syntax

krbRealm=<">*realm-name*<">

Required Argument

realm-name
specifies the Kerberos realm (authentication domain) of the service principal.

Details

SAS uses the values that you provide for krbServiceName=, krbHostFQDN=, and krbRealm= to build the Kerberos service principal that is passed to the data source:

`<krbServiceName>/<krbHostFQDN>@<krbRealm>`

For more information, see "Using Kerberos" and "Connector Configuration Options" on the insightsoftware website.

krbServiceName= Data Connector Option

Specifies the Kerberos service to request a ticket for.

Valid in:	CASLIB statement
Default:	none
Requirement:	When connecting to an environment that uses Kerberos, specify krbHostFQDN=, krbRealm=, and krbServiceName=.
Data source:	Hadoop, Impala
Notes:	Support for Impala was added in 2025.09. Support for this option was added in 2025.07.
See:	krbHostFQDN= data connector option, krbRealm= data connector option
Example:	`krbServiceName=hive` `krbServiceName=impala`
Examples:	“Example 3: Connecting to Hive with Kerberos Authentication” on page 220, “Example 4: Connect to Impala with Kerberos Authentication”

Syntax

krbServiceName=<">*service-name*<">

Required Argument

service-name
specifies the Kerberos service to request a ticket for.

Details

SAS uses the values that you provide for krbServiceName=, krbHostFQDN=, and krbRealm= to build the Kerberos service principal that is passed to the data source:

`<krbServiceName>/<krbHostFQDN>@<krbRealm>`

For more information, see "Using Kerberos" and "Connector Configuration Options" on the insightsoftware website.

Hadoop: Typically, the value for Hadoop is **hive**.

Impala: Typically, the value for Impala is **impala**.

label= Data Connector Option

specifies a label name for the created SPD Engine table.

Valid in: CASLIB statement

Default: none

Restriction: This option applies to serial data transfer only.

Data source: SPD Engine Files

Syntax

label=*"label-name"*

Required Argument

label-name
specifies a label name for the created SPD Engine table.

largeResultsDataSet= Data Connector Option

specifies a data set ID that stores query results. If the specified data set ID does not exist, then it is created.

Valid in: CASLIB statement

Alias: LRD=

Default: _sasbq_temp_tables

Requirement: You must have Write permission for Google BigQuery to specify this option.

Interaction: This option is used in conjunction with the allowLargeResults= and largeResultsExpirationTime= options.

Data source: Google BigQuery

Syntax

largeResultsDataSet=*BigQuery-data-set-ID*

Required Argument

BigQuery-data-set-ID
specifies a data set ID that stores query results. If the specified data set ID does not exist, then it is created.

largeResultsExpirationTime= Data Connector Option

specifies the time, in milliseconds, until a temporary table is deleted.

Valid in:	CASLIB statement
Alias:	LRET=
Default:	86400000 (24 hours)
Range:	1 (do not expire), 3600000–2147483647
Interaction:	This option is used in conjunction with the allowLargeResults= and largeResultsDataSet= options.
Data source:	Google BigQuery

Syntax

largeResultsExpirationTime=*value*

Required Argument

value
specifies the time, in milliseconds, until a temporary table is deleted. The special value 1 indicates that the temporary table should not expire.

largeResultsProject= Data Connector Option

specifies a project ID that stores query results.

Valid in:	CASLIB statement
Alias:	LRP=
Default:	none

Requirement: You must have Write permission for Google BigQuery to specify this option.

Interaction: This option is used in conjunction with the allowLargeResults=, largeResultsDataSet=, and largeResultsExpirationTime= options.

Data source: Google BigQuery

See: allowLargeResults=, largeResultsDataSet=, largeResultsExpirationTime=

Syntax

largeResultsProject=*project-ID*

Required Argument

project-ID
specifies a project ID that stores query results.

Details

By default, large results are saved to the project that is accessed when you log into Google BigQuery. You can specify the login project for this option, or you can use this option to specify an alternative project to use for large query results.

ldPath= Data Connector Option

specifies a full path to append to the existing LD_LIBRARY_PATH value for your system.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements

Default: none

Restriction: To specify this option, you must also specify bulkLoad=TRUE or bulkUnload=TRUE.

Data source: PostgreSQL

Note: Support for this option was removed in 2025.03.

Syntax

ldPath=*<'>LD-library-path<'>*

Required Argument

LD-library-path
specifies a full path to append to the existing LD_LIBRARY_PATH value for your system.

This option applies only to bulk loading to PostgreSQL or bulk unloading from PostgreSQL.

Details

Starting in 2025.03, bulk loading and bulk unloading uses the COPY API for PostgreSQL. For this reason, this option, which works with the PSQL utility, is no longer used.

login_timeout= Data Connector Option

specifies the login time limit, in seconds, for connecting to and accessing data sources.

Valid in: CASLIB statement

Default: 30

Data source: Hadoop, JDBC, Spark

Syntax

login_timeout=*seconds*

Required Argument

seconds
specifies the default login time-out in seconds for connecting to and accessing data sources before ending a login attempt.

mapReduceJobName= Data Connector Option

specifies the MapReduce job name.

Valid in: CASLIB statement

Default: value configured in hive-site.xml file

Data source: Hadoop, Spark

See: mapReduceQueueName= data connector option

Syntax

mapReduceJobName=*"name"*

Required Argument

name
specifies the name of the MapReduce job.

mapReduceQueueName= Data Connector Option

specifies the MapReduce queue name.

Valid in: CASLIB statement

Default: value configured in hive-site.xml file

Data source: Hadoop, Spark

See: mapReduceJobName= data connector option

Syntax

mapReduceJobQueueName=*"name"*

Required Argument

name
specifies the name of the queue to route MapReduce jobs.

masterPW= Data Connector Option

specifies the password to access, delete, modify, read, or save a password-protected SPD Engine table.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Default: none

Restriction: This option applies to parallel data transfer for writing to new tables (SAVE statement). This option applies to serial data transfer when you write to an existing table (SAVE statement) in the SPD Engine data source. This option applies to serial data transfer for loading data to CAS (LOAD statement).

Data source: SPD Engine Files

Syntax

masterPW=*"password"*

Required Argument

password
specifies the password to access, delete, modify, read, or save a password-protected SPD Engine table.

max_binary_len= Data Connector Option

specifies the maximum length of a binary variable when you read binary data.

Valid in: CASLIB statement

Default: 2000

Data source: DuckDB

Note: Support for this option was added in 2026.09.

Syntax

max_binary_len=*value*

Required Argument

value
specifies the maximum length of a binary variable when you read binary data.

max_char_len= Data Connector Option

specifies the maximum column width.

Valid in: CASLIB statement [MongoDB, MySQL], PROC CASUTIL: LOAD statement [MongoDB]

Defaults: 2000 [DuckDB]
32767 [MongoDB]

Range: 1–32767

Restriction: *DuckDB:* This option pertains only to VARCHAR data in DuckDB.

Interaction: *DuckDB:* This option is used only when scanStringColumns=FALSE.

Data source: DuckDB, MongoDB

Note: Support for DuckDB was added in 2026.09.

Syntax

max_char_len=*value*

Required Argument

value
for MongoDB, specifies the maximum column width for results that are returned from the external database. For MySQL, this value specifies the number of characters that are allowed for CHAR values when you create a table.

max_string_size= Data Connector Option

specifies whether to override string limits that are returned via Oracle.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Default: NONE

Data source: Oracle

Syntax

max_string_size=NONE | EXTENDED | STANDARD

Required Arguments

NONE
specifies that SAS/ACCESS uses the MAX_STRING_SIZE value that is returned when Oracle is queried.

EXTENDED
specifies that SAS/ACCESS uses length limits for character values that correspond to those that Oracle uses when the MAX_STRING_SIZE parameter is set to EXTENDED.

STANDARD
specifies that SAS/ACCESS uses length limits for character values that correspond to those that Oracle uses when the MAX_STRING_SIZE parameter is set to STANDARD.

Details

Typically, this option is not required for your connection to Oracle. Use this option if SAS Technical Support advises you to do so.

The value that you specify for this option should match the value of the MAX_STRING_SIZE parameter in Oracle. Verify with your database administrator the MAX_STRING_SIZE value that is used in your Oracle database. In some cases, Oracle returns an incorrect value for the MAX_STRING_SIZE parameter when queried. In those cases, you can use this option to explicitly set the MAX_STRING_SIZE= LIBNAME option to the value of the MAX_STRING_SIZE parameter that is used by your Oracle database.

max_array_len= Data Connector Option

specifies the maximum buffer length for complex data types.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Alias: maxArrayLen=

Default: 2000

Data source: DuckDB

Note: Support for this option was added in 2026.09.

Syntax

max_array_len=*value*

Required Argument

value
specifies the maximum buffer length for complex data types, such as arrays, lists, or structs. This buffer is used to construct a string representation of the complex type.

Details

This option is used when you query complex data types, such as arrays, lists, or structs. The query results are represented with character strings. It is possible to query complex data types that contain additional complex types. The resulting character strings mimic output from querying complex types with the DuckDB CLI.

maxErrors= Data Connector Option

specifies the maximum number of rows that can be rejected before the bulk-loading process ends.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: 0

Data source: Microsoft SQL Server

Syntax

maxErrors=*number*

Required Argument

number
specifies the maximum number of rows that can be rejected before the bulk-loading process ends.

Details

This option is applicable only when bulk loading data to Microsoft Azure or Microsoft Fabric Data Warehouse. For more information, see "Bulk Loading to Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse" on page 241.

maxLoadThreads= Data Connector Option

specifies the maximum number of threads to use per worker node.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Default: 24

Restriction: This option applies to parallel data transfer only.

Data source: Hadoop, Spark, SPD Engine files, Teradata

Syntax

maxLoadThreads=*maximum-number-of-threads*

Required Argument

maximum-number-of-threads
specifies the maximum number of threads to use per worker node.

maxSaveThreads= Data Connector Option

specifies the maximum number of threads to use per worker node when you write data to the data source.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: 24

Restriction: This option applies to parallel data transfer only.

Data source: Hadoop, Spark, SPD Engine files, Teradata

Syntax

maxSaveThreads=*maximum-number-of-threads*

Required Argument

maximum-number-of-threads
specifies the maximum number of threads to use per worker node when you write data to your data source.

mdfPath= Data Connector Option

specifies the HDFS or local disk directory that contains the SPD Engine MDF file.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Default: none

Data source: SPD Engine Files

Syntax

mdfPath=*"MDF-file-directory"*

Required Argument

MDF-file-directory
specifies the HDFS or local disk directory that contains the SPD Engine MDF file.

memoryLimit= Data Connector Option

specifies the memory limit for the DuckDB buffer manager.

Valid in:	CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements
Alias:	memory_limit=
Default:	80% of system reported memory
Data source:	DuckDB
Note:	Support for this option was added in 2026.09.
See:	"SAS_DUCKDB_MEMORY_LIMIT Environment Variable" in *SAS Global Statements: Reference*
Example:	`memoryLimit="8GiB"`

Syntax

memoryLimit=*"value-and-units"*

Required Argument

value-and-units
specifies the memory limit for the DuckDB buffer manager and the corresponding units, such as M, MB, MiB, and so on.

Details

This data connector option overrides any value that you set for the SAS_DUCKDB_MEMORY_LIMIT environment variable.

mode= Data Connector Option

specifies whether to enable multiple performance-related options.

Valid in:	CASLIB statement, PROC CASUTIL: LOAD and SAVE statements

Default: STANDARD

Data source: Google BigQuery

See: ignoreFedSQLObjects=, numReadNodes=, numWriteNodes=, project=, readMode=, scanStringColumns=

Syntax

mode="STANDARD" | "PERFORMANCE"

Required Arguments

STANDARD
specifies to use default values for performance-related data connector options.

PERFORMANCE
specifies that you want to enable several performance-related data connector options.

Details

When you set mode="PERFORMANCE", you automatically set the following option values:

- ignoreFedSQLObjects=TRUE
- numReadNodes=0 (use all available nodes to load data to CAS)
- numWriteNodes=0 (use all available nodes to write to your data source)
- project="" (empty string)
- readMode=STORAGE
- scanStringColumns=TRUE

To override any of these values, explicitly specify that option and assign a different value. For example, if you specify mode=PERFORMANCE and scanStringColumns=FALSE, then the scanStringColumns= option is disabled.

multipassMemory= Data Connector Option

specifies whether to use a temporary memory cache for multipass processing.

Valid in: CASLIB statement

Default: cache

Interactions: If backingStore="CASDISKCACHE", then this option is ignored.
If a table has already been loaded to the CAS server, then this option is ignored.

If you run a multipass action with multipassMemory="cache", and the input data is a view loaded with createViewSnapshot="never", the CAS Disk Cache must have enough space to hold the data to be analyzed.

Data source: SAS SpeedyStore

Syntax

multipassMemory="cache" | "stream"

Required Arguments

cache
specifies to use the CAS disk cache as a temporary memory cache.

stream
specifies to not use a cache.

Details

Overview

To optimize performance in SAS SpeedyStore, you have three choices for memory use:

- The multipassMemory="cache" value is best for customers who need high performance and do not need to limit the use of the CAS disk cache.
- The multipassMemory="stream" value helps to conserve memory use, at the cost of performance. If multiple actions operate on the same table, performance is further reduced.
- Set backingStore="CASDISKCACHE" to completely avoid streaming. This choice is best for moderately sized tables that can completely fit in CAS memory. This choice can also be appropriate when multiple users need access to most or all of a global table.

> **TIP** You can assign multiple caslibs to the data source, in order to specify different options for different use cases. You can also override the caslib setting by specifying multipassMemory= or backingStore= in the dataSourceOptions= parameter of the Table.loadTable action.

The following topics provide the details for each choice. If performance is a high priority, you are advised to test these options in your environment under typical conditions.

Using the Multipass Temporary Cache

When you set multipassMemory="cache", you use the CAS disk cache as a temporary memory cache.

- This feature is available if backingStore="INPLACEPREFERRED", which is the default setting.
- When you use a CAS action that performs multipass analytics, the requested data is loaded into the CAS disk cache. The data is released from the cache after the action completes. The cached data is *session-private*, which means the data is available to the current CAS session and the current action.
- Because the table is stored in SingleStore and is only temporarily cached, you can take advantage of in-database processing. You can filter the data that is loaded from SingleStore by applying a WHERE clause, by selecting columns, or by using computed columns. The performance improvement is most dramatic for filtered data from very large tables.
- The createMultipassColumn= option is ignored. The multipass column (also known as an auto_increment column) is not used. If a multipass column exists, it is ignored.
- The temporary memory cache is not used for single-pass processing.
- The option uses the CAS disk cache location that you have configured for the SAS Viya platform. For more information, see "About the Disk Cache" in *SAS Cloud Analytic Services: Fundamentals*.
- You can limit the amount of CAS_DISK_CACHE space that is used for global caslibs. That setting is honored by this option. For more information, see "Global caslib Policy" in *SAS Viya Platform: SAS Cloud Analytic Services*.
- If the temporary memory cache exceeds the available resources, error messages are written to the log and SAS attempts to run the action again. On this attempt, SAS uses streaming and does not use the CAS disk cache.

Using Streaming and the Multipass Column Feature

If you set multipassMemory="stream", then the temporary memory cache is not used. The multipass column feature is used instead.

- This feature is available if backingStore="INPLACEPREFERRED", which is the default setting.
- Multipass actions cause the entire table to be streamed for each pass.
- For streaming and multipass analytics, you can use the saveMultipassColumn=TRUE option to permanently add a multipass column. For more information, see "saveMultipassColumn= Data Connector Option" on page 563.
- The default behavior in the createMultipassColumn= option depends on the setting of the saveMultipassColumn= option. For more information, see "createMultipassColumn= Data Connector Option" on page 455.

Using the Traditional CAS Disk Cache

If you set backingStore="CASDISKCACHE" in the caslib assignment, then the multipassMemory= option is ignored. The backingStore="CASDISKCACHE" value turns streaming off, regardless of multipass analytics.

- When you load a table to CAS, the table stays in the CAS disk cache until the session ends (for a session table) or until the server shuts down (for a promoted table).

See Also

- "backingStore= Data Connector Option" on page 409
- "createMultipassColumn= Data Connector Option" on page 455
- "saveMultipassColumn= Data Connector Option" on page 563

mutual_auth= Data Connector Option

specifies whether to enforce mutual authentication.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Default: NO

Requirement: When mutual_auth=YES, you must set the path to your local certificate (cert_path=) and a password (cert_pass=).

Data source: Salesforce

Syntax

mutual_auth=YES | NO

Required Argument

YES | NO
specifies whether to enforce mutual authentication.

numDataFiles= Data Connector Option

specifies the number of data streams or data files for bulk loading into ADLS2 storage.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: 4

Range: 1 – 8

Data source: Microsoft SQL Server

Note: Support for this option was added in 2026.05.

Syntax

numDataFiles=*value*

Required Argument

value
specifies the number of data streams or data files to use to bulk load uncompressed CSV files directly into ADLS2 storage.

Details

Starting in 2026.05, uncompressed CSV files can be streamed directly into ADLS2 storage. This removes the intermediate step in which temporary files are created prior to copying that data into ADLS2. For this reason, using this option typically improves bulk load performance for these files.

numReadBuffers= Data Connector Option

specifies the number of buffers that the data connector is to read and convert from the data source to CAS.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Default: 4

Range: 1-10

Restriction: *Hadoop, Spark:* This option applies to serial data transfer only.

Data source: Amazon Redshift, DB2, DuckDB, Google BigQuery, Greenplum, Hadoop, Impala, Informix, JDBC, Microsoft SQL Server, MongoDB, MySQL, Netezza, ODBC, Oracle, PostgreSQL, SAP HANA, SAP IQ, SAS Data Sets, SAS Federation Server, SingleStore standard, Snowflake, Spark, Teradata, Vertica, Yellowbrick

Note: Support for DuckDB was added in 2026.09.

Syntax

numReadBuffers=*number*

Required Argument

number
specifies the number of buffers that the data connector is to read and convert from the data source to CAS.

numReadNodes= Data Connector Option

specifies the number of CAS worker nodes to use for a multinode load of data into CAS.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Defaults: 1 [DB2, DuckDB, Google BigQuery, Hadoop, Impala, JDBC, MongoDB, MySQL, Netezza, ODBC, Oracle, Salesforce, SAP HANA, SAP IQ, SAS Federation Server, SingleStore standard, Spark, Teradata, Vertica, Yellowbrick]

0 (use all nodes) [Amazon Redshift, Greenplum, Informix, Microsoft SQL Server, PostgreSQL, Snowflake, SPD Engine files]

Range: 0–32767

Restriction: *Hadoop, Spark, SPD Engine files, Teradata:* This option applies to serial data transfer only.

Requirements: *Salesforce:* If you set numReadNodes= to a value that is greater than 1, then you must also set sliceExpressions=. Otherwise, numReadNodes= resets to its default.

SAS Federation Server: Multinode read is not supported with the FedSQL execDirect action. Data load proceeds with numReadNodes=1.

Data source: Amazon Redshift, DB2, DuckDB, Google BigQuery, Greenplum, Hadoop, Impala, Informix, JDBC, Microsoft SQL Server, MongoDB, MySQL, Netezza, ODBC, Oracle, PostgreSQL, Salesforce, SAP HANA, SAP IQ, SAS Federation Server, SingleStore standard, Snowflake, Spark, SPD Engine files, Teradata, Vertica, Yellowbrick

Note: Support for DuckDB was added in 2026.09.

See: sliceColumn= data connector option

Syntax

numReadNodes=*number*

Required Argument

number
specifies the number of CAS worker nodes to use for a multinode load of data into CAS.

Details

Setting this option to a value other than 1 lets a data connector use multiple nodes to load data into CAS. You can specify any value up to the total number of available worker nodes. If the available number of nodes is smaller than the value that you specify, all available nodes are used. Specify a value of 0 to use all available worker nodes.

The default value for numReadNode= is 0 (use all nodes) for Amazon Redshift, Greenplum, Informix, Microsoft SQL Server, PostgreSQL, and Snowflake. To disable multinode data loads, set numReadNodes=1.

To specify the column to use for partitioning the data, use the sliceColumn= data connector option. Otherwise, SAS scans the data for the first numeric data type that can be used to partition the data. For more information, see "About Loading Data When Using Multinode Data Transfer" on page 179.

numWriteBuffers= Data Connector Option

specifies the number of buffers that the data connector uses to write and convert data from CAS to the data source.

Valid in:	CASLIB statement, PROC CASUTIL: SAVE statements
Default:	10
Range:	1-100
Restriction:	This option applies to serial data transfer only.
Data source:	SPD Engine files

Syntax

numWriteBuffers=*number*

Required Argument

number
specifies the number of buffers that the data connector uses to write and convert data from CAS to the data source.

numWriteNodes= Data Connector Option

specifies the number of CAS worker nodes to use for a multinode save of data from CAS into a data source table.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Defaults: 0 (use all nodes) [Amazon Redshift, Greenplum, Informix, Microsoft SQL Server, PostgreSQL, Snowflake]

1 [DB2, DuckDB, Google BigQuery, Hadoop, Impala, JDBC, MongoDB, MySQL, Netezza, ODBC, Oracle, Salesforce, SAS Federation Server, SAP HANA, SAP IQ, SingleStore standard, Spark, Teradata, Vertica, Yellowbrick

Range: 0–32767

Restriction: *Hadoop, Spark, Teradata:* This option applies to serial data transfer only.

Data source: Amazon Redshift, DB2, DuckDB, Google BigQuery, Greenplum, Hadoop, Impala, Informix, JDBC, Microsoft SQL Server, MongoDB, MySQL, Netezza, ODBC, Oracle, PostgreSQL, Salesforce, SAS Federation Server, SAP HANA, SAP IQ, SingleStore standard, Snowflake, Spark, Teradata, Vertica, Yellowbrick

Note: Support for DuckDB was added in 2026.09.

Syntax

numWriteNodes=*number*

Required Argument

number
specifies the number of CAS worker nodes to use for a multinode save of data from CAS into a data source table.

Details

Setting this option to a value other than 1 lets a data connector use multiple nodes to save data to a data source table. You can specify any value up to the total number of available worker nodes. If the available number of nodes is smaller than the value that you specify, all available nodes are used.

Specify a value of 0 to use all available worker nodes.

The default value for numWriteNodes= is 0 (use all nodes) for Amazon Redshift, Greenplum, Informix, Microsoft SQL Server, PostgreSQL, and Snowflake. To disable multinode Writes, set numWriteNodes= to 1.

ora_encoding= Data Connector Option

specifies the encoding of the data in the Oracle database.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Default: UNICODE

Data source: Oracle

Tip: If the source data is encoded in UTF-8, then data does not need to be transcoded into UTF-8 when loading it into CAS. This reduces the time it takes to load the data.

Syntax

ora_encoding=*"encoding-name"*

Required Argument

encoding-name
specifies the encoding of the data in the Oracle database.

Valid values include LATIN1, WLATIN1, and UNICODE. Set this value to UNICODE when you are loading non-Latin1 data.

Details

This value is independent of the NLS_LANG environment variable setting. The data is transcoded from the database encoding into UTF-8 when data is loaded into CAS.

padCompress= Data Connector Option

specifies the amount of padding to add when you use the compress= option.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: 0

Restriction: This option applies to serial data transfer only.

Data source: SPD Engine files

Syntax

padCompress=*number*

Required Argument

number
specifies the amount of padding to add when you use the compress= option.

partSize= Data Connector Option

specifies the size of partitions to use for a SPD Engine table when it is created by using the SAVE statement in PROC CASUTIL.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: none

Requirement: This option applies to serial data transfer only.

Data source: SPD Engine files

Syntax

partSize=*number*

Required Argument

number
specifies the size of partitions to use for a SPD Engine table when it is created using the SAVE statement in PROC CASUTIL.

password= Data Connector Option

specifies the data source password for a user.

Valid in: CASLIB statement [except SAS data sets], PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Category: Authentication

Aliases: PASS= [All data sources except SAS data sets]
PWD= [All data sources except SAS data sets]

Default: none

Restrictions: *Hadoop, Spark:* When a Kerberos connection is used, no values should be specified for userName= or password=.

SAS data sets: Use this option only if the SAS data set is password-protected or uses SAS proprietary encryption.

SPD Engine files: This option is valid only when SPD Engine files are stored in Hadoop.

SPD Engine files: Parallel processing has limited support for SPD table passwords.

Requirement: If your data source requires authentication, you must specify valid credentials to access data. You can provide these credentials by specifying either userName= and password= values or by specifying an authenticationDomain= value.

Data source: Amazon Redshift, DB2, Greenplum, Hadoop, Impala, Informix, JDBC, Microsoft SQL Server, MongoDB, MySQL, Netezza, ODBC, Oracle, PostgreSQL, Salesforce, SAP HANA, SAP IQ, SAS data sets, SAS Federation Server, SAS SpeedyStore, SingleStore standard, Snowflake, Spark, SPD Engine files, Teradata, Vertica, Yellowbrick

Note: The userName= and password= options remain valid. However, these options are less secure because you are saving authentication information in your programs. Therefore, as a better security practice, it is recommended that you use the authenticationDomain= option instead.

Syntax

password=*"password"*

Required Argument

password
specifies the data source password for a user.

Details

You typically specify userName= and password= values (or an authenticationDomain= value) when you add a caslib. These credentials are then used for any statement that accesses the data using that caslib. If you specify a userName= and password= value in a statement other than the CASLIB statement, these credentials override any credentials that were specified when the caslib was added.

MongoDB: The authenticationDomain= option is not supported by MongoDB, although it is discussed above as an alternative to the userName= and password= options.

PostgreSQL: If you are connecting to PostgreSQL on Microsoft Azure, the password= value is the security token that you use to authenticate to Azure. For more information, see "Authenticate to Microsoft Azure Using Single Sign-On" in *SAS/ACCESS for Relational Databases: Reference*.

SAS Data Sets: This option specifies the password for a password-protected SAS data set.

SPD Engine files: This option specifies the HDFS password for a user.

path= Data Connector Option

specifies a SAS data set name, a PC files source table, or an Oracle driver, node, and database.

Valid in: CASLIB statement [all], PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements [Oracle]

Default: none

Requirements: *Oracle:* This option is required for the CASLIB statement and the LOAD statement for PROC CASUTIL.

Oracle: Aliases are required if you are using SQL*Net Version 2.0 or later.

PC Files: Although no data source options are required when you work with Microsoft Excel files, you must use a caslib that specifies srcType="path".

SAS data sets: You must specify `srcType="PATH"` in the dataSource= argument when you add a caslib.

SAS data sets: You can specify the SAS7BDAT extension for a file name. However, if you specify filetype="BASESAS", then you can omit this extension for path= when you load data.

Data source: Oracle, PC Files, SAS data sets

Example:

```
caslib oraclecaslib desc='Oracle Caslib'
   datasource=(srcType='oracle'
               username='myuser1'
               password='myPwd'
               path="//machine.lnx.com:1521/exadat");
```

Syntax

path=*"input-data-specifics"*

Required Argument

input-data-specifics
specifies information about input data.

Details

When you use this option with the CONTENTS statement for PROC CASUTIL, you can use wildcards to match a specific set of file names. For more information, see "Using Wildcard Characters for File Name Matching" on page 89.

Oracle: This option specifies the driver, node, and database.

PC Files: This option specifies the location and name of a PC Files source table. Although no data source options are required when you work with Microsoft Excel files, you must use a caslib with srcType="PATH".

SAS data sets: This option specifies an optional subdirectory and SAS data set name. For SAS data sets, you must specify srcType="PATH" and also specify the path= option.

For SAS data sets, when you use the LOAD statement from PROC CASUTIL for a caslib of type path, the value of this option is appended to the path= value that you specified in the CASLIB statement. The value of this option for the PROC CASUTIL LOAD statement can be a file name or a subdirectory and file name. When used in this context, SAS/ACCESS assumes that the caslib was defined to allow subdirectories.

platform= Data Connector Option

specifies the platform that applies to a data source connection.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements [Hadoop, SPD Engine files]

Defaults: spark [Hadoop, SPD Engine files]

none [MongoDB, Spark]

Data source: Hadoop, MongoDB, Spark, SPD Engine files

Syntax

Form 1: Hadoop and SPD Engine files:

platform="spark"

Form 2: MongoDB:

platform="atlas"

Form 3: Spark:

platform=databricks | spark | synapse

Required Arguments

atlas

specifies to connect to a MongoDB data source on an Atlas cloud platform. If the platform= option is not specified for MongoDB, then a connection to a traditional MongoDB database is made.

Note This value is available only for MongoDB.

databricks
SAS Data Accelerator (known as SAS In-Database Technologies for Databricks prior to 2026.06): specifies to run scoring models in Databricks using SAS Embedded Process for Spark. see "Running Scoring Models in Databricks Using PROC SCOREACCEL or CAS Actions" in *SAS In-Database Products: User's Guide*.

Spark Data Connector without SAS Data Accelerator: Starting in 2023.10, this option is no longer necessary. The Spark data connector uses the content of the URI= or SERVER= value to determine the driver assignment.

mapred
specifies to run the SAS Embedded Process job as a MapReduce job.

Note For parallel data transfer in Hadoop or SPD Engine files, platform=spark is always used. If you specify platform=mapred, the data connector prints a warning to the SAS log and uses Spark instead of MapReduce. For parallel data transfer, you can remove the platform= option from your code.

spark
SAS In-Database Technologies for Hadoop: specifies to run the SAS Embedded Process job on a Spark platform.

Note For parallel data transfer in Hadoop or SPD Engine files, platform=spark is always used. If you specify platform=mapred, the data connector prints a warning to the SAS log and uses Spark instead of MapReduce. For parallel data transfer, you can remove the platform= option from your code.

Spark Data Connector without SAS In-Database Technologies for Hadoop: Starting in 2023.10, this option is no longer necessary. The Spark data connector uses the content of the URI= or SERVER= value to determine the driver assignment.

synapse
specifies to run scoring models in Synapse using SAS Embedded Process for Spark. For more information, see "Running Scoring Models in Synapse Using PROC SCOREACCEL or CAS Actions" in *SAS In-Database Products: User's Guide*.

pointObs= Data Connector Option

specifies whether to locate a data set by returning an observation (Obs) number from a subset of observations.

Valid in: PROC CASUTIL: SAVE statement

Default: none

Data source: SAS data sets

Syntax

pointObs="YES" | "NO"

Required Argument

"YES" | "NO"
specifies whether to locate a data set by returning an observation (Obs) number from a subset of observations.

port= Data Connector Option

specifies the port value to use to connect to a data source.

Valid in: CASLIB statement

Defaults:
5432 [Greenplum]
10000 [Hadoop]
21050 [Impala]
none [Informix]
27017 [MongoDB]
none [MySQL]
5480 [Netezza]
5432 [PostgreSQL]
30015 [SAP HANA]
none [SAP IQ]
none [SAS Federation Server]
3306 [SAS SpeedyStore]
none [SingleStore standard]
10016 [Spark]
443 [Databricks]
5433 [Vertica]
none [Yellowbrick]

Range: 1–65535

Restriction: *SAP HANA:* Specify a value for only instance= or port=, not both. (As an alternative to port=, you can instead specify server="*server-identifier:port*.) If you specify a value for both instance= and port= (or a server= value that includes port), then a connection error results.

Requirement: *Informix, SAS Federation Server:* This option is required in the CASLIB statement.

Data source: Greenplum, Hadoop, Impala, Informix, MongoDB, MySQL, Netezza, PostgreSQL, SAP HANA, SAP IQ, SAS Federation Sever, SAS SpeedyStore, SingleStore standard, Spark, Vertica, Yellowbrick

Syntax

port=*port-value*

Required Argument

port-value
specifies the port value to use for a data connector.

Details

Hadoop: The port= value specifies the Hive JDBC port number.

Spark: The port= value specifies the port number for the Spark SQL database.

Databricks: The port= value specifies the port number for the Databricks database.

post_table_opts= Data Connector Option

specifies additional database-specific options to be placed after the table name when creating a table.

Valid in:	CASLIB statement
Default:	none
Restriction:	This option applies to serial data transfer only.
Data source:	Hadoop

Syntax

post_table_opts=*"option-string"*

Required Argument

option-string
specifies additional database-specific options to be placed after the table name when creating a table.

pre_table_opts= Data Connector Option

specifies additional database-specific options to be placed before the table name when creating a table.

Valid in: CASLIB statement

Default: none

Restriction: This option applies to serial data transfer only.

Data source: Hadoop

Syntax

pre_table_opts=*"option-string"*

Required Argument

option-string
specifies additional database-specific options to be placed before the table name when creating a table.

profile= Data Connector Option

specifies the profile to use when there is more than one profile in the AWS configuration file.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: default

Requirements: *Amazon Redshift:* To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.
Snowflake: To specify this option, you must also set bulkLoad=TRUE.

Data source: Amazon Redshift, Snowflake

Syntax

profile=*"profile-name"*

Required Argument

profile-name
specifies the profile to use when there is more than one profile in the AWS configuration file.

Details

When more than one profile is used in the AWS configuration file, the profile name precedes the set of name-value pairs in square brackets. Here is an example configuration file with two profiles:

```
[default]
region=us-west-2
output=text

[analyst]
region=us-east-1
output=json
```

project= Data Connector Option

specifies the project ID for a Google BigQuery project.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Default: none

Requirement: This option is required in the CASLIB statement.

Data source: Google BigQuery

Example: `project='project1'`

Syntax

project=*"project-ID"*

Required Argument

project-ID
specifies the project ID for a Google BigQuery project.

properties= Data Connector Option

specifies additional driver-specific JDBC connection properties that will be added to the generated JDBC URL.

Valid in: CASLIB statement

Default: none

Interaction: If you use both the schema= option and the properties= option to specify schemas, the behavior could be unexpected. You can use one schema as the current_schema

for connecting to the data source and a different schema for SQL generation. Although this usage is valid, it could cause confusion. To ensure consistent behavior, it is recommended to specify one schema using the schema= option only. For an example that demonstrates this behavior in the LIBNAME statement, see "Specifying Different Schemas Using the PROPERTIES= Option and the SCHEMA= Option" in *SAS/ACCESS for Relational Databases: Reference*.

Data source: Hadoop, Spark

Note: The properties= option is ignored if you also specify the uri= option.

Example:
```
properties=
   "hive.exec.dynamic.partition.mode=nonstrict;
    hive.exec.dynamic.partition=true"
```

Syntax

properties=*"JDBC-properties-value"*

Required Argument

JDBC-properties-value
specifies a free-form value for JDBC properties. The value is appended to the JDBC connection URI. You can use it to override default Apache Hive, Apache Spark, and Databricks driver behaviors.

Details

When the uri= option is not provided, SAS constructs a JDBC URL from SAS data connector options that are specified in the CASLIB statement. By specifying properties=, you can include additional connection properties in the generated URL.

JDBC properties are specific to the driver that is used to make the connection.

For information about the properties that are available for the Simba drivers that are redistributed with the SAS Viya platform, see your Simba documentation.

proxy= Data Connector Option

specifies the URL of a proxy server that is used to connect to Google BigQuery.

Valid in: CASLIB statement

Default: none

Data source: Google BigQuery

Example:
```
proxy="mycompany/proxy.htm"
```

Syntax

proxy=*"proxy-server"*

Required Argument

proxy-server
specifies the URL of a proxy server that is used to connect to Google BigQuery.

proxy_host= Data Connector Option

specifies the host name or IP address of a proxy server.

Valid in:	CASLIB statement
Alias:	proxyHost=
Default:	none
Data source:	Salesforce
Tip:	You might use the (optional) proxy options to request a connection from and allow login access to Salesforce through a specific IP address using a proxy server.

Syntax

proxy_host=*"host-name"*

Required Argument

host-name
specifies the host name or IP address of a proxy server.

proxy_pass= Data Connector Option

specifies the password that is associated with proxy_user=.

Valid in:	CASLIB statement
Alias:	proxyPass=
Default:	none
Data source:	Salesforce
Tip:	You might use the (optional) proxy options to request a connection from and allow login access to Salesforce through a specific IP address using a proxy server.

Syntax

proxy_pass=*"value"*

Required Argument

value
specifies the password that is associated with proxy_user=.

proxy_port= Data Connector Option

specifies the port number of the proxy server.

Valid in:	CASLIB statement
Alias:	proxyPort=
Default:	8080
Data source:	Salesforce
Tip:	You might use the (optional) proxy options to request a connection from and allow login access to Salesforce through a specific IP address using a proxy server.

Syntax

proxy_port=*"port-number"*

Required Argument

port-number
specifies the port number of the proxy server.

proxy_user= Data Connector Option

specifies the user ID that is used to access the proxy server.

Valid in:	CASLIB statement
Alias:	proxyUser=
Default:	none
Data source:	Salesforce
Tip:	You might use the (optional) proxy options to request a connection from and allow login access to Salesforce through a specific IP address using a proxy server.

Syntax

proxy_user=*"user-id"*

Required Argument

user-id
specifies the user ID that is used to access the proxy server.

query_tag= Data Connector Option

specifies an optional string that you can use to tag queries to your data source.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Alias: queryTag=

Default: none

Data source: Snowflake

Syntax

query_tag=*"string"*

Required Argument

string
specifies an optional string that you can use to tag queries to your data source. The string can be up to 2000 characters.

query_timeout= Data Connector Option

specifies the amount of time in seconds to wait for a query response.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Alias: queryTimeout=

Default: none

Range: 0 (minimum)

Data source: JDBC

Syntax

query_timeout=*seconds*

Required Argument

seconds
specifies the amount of time in seconds to wait for a query response.

range= Data Connector Option

specifies the range of data (a subset of cells) within a worksheet to import.

Valid in: CASLIB statement

Default: none

Restriction: For XLSX files (fileType="EXCEL"), do not include the worksheet name in the range.

Requirement: To use this option, you must specify a range that represents a rectangle in the worksheet.

Data source: PC Files

Examples: A17–D51

For XLS files, the range "Sheet1$A1:B5" is the range address for a rectangular block of 10 cells. The top left cell is A1, and the bottom right cell is B5.

Syntax

range=*"cell-range"*

Required Argument

cell-range
specifies the range of data (a subset of cells) within a worksheet to import. The range can be absolute (for XLS and XLSX files) or named (for XSLX). For details, refer to *SAS/ACCESS Interface to PC Files: Reference for the SAS Viya Platform*.

Details

If no range is specified, then SAS imports the entire worksheet.

read= Data Connector Option

specifies the password to access and read a password-protected data set.

Default: none

Requirement: This option is required only if a data set is password-protected.

Data source: SAS data sets

Syntax

read=*"password"*

Required Argument

password
specifies the password to access and read a password-protected data set.

readBuff= Data Connector Option

specifies the number of rows to fetch per block of data retrieved.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Aliases: ras=
row_array_size=

Default: Calculated automatically based on row size

Restriction: *Hadoop, Spark, Teradata:* This option is not valid with parallel data transfer (dataTransferMode="PARALLEL"). Specify bufferSize= when loading data in parallel.

Data source: Amazon Redshift, DB2, DuckDB, Google BigQuery, Greenplum, Hadoop, Impala, Informix, JDBC, Microsoft SQL Server, MongoDB, MySQL, Netezza, ODBC, Oracle, PostgreSQL, SAP HANA, SAP IQ, SingleStore standard, Spark, Teradata, Vertica, Yellowbrick

Note: Support for DuckDB was added in 2026.09.

Syntax

readBuff=*number*

Required Argument

number
specifies the number of rows to fetch per block of data retrieved.

readMode= Data Connector Option

specifies the method to use when moving data from Google BigQuery into CAS.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Default: STANDARD

Data source: Google BigQuery

Syntax

readMode=STANDARD | STORAGE

Required Arguments

STANDARD
use the default SAS/ACCESS method to move data into CAS.

STORAGE
use the Storage API for Google to move data into CAS.

readOnly= Data Connector Option

specifies whether to connect to your database in Read mode (TRUE) or Read-Write (FALSE) mode.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Default: FALSE

Data source: DuckDB

Note: Support for this option was added in 2026.09.

Syntax

readOnly=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether to connect to the database in Read mode (TRUE) or Read-Write (FALSE) mode.

readPW= Data Connector Option

specifies the password to access and read a password-protected SPD Engine table.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, LOAD, and SAVE statements

Default: none

Restriction: This option applies to parallel data transfer when saving to a newly created table. This option applies to serial data transfer when you load data into CAS or when you save to an existing SPD Engine table.

Data source: SPD Engine files

Syntax

readPW=*"password"*

Required Argument

password
specifies the password to access and read a password-protected SPD Engine table.

refresh_token= Data Connector Option

specifies the refresh token value for the OAuth client that is used to connect to Google BigQuery using the OAuth authentication method.

Valid in: CASLIB statement

Default: none

Interaction: This option is used in conjunction with the client_id= and client_secret= options.

Data source: Google BigQuery

Syntax

refresh_token=*"token-value"*

Required Argument

token-value
specifies the refresh token value for the OAuth client that is used to connect to Google BigQuery using the OAuth authentication method.

region= Data Connector Option

specifies the AWS region for Amazon S3 data.

Valid in: CASLIB statement, PROC CASUTIL: LOAD, SAVE statements

Default: none

Requirements: To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.
You must specify the region when working with data in Amazon S3.

Data source: Amazon Redshift, Snowflake

Example:
```
region="us-east-1"
```

Syntax

region=*"region"*

Required Argument

region
specifies the AWS region for Amazon S3 data.

Details

Use the regions values that are available for the Amazon S3 service.

remoteDriverOptions= Data Connector Option

specifies the driver options for the REMFS driver, which is used to connect to SAS Federation Server.

Valid in: CASLIB statement

Default: none

Data source: SAS Federation Server

Syntax

remoteDriverOptions=*"options"*

Required Argument

options
specifies the driver options for the REMFS driver, which is used to connect to SAS Federation Server.

Details

For the list of supported options, see "REMOTE_DRIVER_OPTIONS= LIBNAME Statement Option" in *SAS LIBNAME Engine for SAS Federation Server: User's Guide*.

replace= Data Connector Option

specifies that a new file with a given name replaces an existing file with the same name.

Default: FALSE

Data source: SAS data sets

Syntax

replace=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies that a new file with a given name replaces an existing file with the same name.

reuse= Data Connector Option

specifies whether to write new rows to freed space in a compressed data set.

Valid in: PROC CASUTIL: SAVE statement

Default: NO

Interaction: You can set this option when you use the compress= option.

Data source: SAS data sets

Syntax

reuse="YES" | "NO"

Required Argument

"YES" | "NO"
specifies whether to write new rows to freed space in a compressed data set.

role= Data Connector Option

specifies a the role name to use when connecting to a data source.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Default: none

Restriction: *Teradata:* This option applies to serial data transfer only (dataTransferMode="SERIAL").

Data source: Snowflake, Teradata

Syntax

role=<">*name*<">

Required Argument

name
specifies a defined role name.

Details

Snowflake: The role that you specify should be one that you have assigned to the user. If the specified role does not match any role that has been assigned to the user, then the connection does not include a role.

s3AccessKey= Data Connector Option

specifies your AWS access key.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Aliases: s3_key=
s3_key_id=

Default: none

Data source: DuckDB

Note: Support for this option was added in 2026.09.

See: s3Region= on page 562 s3Secret= on page 562 s3Token= on page 563

Syntax

s3AccessKey=*"access-key"*

Required Argument

access-key
specifies your Amazon Web Services access key that is used with key-based access control.

s3Config= Data Connector Option

specifies the location of a local configuration file for Amazon S3.

Valid in: CASLIB statement, PROC CASUTIL: LOAD, SAVE statements

Default: none

Requirements: To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.
You must specify this option when you work with data in S3.

Data source: Amazon Redshift, Snowflake

Syntax

s3Config=*"path-and-configuration-file-name"*

Required Argument

path-and-configuration-file-name
specifies the location of a local configuration file for Amazon S3.

Details

For more information about Amazon S3 configuration files, see the Amazon Web Services documentation.

s3Region= Data Connector Option

specifies your AWS region.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Alias: s3_region=

Default: none

Requirement: This option is required to connect to an S3 bucket in AWS.

Data source: DuckDB

Note: Support for this option was added in 2026.09.

See: s3AccessKey= on page 560 s3Secret= on page 562 s3Token= on page 563

Syntax

s3Region=*"region"*

Required Argument

region
specifies the Amazon Web Services region where your data is stored.

s3Secret= Data Connector Option

specifies your AWS secret access key or a temporary secret access key.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Alias: s3_secret=

Default: none

Data source: DuckDB

Note: Support for this option was added in 2026.09.

See: s3AccessKey= on page 560 s3Region= on page 562 s3Token= on page 563

Syntax

s3Secret=*"secret-access-key"*

Required Argument

secret-access-key
specifies your Amazon Web Services secret access key or a temporary secret access key.

s3Token= Data Connector Option

specifies a temporary token for accessing AWS.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Aliases: s3_token=
s3_session_token=

Default: none

Interaction: The value that you provide corresponds with the values for s3AccessKey= and s3Secret=.

Data source: DuckDB

Note: Support for this option was added in 2026.09.

See: s3AccessKey= on page 560 s3Region= on page 562 s3Secret= on page 562

Syntax

s3Token=*"token"*

Required Argument

token
specifies a temporary token for accessing AWS.

saveMultipassColumn= Data Connector Option

specifies to create a multipass column when a table is saved.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: FALSE

Data source: SingleStore

Syntax

saveMultipassColumn=TRUE | FALSE

Required Arguments

TRUE
specifies to create the multipass column.

FALSE
specifies to not create the multipass column.

Details

To permanently add a multipass column, specify saveMultipassColumn=TRUE when you save a table to the data source. The multipass column can improve performance if your analysis uses multiple passes against a table, especially for a promoted (global) table. For details about how the multipass column is used, see .

The multipass column is a columnstore sort key that uses the SingleStore AUTO_INCREMENT feature. The multipass column improves the performance of parallel Read operations by reducing the memory footprint. The feature enables CAS to maintain a sort key across multiple reads.

Here are some details about the option:

- Setting saveMultipassColumn=TRUE adds the multipass column when a table is saved but not under other operations such as appending.
- The added column is not shown in the output of the table.columninfo action. However, if you query SingleStore directly, you can see the column.
- If a table is altered during CAS processing, such as dropping or appending rows, the multipass column is updated automatically. Additional actions continue to return the correct data.
- If an AUTO_INCREMENT column was previously created in the table by programming outside CAS, the column is ignored by this option. If you set saveMultipassColumn=TRUE, a new column is created.

Here is some guidance to optimize performance:

- Determine how often multipass processing is needed for the tables in a caslib. Multipass processing is common in analytics. If many tables in a caslib need multipass processing, then set saveMultipassColumn=TRUE for the caslib.
- Note that the TRUE setting can reduce performance for tables that do not need multipass processing. You can choose to set saveMultipassColumn= in the SAVE statement of the CASUTIL procedure (or in the table.save action) for a table. Setting the option for a table overrides the setting in the caslib.

- Some examples of single-pass processing include basic DATA steps, the summary action, and the fetch action. However, your single-pass processing step could be followed by multipass processing on the same table.

Comparisons

The data connector option saveMultipassColumn= affects a table when it is saved. The option createMultipassColumn= affects a table when it is loaded or when an action requires multiple passes, depending on the setting.

See Also

"createMultipassColumn= Data Connector Option" on page 455

scanStringColumns= Data Connector Option

specifies whether to scan all VARCHAR or long type columns in a table to determine the actual maximum length of the columns.

Valid in:	CASLIB statement, PROC CASUTIL: LOAD statement
Aliases:	scan_strings= scanStrings=
Defaults:	TRUE [DuckDB] FALSE [Snowflake] none [Google BigQuery, ODBC, Oracle, Teradata]
Data source:	Google BigQuery, DuckDB, ODBC, Oracle, Snowflake, Teradata
Note:	Support for DuckDB was added in 2026.09.

Syntax

scanStringColumns=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether to scan all VARCHAR or long type columns in a table to determine the actual maximum length of the columns.

Details

If VARCHAR or long type columns have been created with the maximum precision, using this option can reduce the size of the resulting CAS table and accelerate table loading.

Using this option might impact performance when loading smaller tables, because the data connector must execute a secondary SELECT statement to scan the VARCHAR columns. For larger tables, the performance impact of this secondary SELECT is offset with the reduction in the amount of data that is transferred from the data source.

The columns that are scanned for length depend on the data source and underlying driver that you use to access the data. Here is a list of the data types that are typically scanned for the data sources that use this option:

- Google BigQuery: STRING and BYTE columns
- ODBC: typically BLOB, CLOB, and NCLOB columns are scanned. However, this varies by data source. Here is a list of scanned data types for a sample of ODBC data sources:
  - ODBC to Dremio: VARCHAR and VARBINARY columns
  - ODBC to Microsoft SQL Server (with DataDirect driver): VARCHAR(max), NVARCHAR(max), VARBINARY(max) columns
  - ODBC to Oracle: BLOB, CLOB, NCLOB columns
- Snowflake: VARCHAR columns

schema= Data Connector Option

specifies the schema name to use for the connection to a data source.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, LIST, LOAD statements

Default: none

Interactions: *Amazon Redshift, DB2, DuckDB, Greenplum, Informix, MongoDB, MySQL, Netezza, PostgreSQL, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick:* If you specify values for schema= and database=, the value for schema= is used to qualify the table name.

Hadoop, Spark: database= is an alias for schema=.

Data source: Amazon Redshift, DB2, DuckDB, Google BigQuery, Greenplum, Hadoop, Impala, Informix, JDBC, Microsoft SQL Server, MongoDB, MySQL, Netezza, ODBC, Oracle, PostgreSQL, Salesforce, SAP HANA, SAP IQ, SAS Federation Server, SingleStore standard, Snowflake, Spark, Teradata, Vertica, Yellowbrick

Notes: The schema= location is the default location for storing temporary data that is generated by Hadoop parallel data transfer. To specify a different location, you can use the scratchDB= option.

Support for DuckDB was added in 2026.09.

Syntax

schema=*"schema-name"*

Required Argument

schema-name
specifies the schema name to use for the connection to the database or data source. If the schema name contains spaces or non-alphanumeric characters, enclose the value in quotation marks.

Details

When you specify a value for schema=, the table name is qualified with the schema name. For example, if you set schema="mySchema" and you want to access table Studydata, the table mySchema.Studydata is accessed. Specify a non-blank value when you set a value for schema=.

If you specify a value for schema= in a statement other than the CASLIB statement, this value overrides any value of schema= that was set when the caslib was added.

schema_auth= Data Connector Option

specifies the authorization method that is used to access a MongoDB schema.

Valid in: CASLIB statement

Default: auth= value (unless port= and schema_port= values differ or server= and schema_server= values differ. See Details.)

Data source: MongoDB

Syntax

schema_auth=AUTO | KERBEROS | LDAP | PASSWORD

Required Arguments

AUTO
specifies that SAS uses the default method of userName= and password= authentication to access a MongoDB schema.

KERBEROS
specifies that Kerberos authentication is used to access a MongoDB schema.

LDAP
specifies that LDAP authentication is used to access a MongoDB schema.

PASSWORD
specifies that userName= and password= values are used to access a MongoDB schema.

Details

The schema_auth= value inherits the value that is specified for auth= if the schema_port= value is the same as the port= value and if the schema_server= value is the same as the server= value. When the schema_port= value is not specified, schema_port= inherits the port= value. Similarly, when the schema_server= value is not specified, schema_server= inherits the server= value. If either the port or server values differ, then schema_auth= defaults to authentication that is based on the username= and password= values.

It is possible to use one authentication method when you connect to the MongoDB data source and a different method when you connect to a MongoDB schema. To use a different authentication method when connecting to MongoDB from that used when connecting to a schema, use the AUTH= data connector and this option.

schema_collection= Data Connector Option

specifies that a connection to a MongoDB schema should reference the stored schema file for the specified collection.

Valid in:	CASLIB statement
Default:	sas-default-schema
Restriction:	There can be only one schema per collection. However, there can be multiple collections (and therefore multiple schemas) in a database.
Data source:	MongoDB
Tip:	Creating a schema in this way creates a MongoDB collection with the same name as the schema alongside your operational data collections. Consider using the schema_db= option to keep the schema collection separate from your operational data.
See:	"Working with Schemas for MongoDB Data" in *SAS/ACCESS for Nonrelational Databases: Reference*

Syntax

schema_collection=*'collection-name'*

Required Argument

collection-name
specifies that a connection to a MongoDB schema should reference the stored schema file for the specified collection.

Details

If the specified collection does not already have a stored schema file, then a scan is performed and the schema file is created. You must have Write permission for the location where your schema is stored if you specify a new schema name. Schema names must be unique across MongoDB collections.

To generate a temporary schema that is held only in memory, specify SCHEMA_COLLECTION='$temp$'. For more information, see "Creating a Temporary Schema" on page 252.

schema_db= Data Connector Option

specifies the MongoDB database where a schema is stored.

Valid in:	CASLIB statement
Alias:	schema_database=
Default:	database= value
Requirement:	The user name specified in the CASLIB statement must have Write access to the database that is specified in the schema_db= option (unless you use the schema_user= and the schema_password= options).
Data source:	MongoDB

Syntax

schema_db=*"database-name"*

Required Argument

database-name
specifies the MongoDB database where a schema is stored.

schema_password= Data Connector Option

specifies the password to use to access a schema.

Valid in: CASLIB statement

Alias: schema_pwd=

Default: password= value

Data source: MongoDB

Syntax

schema_password=*'string'*

Required Argument

string
specifies the password to use to access a schema.

schema_platform= Data Connector Option

specifies the platform that applies to a connection to a MongoDB schema.

Valid in: CASLIB statement

Default: platform= value

Data source: MongoDB

Syntax

schema_platform="atlas"

Required Argument

atlas
specifies to connect to a MongoDB schema on an Atlas cloud platform.

schema_port= Data Connector Option

specifies the port to use to access a schema.

Valid in: CASLIB statement

Default: port= value

Data source: MongoDB

Syntax

schema_port=*value*

Required Argument

value
specifies the port to use to access a schema.

schema_server= Data Connector Option

specifies the server name or IP address of the server that stores the schema mapping.

Valid in: CASLIB statement

Default: server= value

Requirement: Use quotation marks if *server-name* contains spaces or nonalphanumeric characters or if the name is an IP address.

Data source: MongoDB

Syntax

schema_server=*<'>server-name<'>*

Required Argument

server-name
specifies the server name or IP address of the server that stores the schema mapping.

schema_ssl_ca= Data Connector Option

specifies the full path name to the certificate authority file for the MongoDB schema location.

Valid in: CASLIB statement

Default: ssl_ca= value

Interaction: If you use schema_ssl_ca= to connect to the MongoDB schema file, you must also use the ssl_ca= option to connect to the MongoDB database.

Data source: MongoDB

Syntax

schema_ssl_ca=*"path-and-file-name"*

Required Argument

path-and-file-name
specifies the full path name to the certificate authority file for the MongoDB schema location.

schema_ssl_cert= Data Connector Option

specifies the full path name to the certificate file for the MongoDB schema location.

Valid in: CASLIB statement

Default: ssl_cert= value

Interaction: If you use schema_ssl_cert= to connect to the MongoDB schema file, you must also use the ssl_cert= option to connect to the MongoDB database.

Data source: MongoDB

Syntax

schema_ssl_cert=*"path-and-file-name"*

Required Argument

path-and-file-name
specifies the full path name to the certificate file for the MongoDB schema location.

schema_uid= Data Connector Option

specifies the user ID to use to access a schema.

Valid in: CASLIB statement

Default: userName= value

Data source: MongoDB

Syntax

schema_uid=*'user-ID'*

Required Argument

user-ID
specifies the user ID to use to access a schema.

scratchDb= Data Connector Option

specifies the Hive database to use when creating temporary tables and views.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Default: none

Data source: Hadoop, Impala, Spark

Syntax

scratchDb=*"Hive-schema-name"*

Required Argument

Hive-schema-name
specifies the Hive database to use when creating temporary tables and views.

secret= Data Connector Option

specifies the secret access key or temporary secret access key that is used to transfer data in bulk.

Valid in: CASLIB statement [all], PROC CASUTIL: LOAD [Amazon Redshift], SAVE statement [all]

Default: none

Requirement: To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.

Data source: Amazon Redshift, Microsoft SQL Server, Snowflake

Tip: *Amazon Redshift, Snowflake:* To increase security, you can encode the key value by using PROC PWENCODE.

Syntax

secret=*"secret-access-key"*

Required Argument

secret-access-key
specifies the secret access key or temporary secret access key that is used to bulk load data.

Details

Amazon Redshift, Snowflake: An Amazon Web Services (AWS) secret access key is associated with the key ID that you specify with the key= option.

Microsoft SQL Server: The identity= value and the secret= value are supplied to the CREDENTIAL argument in the COPY command for Azure Synapse Analytics (SQL DW) or Microsoft Fabric Data Warehouse. This option is applicable only when bulk loading data to Microsoft Azure. For more information, see "Bulk Loading to Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse" on page 241.

server= Data Connector Option

specifies the server identifier for the data source server.

Valid in:	CASLIB statement
Default:	none
Restriction:	*Hadoop:* If your Hive server does not support high availability, you must specify server= for a Hive caslib.
Requirement:	*All data sources except Hadoop or Spark:* This option is required in the CASLIB statement.
Data source:	Amazon Redshift, Greenplum, Hadoop, Impala, Informix, MongoDB, Netezza, PostgreSQL, SAP HANA, SAP IQ, SAS Federation Server, Snowflake, Spark, Teradata, Vertica, Yellowbrick

Syntax

Form 1: All data sources:

server=*"server-identifier"*

Form 2: SAP HANA:

server=*"server-identifier:port"*

Required Arguments

server-identifier
specifies the server identifier for the data source.

server-identifier:port
specifies the server identifier for the data source and the associated port value.

Details

Hadoop: This option specifies the name of the node in the Hadoop cluster where the Hive service is running.

Spark: This option specifies the name of the node in the Hadoop cluster where the Spark2 Thrift Server is running.

serverDir= Data Connector Option

specifies the location that contains the data files for bulk loading or bulk unloading.

Valid in:	CASLIB statement, PROC CASUTIL: SAVE statement
Default:	none
Interactions:	If a value is not specified for clientDir=, then serverDir= is ignored. To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.
Data source:	SAP IQ
See:	bulkLoad= data connector option, bulkUnload= data connector option, clientDir= data connector option

Syntax

serverDir=*"location"*

Required Argument

location
specifies the location that contains the data files for bulk loading.

Details

This option specifies the location of the data files that contain the table rows to load to the data source (bulk loading) or from the data source (bulk unloading). This location is typically an NFS directory.

> **TIP** When specified, this option can improve bulk-load or bulk-unload performance.

The location that you specify for serverDir= is the same location as that specified for clientDir=. The serverDir= option specifies the path that the data source server uses to access the data file location. In contrast, clientDir= specifies the path that is used by the CAS nodes to access the location. The values for serverDir= and clientDir= might look different, depending on how the file systems are mounted.

sheet= Data Connector Option

specifies the name of a worksheet within an Excel file to import.

Valid in: CASLIB statement

Default: none (imports the entire worksheet)

Interactions: If you plan to import a range of cells from an XLS file, do not specify the sheet= option. Instead, specify the worksheet as part of the range= option.

Setting getSheetNames=TRUE overrides the value for any other option.

Data source: PC Files

Syntax

sheet=*"worksheet-name"*

Required Argument

worksheet-name
specifies the name of a worksheet within an Excel file to import.

show_metadata= Data Connector Option

specifies whether to display metadata tables and columns.

Valid in: CASLIB statement

Default: NO

Data source: Salesforce

Syntax

show_metadata=YES | NO

Required Argument

YES | NO
specifies whether to display metadata tables and columns.

show_recycled= Data Connector Option

specifies whether to display deleted table rows in the recycle bin.

Valid in: CASLIB statement

Alias: show_deleted=

Default: none

Data source: Salesforce

Syntax

show_recycled=YES | NO

Required Argument

YES | NO
specifies whether to display deleted table rows in the recycle bin.

sliceColumn= Data Connector Option

specifies the column to use when partitioning data for a multinode data Read.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Default: none

Restrictions: *Hadoop:* Do not specify a column of type DOUBLE for this option.
Hadoop, Spark, Teradata: This option applies to serial data transfer only.

Interactions: Multinode data Read is not on by default. Use numReadNodes= to specify a number other than 1 (or specify 0, which uses all nodes).
Do not specify both the sliceColumn= option and the sliceExpressions= option.

Data source: Amazon Redshift, DB2, DuckDB, Greenplum, Hadoop, Impala, Informix, JDBC, Microsoft SQL Server, MongoDB, MySQL, Netezza, ODBC, Oracle, PostgreSQL,

SAP HANA, SAP IQ, SAS Federation Server, SingleStore standard, Snowflake, Spark, Teradata, Vertica, Yellowbrick

Note: Support for DuckDB was added in 2026.09.

See: numReadNodes= data connector option

Syntax

sliceColumn=*"column-name"*

Required Argument

column-name
specifies the column to use when partitioning data for a multinode data Read.

Details

Ideally, the column that you select should contain integer data with a relatively small number of distinct values that are fairly evenly distributed. Columns with continuous numeric values can work as well. SAS uses the MOD function, applied to the values in the specified column, to assign data to a partition. Each partition is assigned to a node that performs the Read operation for its slice of data. For more information, see "About Loading Data When Using Multinode Data Transfer" on page 179.

If you specify a NUMERIC or DECIMAL value for sliceColumn=, then these values must be formatted so that 0 < width - decimal digits < 19. This ensures that the value can be used as a BIGINT value for the slicing process. For example, a DECIMAL column that is formatted as `21.3` can be used because 21 - 3 = 18, but a DECIMAL column that is formatted as `22.3` could not be used because 22 - 3 = 19. Similarly, a NUMERIC column that is formatted as `18.1` can be used, but a NUMERIC column with format `24.3` could not be used.

sliceExpressions= Data Connector Option

specifies expressions that partition your data for multinode loading into CAS.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Alias: dbSlice=

Default: none

Restriction: *Hadoop, Spark, Teradata:* This option applies to serial data transfer only.

Requirement: The number of expressions that you provide must match the numReadNodes= value, if that value is greater than 1. If numReadNodes=0 (use all active nodes),

then the number of expressions must match the number of nodes that are loading the data into CAS.

Interactions: Do not specify both the sliceExpressions= option and the sliceColumn= option.

Salesforce: If you set a numReadNodes= value that is greater 1, then you must also set sliceExpressions=. Otherwise, numReadNodes= resets to its default (1).

Data source: Amazon Redshift, DB2, DuckDB, Google BigQuery, Greenplum, Hadoop, Impala, Informix, JDBC, Microsoft SQL Server, MongoDB, MySQL, Netezza, ODBC, Oracle, PostgreSQL, Salesforce, SAP HANA, SAP IQ, SAS Federation Server, SingleStore standard, Snowflake, Spark, Teradata, Vertica, Yellowbrick

Note: Support for DuckDB was added in 2026.09.

Tip: You can check in SAS Environment Manager to determine the number of active nodes for the CAS server.

Example: If numReadNodes=2, then you might specify this option as shown:

```
sliceExpressions=("Idnum < 50", "Idnum >= 50")
```

Syntax

sliceExpressions=(*"expression-1"*, *"expression-2"*, ...)

Required Argument

expression-*N*
specifies an expression that partitions your data for multinode loading into CAS. Enclose each expression in double quotation marks, and separate expressions with a comma.

The expressions that you specify are passed exactly as you enter them to your data source.

socket= Data Connector Option

specifies the path and file name of the socket file that listens for connections to MySQL.

Valid in: CASLIB statement

Default: **`/tmp/mysql.sock`**

Data source: MySQL

Syntax

socket=*"file-name"*

Required Argument

file-name
specifies the path and file name of the socket file that listens for connections to MySQL. If you provide this value, the MySQL server creates the socket file in the data directory unless you specify an absolute path to a different directory.

srcType= Data Connector Option

specifies the type of data source that you are connecting to.

Valid in: CASLIB statement

Default: none

Requirement: This option is required for the CASLIB statement.

Data source: Amazon Redshift, DB2, DuckDB, Google BigQuery, Greenplum, Hadoop, Impala, Informix, JDBC, Microsoft SQL Server, MongoDB, MySQL, Netezza, ODBC, Oracle, PostgreSQL, Salesforce, SAP HANA, SAP IQ, SAS Federation Server, SAS SpeedyStore, SingleStore standard, Snowflake, Spark, SPD Engine files, Teradata, Vertica, Yellowbrick

Note: Support for DuckDB was added in 2026.09.

Syntax

srcType=*"data-source-type"*

Required Argument

data-source-type
specifies the type of data source that you are connecting to. The value that you provide is specific to your data source. See the listed values in Details. Enclose the value in single or double quotation marks.

Details

Here is the list of valid values for each data source type.

Table 8.3 *Valid Data Source Values for Each Data Connector*

Data Source	srcType= Value
Amazon Redshift	redshift

Data Source	srcType= Value
DB2	db2
DuckDB	duckdb
Google BigQuery	bigquery
Greenplum	greenplum
Hadoop	hadoop
Impala	impala
Informix	informix
JDBC	jdbc
Microsoft SQL Server	sqlserver
MongoDB	mongodb
MySQL	mysql
Netezza	netezza
ODBC	odbc
Oracle	oracle
PostgreSQL	postgres
Salesforce	sforce
SAP HANA	hana
SAP IQ	sapiq
SAS Federation Server	fedsvr
SingleStore	singlestore
SingleStore standard	singlestore_standard
Snowflake	snowflake
Spark	spark

Data Source	srcType= Value
SPD Engine files	spde
Teradata	teradata
Vertica	vertica
Yellowbrick	yellowbrick

ssl_ca= Data Connector Option

specifies the full path to the certificate authority file to use to access an external data source.

Valid in: CASLIB statement

Alias: sslca= [Teradata]

Defaults: none [MongoDB, MySQL, SingleStore standard, Teradata]
Inherits from the configuration for the SAS Viya platform [SAS SpeedyStore]

Interactions: *MongoDB:* If you use the ssl_ca= option to connect to your MongoDB database, then you must also specify the schema_ssl_ca= option to connect to a schema file.
Teradata: If you specify the ssl_ca= option, do not also specify the ssl_ca_path= option.

Data source: MongoDB, MySQL, SAS SpeedyStore, SingleStore standard, , Teradata

Note: Support for Teradata was added in 2026.03.

Example: ssl_ca="/u/myuser/mysql/certs/ca-cert.pem"

Syntax

ssl_ca=*"path-and-file-name"*

Required Argument

path-and-file-name
specifies the full path to the certificate authority file to use to access an external data source. This file is in PEM format.

ssl_ca_path= Data Connector Option

specifies the path to a directory that contains the certificates to use for a secure connection to your data source.

Valid in: CASLIB statement

Alias: sslcapath=

Default: none

Interaction: If you specify this option, do not also specify the ssl_ca= data connector option.

Data source: Teradata

Note: Support for this option was added in 2026.03.

See: ssl_ca= data connector option

Syntax

ssl_ca_path=*"value"*

Required Argument

value
specifies the directory that contains certificate files that are used for secure connections to your data source.

ssl_cert= Data Connector Option

specifies the full path to the client public key certificate file.

Valid in: CASLIB statement

Default: none

Interaction: *MongoDB:* If you use the ssl_cert= option to connect to your MongoDB database, then you must also specify the schema_ssl_cert= option to connect to a schema file.

Data source: MongoDB, MySQL, SingleStore standard

Example: ssl_cert="/u/myuser/mysql/certs/client-cert.pem"

Syntax

ssl_cert=*"path-and-file-name"*

Required Argument

path-and-file-name
specifies the full path to the client public key certificate file. This file is in PEM format.

ssl_cipher= Data Connector Option

specifies a list of ciphers that are permitted to be used for encryption.

Valid in: CASLIB statement

Default: none

Data source: MySQL, SingleStore standard

Example: ssl_cipher="DHE-RSA-AES256-SHA"

Syntax

ssl_cipher=*"cipher-list"*

Required Argument

cipher-list
specifies a list of ciphers that are permitted to be used for encryption.

Details

For more information about using ciphers, see the security information in your data source documentation.

ssl_key= Data Connector Option

specifies the path to the client private key file.

Valid in: CASLIB statement

Default: none

Data source: MySQL, SingleStore standard

Example: ssl_key="/u/myuser/mysql/newcerts/client-key.pem"

Syntax

ssl_key=*"key-file-path"*

Required Argument

key-file-path
specifies the path to the client private key file. This file is in PEM format.

sslcrc= Data Connector Option

specifies the certificate revocation status values for which the CLI allows a TLS connection.

Valid in: CASLIB statement

Alias: ssl_crc=

Default: ALLOW

Interaction: This option is used only when sslMode="VERIFY-FULL". For any other sslMode= value, the certificate status is not checked.

Data source: Teradata

Note: Support for this option was added in 2026.03.

See: sslMode= data connector option

Syntax

sslcrc=ALLOW | PREFER

Required Argument

ALLOW | PREFER
specifies the certificate revocation status values for which the CLI allows a TLS connection.

sslKeyStore= Data Connector Option

Specifies the path to the keystore file.

Valid in: CASLIB statement

Default: none

Data source: Hadoop

Note: Support for this option was added in 2025.07.

Syntax

sslKeyStore=*"file-path"*

Required Argument

file-path
specifies the path to the keystore file. For more information, see "Using Kerberos" and "Connector Configuration Options" on the insightsoftware website.

sslKeyStorePwd= Data Connector Option

Specifies the password for accessing the keystore file.

Valid in: CASLIB statement

Default: none

Interaction: You create the keystore file with the sslKeyStore= option.

Data source: Hadoop

Note: Support for this option was added in 2025.07.

Example: `sslKeyStorePwd='GQf5banoZL87qUK2lIO0WUiGr61mDspbWT7AwQvYWUN'`

Syntax

sslKeyStorePwd=*"password-value"*

Required Argument

password-value
specifies the password for accessing the keystore file. For more information, see "Using Kerberos" and "Connector Configuration Options" on the insightsoftware website.

sslMode= Data Connector Option

specifies the support mode for a TLS encrypted connection to your data source.

Valid in:	CASLIB statement
Category:	Data Access
Alias:	ssl_mode= [Teradata]
Defaults:	prefer [PostgreSQL, Yellowbrick] none [Teradata]
Restriction:	PostgreSQL, Yellowbrick: Specify the value for this option in lowercase.
Requirement:	This option is required for CASLIB connections that use TLS encryption.
Data source:	PostgreSQL, Teradata, Yellowbrick
Note:	Support for Teradata was added in 2026.03.
See:	For more information about the behavior of each value, see Details in "SSLMODE= LIBNAME Statement Option" in *SAS/ACCESS for Relational Databases: Reference*.

Syntax

sslMode='allow | disable | prefer | require | verify-ca | verify-full'

Required Arguments

allow
specifies that encryption is not required, but encryption is accepted if the DBMS server requires it.

disable
specifies not to use encryption when connecting to the database.

prefer
specifies that encryption is not required, but encryption should be used if it is available.

require
specifies that encryption is required when connecting to the database.

verify-ca
specifies that encryption is required, and an authentication certificate might be required. Whether an authentication certificate is required depends on your database policy.

verify-full
specifies that encryption and an authentication certificate are required.

sslProtocol= Data Connector Option

specifies the TLS protocol version for a secure connection.

Valid in: CASLIB statement

Alias: ssl_protocol=

Default: none

Data source: Teradata

Note: Support for this option was added in 2026.03.

Syntax

sslProtocol=*"value"*

Required Argument

value
specifies the TLS protocol version for a secure connection. Possible values are "TLS1_2" or "TLS1_3". Other values are ignored.

sslTrustStore= Data Connector Option

specifies the path to a Transport Layer Security trust store.

Valid in: CASLIB statement

Default: none

Data source: Hadoop, Spark

See: allowSelfSignedCerts= data connector option, trustStorePassword= data connector option

Syntax

sslTrustStore=*"path"*

Required Argument

path
specifies the path to the certificate to be accepted from the server when connecting using Transport Layer Security.

statusInterval= Data Connector Option

specifies whether to print a message to the client when a node adds *n* buffers to the table by a node, where *n* is the value of this option.

Valid in: CASLIB statement, PROC CASUTIL: LOAD and SAVE statements

Default: 0 (no message)

Restriction: This option applies to parallel data transfer only.

Data source: Hadoop, Spark, SPD Engine files, Teradata

Syntax

statusInterval=*number*

Required Argument

number
specifies whether to print a message to the client when a node adds *n* buffers to the table by a node, where *n* is the value of this option.

Details

If you specify a value for this option outside of a CASLIB statement, it overrides the value in a CASLIB statement.

table_qualifier= Data Connector Option

specifies a table qualifier when you require a value other than the default project qualifier.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, LOAD statements

Alias: qualifier=

Default: none

Data source: Google BigQuery

Syntax

table_qualifier=*"qualifier"*

Required Argument

qualifier
specifies a table qualifier when you require a value other than the default project qualifier.

tableType= Data Connector Option

specifies the format type of the SAP HANA table to return.

Valid in: CASLIB statement
Default: ROW
Data source: SAP HANA

Syntax

tableType="ROW" | "COLUMN"

Required Argument

"ROW" | "COLUMN"
specifies the format type of the SAP HANA table to return.

tempDirectory= Data Connector Option

specifies the directory for temporary files.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements
Alias: temp_directory=
Default: location that is specified by the UTILLOC system option
Data source: DuckDB
Note: Support for this option was added in 2026.09.

Syntax

tempDirectory=<">*path*<">

Required Argument

path
specifies the directory for temporary files. If the path includes spaces, then enclose the path value in quotation marks.

tezQueueName= Data Connector Option

specifies an Apache Tez queue name in which to execute client requests.

Valid in: CASLIB statement

Default: value configured in hive-site.xml file

Data source: Hadoop, Spark

See: zooKeeperNamespace= data connector option, zooKeeperPort= data connection option, zooKeeperQuorum= data connector option

Syntax

tezQueueName=*"name"*

Required Argument

name
specifies the name of the Apache Tez queue.

threadedLoad= Data Connector Option

specifies whether to use multiple threads to load data into CAS.

Valid in: CASLIB statement, PROC CASUTIL: LOAD statement

Alias: threadedRead=

Default: FALSE

Interaction: This option is used only when readMode=STORAGE.

Data source: Google BigQuery

Syntax

threadedLoad=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether to use multiple threads to load data into CAS.

Details

As a best practice, enable this option when you are loading data from very large tables into CAS. Because of the overhead of partitioning your data, you might not notice performance improvements for small tables. Performance improvements also depend on the resources that are available for your process, such as CPU availability, the size of the data to be streamed, and other factors. It is recommended that you test data on your system to determine when to enable this option.

If threaded reads cannot be executed due to system configurations, a message is printed to the log to notify you.

timeout= Data Connector Option

specifies the maximum completion time in seconds for a storage task.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: 60

Data source: Microsoft SQL Server, Snowflake

Syntax

timeout=*seconds*

Required Argument

seconds
specifies the maximum completion time in seconds for a storage task.

Details

If the task does not complete within the specified time, the storage system might return an error. Use this option to establish a connection to the external storage system.

This option is applicable only when bulk loading data to Microsoft Azure or to Microsoft Fabric Data Warehouse. For more information, see "Bulk Loading to Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse" on page 241.

token= Data Connector Option

specifies a temporary token that is associated with the temporary credentials that you specify with the key= and secret= options.

Valid in: CASLIB statement [all], PROC CASUTIL: LOAD statement [Amazon Redshift], SAVE statement [all]

Category: Authentication

Default: none

Requirements: *Amazon Redshift:* To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE.

Snowflake: To specify this option, you must also set bulkLoad=TRUE.

Data source: Amazon Redshift, Snowflake

Syntax

token=*"temporary-token"*

Required Argument

temporary-token
specifies a temporary token that is associated with the temporary credentials that you specify with the key= and secret= options.

trustStorePassword= Data Connector Option

specifies the password for the Transport Layer Security truststore.

Valid in: CASLIB statement

Default: none

Data source: Hadoop, Spark

See: allowSelfSignedCerts= data connector option, sslTrustStore= data connector option

Syntax

trustStorePassword=*"password"*

Required Argument

password
specifies the password for the Transport Layer Security truststore.

udfDatabase= Data Connector Option

specifies the location of user-defined aggregate functions for statistics pushdown.

Valid in: CASLIB statement

Default: If you do not specify udfDatabase= then the database specified in the database= option is used.

Interaction: To process an aggregation request in the database, you must either specify the aggregatePushdown=true data connector option or specify sql=true in an action that supports aggregate pushdown. The action parameter takes precedence over the data connector option.

Data source: SAS SpeedyStore

Note: This option was added in 2024.11.

See: “aggregatePushdown= Data Connector Option” on page 389, “Simple Analytics Action Set” in *SAS Visual Analytics: Programming Guide*

Syntax

udfDatabase=*"database-name"*

Required Argument

database-name
specifies the database where the user-defined functions (UDFs) are stored.

Details

Use this option when the following two conditions are both met:

- The data connector option aggregatePushdown= is set to true or the simple.summary action parameter sql= is set to true. The action parameter takes precedence over the data connector option.
- The action requests kurtosis or skewness statistics. In the simple.summary action, you can specify KURTOSIS or SKEWNESS in the subSet= parameter. If you do not specify statistics in the subSet= parameter, kurtosis and skewness are among the default statistics that are returned.

When both of the above conditions are met, you must download the open-aggregates collection of user-defined functions (UDFs) and store them in a database. The UDFs, and the instructions for storing them, are available in GitHub at https://github.com/sassoftware/open-aggregates.

Specify the database name in the udfDatabase= option. Alternatively, you can store the UDFs in the default database that you specify in the database= option. In that case, the udfDatabase= option is not required.

When an aggregation request is pushed down to the database, the following message is written in the SAS log:

```
NOTE: The aggregation is being calculated in the database.
```

Error processing for the UDFs depends on whether you enable aggregate pushdown with the sql= action parameter or with the aggregatePushdown= data connector option:

- If sql=true in the summary action, then aggregate pushdown is attempted. If it is not possible, the request fails and messages are written to the SAS log. The following messages indicate that the database that contains the UDFs is not found:

  ```
  NOTE: The aggregation could not be calculated in the database. A SingleStore
  user-defined function required for the aggregation does not exist in the
  database. See SAS documentation about aggregate pushdown with SingleStore.
  ERROR: The SQL generation request is not supported.
  ERROR: Unsupported function
  ```

- If sql= is not specified in the summary action, but the data connector option aggregatePushdown=true, then aggregate pushdown is attempted. If the database that contains the UDFs is not found, the action continues without aggregate pushdown, and a message is written to the SAS log:

  ```
  NOTE: The aggregation could not be calculated in the database. A SingleStore
  user-defined function required for the aggregation does not exist in
  the database. See SAS documentation about aggregate pushdown with SingleStore.
  ```

uri= Data Connector Option

specifies a free-form JDBC URL to use as the Hive or Spark JDBC connection URI.

Valid in: CASLIB statement

Default: none

Interaction: If you use this option, options that alter the generated JDBC URL (such as properties=) are ignored.

Data source: Hadoop, Spark

Syntax

uri=*'JDBC-URL'*

Required Argument

JDBC-URL
specifies a free-form JDBC URL to use as the Hive or Spark JDBC connection URI.

Details

You can use this option to connect with a JDBC driver that is redistributed with the SAS Viya platform or with a third-party JDBC driver that you obtain and install.

In the JDBC URL for all drivers (Simba and other third-party), specify UseNativeQuery=1 (or the equivalent driver option) in the JDBC URL. Specify this option because SAS generates SQL that is native to the data source. You might also want to specify an option that limits the length of STRING columns that are read into the SAS Viya platform.

For more information about the URL required by the redistributed Simba drivers, see your Simba documentation.

url= Data Connector Option

specifies the JDBC connection string.

Valid in: CASLIB statement

Default: none

Requirement: This option is required in the CASLIB statement.

Data source: JDBC

Syntax

url="jdbc:*driver-name*:*//driver-connection-options*"

Required Argument

jdbc:*driver-name*:*//driver-connection-options*
specifies the JDBC connection string.

use_information_schema= Data Connector Option

specifies whether to use INFORMATION_SCHEMA in Google BigQuery to retrieve a list of schemas and tables or views.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Default: TRUE

Data source: Google BigQuery

Syntax

use_information_schema=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether to use INFORMATION_SCHEMA in Google BigQuery to retrieve a list of schemas and tables or views.

Details

By default, SAS/ACCESS uses INFORMATION_SCHEMA to get metadata for the list of schemas and tables or views if the names contain a pattern. Names with patterns can be searched using the wildcard characters '%' or '_'.

You might want to disable INFORMATION_SCHEMA to improve performance, especially if you are not using patterns to query schema or table names.

Note: SAS directory services, including calls to PROC DATASETS, do not use patterns to retrieve schema or table names.

use_native_names= Data Connector Option

specifies whether to restore native naming conventions for custom tables and columns.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Alias: native_names=

Default: NO

Restriction: Do not use the __c suffix in queries.

Requirement: When use_native_names=YES, the __s suffix is required on component fields to update them.

Data source: Salesforce

Syntax

use_native_names=YES | NO

Required Argument

YES | NO
specifies whether to restore native naming conventions for custom tables and columns.

useLog= Data Connector Option

specifies whether to receive a report of rejected rows and corresponding error messages.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: FALSE

Requirement: Ensure that the account that you use and the COPY command have permission to write to the storage account.

Interaction: The log and error files are downloaded to the location that is specified by the defaultDir= data connector option.

Data source: Microsoft SQL Server

Syntax

useLog=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether to receive a report of rejected rows and corresponding error messages.

Details

When USELOG=TRUE, an error file location is added to the COPY command that performs bulk loading. A log of rejected records is also saved to this location.

This option is applicable only when bulk loading data to Microsoft Azure or to Microsoft Fabric Data Warehouse. For more information, see "Bulk Loading to Microsoft Azure Synapse Analytics or Microsoft Fabric Data Warehouse" on page 241.

useNarrowCharacterTypes= Data Connector Option

specifies whether to use narrow character types, such as CHAR or VARCHAR, for all character variables when writing data to the data source.

Valid in:	CASLIB statement, PROC CASUTIL: SAVE statement
Aliases:	useNCT= useNarrowCharTypes=
Defaults:	FALSE [all except SAP IQ] TRUE [SAP IQ]
Restriction:	*Hadoop:* This option applies to serial data transfer only.
Data source:	Amazon Redshift, DB2, DuckDB, Greenplum, Hadoop, Impala, Informix, JDBC, Microsoft SQL Server, MongoDB, MySQL, Netezza, ODBC, Oracle, PostgreSQL, Salesforce, SAP IQ, SAS Federation Server, SingleStore standard, Snowflake, Vertica, Yellowbrick
Note:	Support for DuckDB was added in 2026.09.

Syntax

useNarrowCharacterTypes=TRUE | FALSE

Required Argument

TRUE | FALSE
when set to TRUE, specifies to use narrow character types, such as CHAR or VARCHAR, for all character variables when writing data to the data source. When set to FALSE, this option specifies to use wide character types, such as NCHAR or NVARCHAR, for all character variables when writing data to the data source.

Details

Because the CAS server uses UTF-8 encoding by default, this option defaults to FALSE.

usePipe= Data Connector Option

specifies whether to use a pipe file to facilitate bulk loading.

Valid in: CASLIB statement, PROC CASUTIL: SAVE statement

Default: FALSE

Requirement: To specify this option, you must also set bulkLoad=true.

Data source: SAP IQ

See: bulkLoad= data connector option

Syntax

usePipe=TRUE | FALSE

Required Arguments

TRUE
specifies that SAS should use a pipe file to facilitate bulk loading to your data source.

FALSE
specifies that SAS should not use a pipe file for bulk loading.

Details

Enabling the usePipe= data connector option typically improves performance when bulk loading data to your data source. If enabled, data is loaded to your database

via a named pipe file, and there is no bulk-load data file. The named pipe file is deleted when bulk loading completes.

userName= Data Connector Option

specifies the data source user name.

Valid in: CASLIB statement, PROC CASUTIL: CONTENTS, DELETESOURCE, LIST, LOAD, and SAVE statements

Category: Authentication

Aliases: uid=
user=

Default: none

Restrictions: *Hadoop, Spark:* When a Kerberos connection is used, no values should be specified for userName= and password=.

SPD Engine files: This option is valid only when SPD Engine files are stored in Hadoop (hdfs=TRUE).

Requirements: If it is required by your data source, you must specify valid credentials to access data. You can provide these credentials by specifying either userName= and password= values or by specifying an authenticationDomain= value.

Oracle: Specify the userName= value in uppercase. Oracle is case-sensitive and stores user name values in uppercase.

Data source: Amazon Redshift, DB2, Greenplum, Hadoop, Impala, Informix, JDBC, Microsoft SQL Server, MongoDB, MySQL, Netezza, ODBC, Oracle, PostgreSQL, Salesforce, SAP HANA, SAP IQ, SAS Federation Server, SAS SpeedyStore, SingleStore standard, Snowflake, Spark, SPD Engine files, Teradata, Vertica, Yellowbrick

Note: The userName= and password= options remain valid. However, these options are less secure because you are saving authentication information in your programs. Therefore, as a better security practice, it is recommended that you use the authenticationDomain= option instead.

Syntax

userName=*"user-name"*

Required Argument

user-name
specifies the data source user name.

Details

Typically, you specify userName= and password= values (or an authenticationDomain= value) when you add a caslib. These credentials are then used for any statement that accesses the data using that caslib. If you specify userName= and password= values in a statement other than the CASLIB statement, these credentials override any credentials that were specified when the caslib was added.

Hadoop: If you do not specify this option, the current user context is used. One scenario for leaving this option unspecified is to use the current Kerberos context.

MongoDB: The authenticationDomain= option, which is mentioned as an alternative to the userName= and password= options, is not supported by MongoDB.

PostgreSQL: If you are connecting to PostgreSQL on Microsoft Azure, the userName= value is the user ID that you use to authenticate to Azure. For more information, see "Authenticate to Microsoft Azure Using Single Sign-On" in *SAS/ACCESS for Relational Databases: Reference*.

useSSL= Data Connector Option

specifies whether to use Transport Layer Security (TLS) to provide encryption for connections.

Valid in:	CASLIB statement, PROC CASUTIL: SAVE statement
Defaults:	TRUE [Amazon Redshift, Snowflake] FALSE [Hadoop, Spark]
Requirements:	[Amazon Redshift] To specify this option, you must also set bulkLoad=TRUE or bulkUnload=TRUE. [Snowflake] To specify this option, you must also set bulkLoad=TRUE.
Data source:	Amazon Redshift, Hadoop, Snowflake, Spark

Syntax

useSSL=TRUE | FALSE

Required Argument

TRUE | FALSE
specifies whether to perform TLS encryption, when TLS is configured for the target server.

Note: All discussion of TLS is also applicable to the predecessor protocol, Secure Sockets Layer (SSL).

varcharConversion= Data Connector Option

specifies the column length at which to begin converting CHAR to VARCHAR.

Default: none

Range: 1–32767

Data source: SAS data sets

Syntax

varcharConversion=*column-length*

Required Argument

column-length
specifies the column length at which to begin converting CHAR to VARCHAR.

warehouse= Data Connector Option

specifies the Snowflake warehouse to use for connections to your data source.

Valid in: CASLIB statement

Default: none

Data source: Snowflake

Syntax

warehouse=*"name"*

Required Argument

name
specifies the default Snowflake warehouse to use.

webHdfsUrl= Data Connector Option

specifies the URL to use to access the Hadoop distributed file system (HDFS) through a REST API.

Valid in: CASLIB statement

Default: none

Requirement: *SPD Engine files:* To use this option, you must also specify hdfs=TRUE.

Data source: Hadoop, Spark, SPD Engine files

Syntax

webHdfsUrl=*"web-HDFS-URL"*

Required Argument

web-HDFS-URL
specifies the URL to use to access the Hadoop distributed file system (HDFS) through a REST API.

Details

Starting in 2025.08, when using WebHDFS, you can specify the REST URL in the WEBHDFSURL= option. When you use the WEBHDFSURL= option, you are not required to run the Hadoop tracer script to copy the Hadoop JAR and configuration files to the SAS client machine. Prior to 2025.08, those steps were required. In addition, you are no longer required to specify environment variables SAS_HADOOP_CONFIG_PATH or SAS_HADOOP_RESTFUL=1. If you have Apache Knox Gateway security enabled, specify the Knox Gateway endpoint in the WEBHDFSURL= option. You are no longer required to specify environment variable KNOX_GATEWAY_URL. For more information, see "Configuring SAS/ACCESS Interface to Hadoop" in *SAS/ACCESS for Relational Databases: Reference*.

write= Data Connector Option

specifies the password to access and save a password-protected data set.

Default: none

Requirement: This option is required only if the data set is password-protected.

Data source: SAS data sets

Syntax

write=*"password"*

Required Argument

password
specifies the password to access and save a password-protected data set.

writePW= Data Connector Option

specifies the password to access and save a password-protected SPD Engine table.

Valid in: CASLIB statement

Default: none

Restriction: This option applies to serial data transfer only.

Data source: SPD Engine files

Syntax

writePW=*"password"*

Required Argument

password
specifies the password to access and save a password-protected SPD Engine table.

zooKeeperNamespace= Data Connector Option

specifies the ZooKeeper namespace.

Valid in: CASLIB statement

Default: value configured in hive-site.xml file

Data source: Hadoop, Spark

See: tezQueueName= data connector option, zooKeeperPort= data connection option, zooKeeperQuorum= data connector option

Syntax

zooKeeperNamespace=*"namespace"*

Required Argument

namespace
specifies the namespace configured on ZooKeeper for the HiveServer2 znodes.

zooKeeperPort= Data Connector Option

specifies the ZooKeeper port number.

Valid in: CASLIB statement, PROC CASUTIL statement

Default: none

Data source: Hadoop, Spark

See: tezQueueName= data connector option, zooKeeperNamespace= data connector option, zooKeeperQuorum= data connector option

Syntax

zooKeeperPort=*"port-number"*

Required Argument

port-number
specifies the port number that ZooKeeper listens to for client requests.

zooKeeperQuorum= Data Connector Option

specifies the server nodes that are available to run client requests.

Valid in: CASLIB statement

Default: value configured in hive-site.xml file

Data source: Hadoop, Spark

See: tezQueueName= data connector option, zooKeeperNamespace= data connector option, zooKeeperPort= data connection option

Syntax

zooKeeperQuorum=*"nodes"*

Required Argument

znodes

specifies the server nodes that are available to run client requests, separated by commas.

Note znode values must be submitted within double quotation marks.

9

Data Types

Overview of Data Types for CAS Table Columns 609
Summary of Data Types 610
Character Data 611
Numeric Data 613

Overview of Data Types for CAS Table Columns

A *data type* is an attribute that specifies the type of data that a table column stores. For example, the data type indicates whether the values in a particular column are characters, integers, or floating-point numbers. The data type also determines how much memory can be allocated for the column value.

The following sections describe the types of data that can be stored in CAS table columns. All CAS table data types support missing values.

- "Summary of Data Types" on page 610
- "Numeric Data" on page 613
- "Character Data" on page 611

For more information about data types for tables external to CAS, see the following documents:

- Data Types in SAS
- Data Types in Tables Accessed by Data Connectors on page 167
- Data Types for the CAS Engine on page 41
- Data Types for the CAS Language (CASL)
- Data Types for Parquet Tables on page 159

Summary of Data Types

Table 9.1 *Data Types for SAS Cloud Analytic Services Table Columns*

Data Type	Description	Defining Missing Values

CHAR(n)	stores a fixed-length character string. *n* is the number of bytes used to store all values. If the value stored contains fewer than *n* bytes, then SAS adds blank spaces to the end of the value to fill the column.
	Storage size (range): 1 – 32767 bytes
		Specify zero or more blank spaces surrounded by single or double quotation marks:
		`fname="";`
		`Lname="    ";`
		Note: fname contains zero spaces, Lname contains four spaces.
		`L1 = lengthc(fname);`
		`L2 = lengthc(Lname);`
		Note: The value of `L1` is one and the value of `L2` is four when theLENGTHC function is specified.

VARCHAR(n)	stores a varying-length character string. *n* is the maximum number of characters to store.
	Storage size (range): 0–536,870,911 Unicode characters
		Specify zero or more blank spaces surrounded by single or double quotation marks:
		`fname="";`
		`Lname="    ";`
		Note: `fname` contains zero spaces, `Lname` contains four spaces.
		`L1 = lengthc(fname);`
		`L2 = lengthc(Lname);`
		Note: The value of `L1` is zero and the value of `L2` is four when the LENGTHC function is specified.

DOUBLE	stores a signed, 64-bit double-precision, floating-point number. The DOUBLE data type stores numbers of large magnitude and permits computations that	Specify a single period: `num1=.;` For special numeric missing variables, specify a period followed by an upper or

Data Type	Description	Defining Missing Values

	require many digits of precision to the right of the decimal point.
	For information about precision when using DOUBLE numeric values, see "Numeric Precision" in *SAS Programmer's Guide: Essentials*.
	Storage size: 8 (CAS engine)
		lowercase letter from A to Z, or use an underscore:
		`num1=.a;`
		`num2=.S;`
		`num3=_;`
		Note: SAS Cloud Analytic Services does not support DOUBLE columns that use less than 8 bytes.
		For more information, see missing numeric values.

INT32	Stores a 32-bit signed integer value. When SAS encounters a value that corresponds to the most negative possible INT32 value, Value range: Value range: -2,147,483,648 to 2,147,483,647	-2,147,483,648 or '80000000'x (the most negative possible INT32 value is treated as missing)
INT64 (BIGINT)	Stores a 64-bit signed integer value. Data type BIGINT is the same as INT64. The range of values that can be represented as an INT64 is 1024 times as large as the range that can be represented as a DOUBLE value. Value range: -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	-9,223,372,036,854,775,808 or '8000000000000000'x (the most negative possible INT64 value is treated as missing)

Character Data

CHAR(*n*)
: stores a fixed-length character string. *n* is the number of bytes used to store all values. If the value stored contains fewer than *n* bytes, then SAS adds blanks (spaces) to the end of the value to fill the column.

  Missing values are supported and are represented using single or double quotation marks that contain no characters or any number of blank spaces.

Note: The CHAR data type does not support zero-length character strings. For example, the assignment statements for the CHAR strings below are defined using empty quotation marks, and they both have a length of 1 even though they contain zero characters:

```
c_var2='';

c_var1="";
```

VARCHAR(*n*)
stores a varying-length character string. *n* is the maximum number of characters to store, up to a maximum of 536,870,911 UTF-8 encoded characters. If VARCHAR(10) is specified and a character string is only five characters long, then only five characters are stored. The value is not padded with spaces.

Missing values are supported and are represented using single or double quotation marks that contain no characters or any number of blank spaces.

Note: The VARCHAR data type supports zero-length character strings if they are defined in an assignment statement with empty strings. For example, the assignment statements in the example below define the two VARCHAR strings as zero length using empty quotation marks that contain no characters, including blank spaces. The LENGTHC function returns a length of zero for both strings.

```
data _null_;
   length vc1 varchar(32) vc2 varchar(10);
   vc1 = '';
   vc2 = "";
   L1 = lengthc(vc1);
   L2 = lengthc(vc2);
   put L1=; put L2=;
run;
```

Log output:

```
L1=0
L2=0
```

Here are some facts to keep in mind when deciding whether to use the VARCHAR or CHAR data type.

- The VARCHAR data type is ideal for storing character strings that vary widely in length.
- Each VARCHAR value has a 16-byte header, so storing values that are smaller than 16 bytes takes more memory than using the CHAR data type.
- VARCHAR(*) indicates that no maximum length is set for the variable that you are defining. Values containing up to 536,870,911 characters can be stored in a variable defined in this way. Using VARCHAR(*) can be helpful if the maximum length of the data for a variable is not known when you are defining the variable.
- If you copy a table that is defined with a VARCHAR(*) to an engine library that does not support VARCHAR, a CHAR data type is created instead. The

CHAR is defined with a maximum length of 32,767 bytes, or 8,191 UTF8 characters, and truncation can occur. If a VARCHAR column has a defined maximum length, such as VARCHAR(10), the CHAR column is created in the new table with a byte length of 40. A maximum length of 40 bytes is required to hold 10 characters in a UTF8 session.

Numeric Data

DOUBLE
Stores a signed, approximate, double-precision, floating-point number. Allows numbers of large magnitude and permits computations that require many digits of precision to the right of the decimal point. For SAS Cloud Analytic Services, this is a 64-bit double precision, floating-point number.

Missing numeric values are represented by a single period:

```
num1=.;
```

Special numeric missing variables are represented by an uppercase or lowercase letter, A through Z, or by the underscore (_) character:

```
x = .M;
```

```
x = _;
```

Note: The SAS NUMERIC data type is the same as the DOUBLE data type in CAS.

INT32
Stores a 32-bit signed integer value. Calculations for integers are more efficient than those for floating-point values, and there is no loss of precision.

When SAS encounters a value that corresponds to the most negative possible INT32 value, -2147483648 or 0x80000000, SAS considers it a missing value.

Note: The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved.

INT64 (BIGINT)
Stores a 64-bit signed integer value. Data type BIGINT is the same as INT64. Calculations for integers are more efficient than those for floating-point values, and there is no loss of precision. The range of values that can be represented as an INT64 is 1024 times as large as the range that can be represented as a DOUBLE value.

When SAS encounters a value that corresponds to the most negative possible INT64 value, -9223372036854775808 or 0x8000000000000000, SAS considers it a missing value.

Note: The CAS server supports loading, storing, and writing integer data types (INT32 and INT64). Some computations that can be completed by using the CAS server maintain the original data type. Consult the documentation to determine whether a CAS action supports an integer type. Any tasks that are not completed by the CAS server convert these data types into a SAS DOUBLE before the data is printed or saved.

10

CAS LIBNAME Options

Dictionary **615**
CASLIB= LIBNAME Statement Option 615
COMPRESS= LIBNAME Statement Option 616
DATALIMIT= LIBNAME Statement Option 617
HOST= LIBNAME Statement Option 618
NCHARMULTIPLIER= LIBNAME Statement Option 619
PORT= LIBNAME Statement Option 620
MAXTABLEMEM= LIBNAME Statement Option 621
READTRANSFERSIZE= LIBNAME Statement Option 622
SESSREF= LIBNAME Statement Option 624
TAG= LIBNAME Statement Option 624
TIMEOUTSECONDS= LIBNAME Statement Option 625
TRANSCODE_FAIL= LIBNAME Statement Option 626
UUID= LIBNAME Statement Option 627
UUIDMAC= LIBNAME Statement Option 628
WRITETRANSFERSIZE= LIBNAME Statement Option 629

Dictionary

CASLIB= LIBNAME Statement Option

Specifies the name of the caslib to use for engine operations in the LIBNAME statement.

Valid in: CAS LIBNAME Statement

Categories: Data Access
CAS

Default: the active caslib

Note: The CASLIB= LIBNAME option accepts a caslib reference name as a string, quoted or not quoted. Caslib reference names do not support named literals.

Syntax

CASLIB=*caslib-reference-name*

Required Argument

caslib-reference-name
specifies the name of the caslib. Caslib names must be unique within the session.

Details

The CASLIB= LIBNAME option binds the specified caslib to the libref. If you do not specify this option, then the libref uses the active caslib for your CAS session. For information about the active caslib, see "The Active Caslib for a CAS Session" on page 750.

Comparisons

The CASLIB= data set option overrides the CASLIB= LIBNAME statement option.

The CASLIB= LIBNAME statement option overrides the CASLIB= session option.

See Also

- "CASLIB Statement"
- "CASLIB= Data Set Option" in *SAS Data Set Options: Reference*
- "CASLIB= Session Option"
- "CASLIB= System Option" in *SAS System Options: Reference*

COMPRESS= LIBNAME Statement Option

Requests that the CAS table is compressed.

Valid in: CAS LIBNAME Statement

Categories: Data Access

CAS

Default: NO

Restriction: This option applies to output, in-memory tables only.

Syntax

COMPRESS=YES | NO

Required Arguments

YES
specifies to compress output, in-memory tables.

NO
specifies that compression is not used when creating output, in-memory tables.

Comparisons

The COMPRESS= data set option overrides the COMPRESS= LIBNAME statement option.

See Also

"COMPRESS= Data Set Option" in *SAS Data Set Options: Reference*

DATALIMIT= LIBNAME Statement Option

Specifies the maximum number of bytes of data that can be transferred between the CAS server and SAS during a single-table read.

Valid in: CAS LIBNAME Statement

Categories: Data Access
CAS

Default: 100M

Restriction: This option affects only Read access.

Tip: This option can prevent you from accidentally transferring a large amount of data from the server to the client.

Syntax

DATALIMIT=*integer* | *integer*K | *integer*M | *integer*G | ALL

Optional Arguments

integer
specifies the maximum number of bytes to read.

***integer*K**
specifies the maximum number of kilobytes to read.

***integer*M**
specifies the maximum number of megabytes to read.

***integer*G**
specifies the maximum number of gigabytes to read.

ALL
specifies that the entire file can be read, no matter how large it is.

Comparisons

The DATALIMIT= data set option overrides the DATALIMIT= LIBNAME statement option.

The DATALIMIT= LIBNAME option overrides the CASDATALIMIT= system option.

See Also

- "CASDATALIMIT= System Option" in *SAS System Options: Reference*
- "DATALIMIT= Data Set Option" in *SAS Data Set Options: Reference*
- "READTRANSFERSIZE= LIBNAME Statement Option"
- "READTRANSFERSIZE= Data Set Option" in *SAS Data Set Options: Reference*
- "WRITETRANSFERSIZE= LIBNAME Statement Option"
- "WRITETRANSFERSIZE= Data Set Option" in *SAS Data Set Options: Reference*

HOST= LIBNAME Statement Option

specifies the CAS server that is associated with a CAS session.

Valid in: CAS LIBNAME Statement

Categories: Data Access

CAS

Syntax

Form 1: **HOST=** "*Kubernetes-service-name*"

Required Argument

Kubernetes-service-name
specifies the name of the primary controller.

Default None

Range 1–256 characters

Details

By default, if you started a CAS session, the host name is determined from the session. If you have not started a session, the host name is determined as follows:

1. the value of the CASHOST= system option
2. the _CASHOST_ macro variable
3. the server uses localhost

See Also

For more information about backup controllers, see Distributed Server: Controller Fault Tolerance.

NCHARMULTIPLIER= LIBNAME Statement Option

Specifies a multiplication factor that is used to increase the number of bytes for a fixed character variable when data is transcoded to the UTF-8 encoding in order to run in the CAS server.

Valid in: CAS LIBNAME Statement

Categories: Data Access
CAS

Defaults: 1 for SBCS encoding
1.5 for DBCS encoding

Example: NCHARMULTIPLIER=1.5 expands an 8–byte field to 12 bytes

Syntax

NCHARMULTIPLIER=*n*

Required Argument

n
specifies the multiplication factor to use when transcoding

Range $0 < n \leq 4$

Comparisons

The NCHARMULTIPLIER= data set option overrides the NCHARMULTIPLIER= LIBNAME statement option.

The NCHARMULTIPLIER= LIBNAME statement option overrides the CASNCHARMULTIPLIER= system option.

See Also

- "CASNCHARMULTIPLIER= System Option" in *SAS System Options: Reference*
- "NCHARMULTIPLIER= Data Set Option" in *SAS Data Set Options: Reference*

PORT= LIBNAME Statement Option

Specifies the network port number on which the controller listens for client connections.

Valid in: CAS LIBNAME Statement

Categories: Data Access
CAS

Default: Port 0 is assumed if the SESSREF= option and the PORT= option are not specified.

Note: Otherwise, the engine checks for a value in the CASPORT= system option and then in the _CASPORT_ macro variable.

Syntax

PORT=*integer*

Required Argument

integer
specifies the network port number.

MAXTABLEMEM= LIBNAME Statement Option

Specifies the maximum amount of memory, in bytes, to allocate for an in-memory table.

Valid in: CAS LIBNAME Statement

Categories: Data Access
CAS

Default: 16M

Syntax

MAXTABLEMEM=*integer* | *integer*K | *integer*M | *integer*G

Required Arguments

integer
specifies the total number of bytes to allocate.

***integer*K**
specifies the total number of kilobytes to allocate.

***integer*M**
specifies the total number of megabytes to allocate.

***integer*G**
specifies the total number of gigabytes to allocate.

Details

Integer values are always converted to the nearest whole megabyte but not less than 1 megabyte. Specifying 0 indicates to use the value from the MAXTABLEMEM= session option.

Comparisons

The MAXTABLEMEM= data set option overrides the MAXTABLEMEM= LIBNAME statement option.

Note: The value for either the data set option or the LIBNAME statement option is limited to the value for the MAXTABLEMEM= session option.

See Also

- "MAXTABLEMEM= Data Set Option" in *SAS Data Set Options: Reference*
- "MAXTABLEMEM= Session Option"

READTRANSFERSIZE= LIBNAME Statement Option

Specifies the maximum number of bytes that the CAS engine reads when transferring data from the CAS server.

Valid in: CAS LIBNAME Statement

Categories: Data Access
CAS

Alias: RTS

Default: 500MB

Restriction: This option affects only Read access.

Syntax

READTRANSFERSIZE=*integer* | *integer*K | *integer*M | *integer*G

Required Arguments

integer
specifies the total number of bytes to read.

***integer*K**
specifies the total number of kilobytes to read.

***integer*M**
specifies the total number of megabytes to read.

***integer*G**
specifies the total number of gigabytes to read.

Details

If the entire result of the read request is smaller than the value of the READTRANSFERSIZE= option, only the necessary number of bytes are transferred. This situation can occur if either the table size or the value of the DATALIMIT= option is smaller than the value of the READTRANSFERSIZE= option.

The following scenarios demonstrate different use cases for the READTRANSFERSIZE= option:

- Table size=45MB, DATALIMIT=100MB, READTRANSFERSIZE=500MB. The entire table is handled in a single read request because the table size is less than or equal to DATALIMIT= and READTRANSFERSIZE=.
- Table size=110MB, DATALIMIT=100MB, READTRANSFERSIZE=500MB. Only 100MB of the table is read and handled in a single read request. Because DATALIMIT= is smaller than the table size and smaller than READTRANSFERSIZE=, this error results:

  ```
  ERROR:  The maximum allowed bytes of data have been fetched
          from SAS Cloud Analytic Services.
  ```

  If you encounter this error, you can summarize the data in the server first and then transfer the smaller summarized data set to SAS. Or you can increase this setting so that the large data transfer can proceed without error.
- Table size=2GB, DATALIMIT="ALL", READTRANSFERSIZE=500MB. The entire table is handled in four read requests because the table size is equal to DATALIMIT= and greater than READTRANSFERSIZE=.

Comparisons

The READTRANSFERSIZE= data set option overrides the READTRANSFERSIZE= LIBNAME statement option.

See Also

- "DATALIMIT= LIBNAME Statement Option" on page 617
- "DATALIMIT= Data Set Option" in *SAS Data Set Options: Reference*
- "READTRANSFERSIZE= Data Set Option" in *SAS Data Set Options: Reference*
- "WRITETRANSFERSIZE= LIBNAME Statement Option" on page 629
- "WRITETRANSFERSIZE= Data Set Option" in *SAS Data Set Options: Reference*

SESSREF= LIBNAME Statement Option

Specifies the name of the CAS session to which you want to connect.

Valid in: CAS LIBNAME Statement

Categories: Data Access
CAS

Tip: Using this option is a preferred alternative to HOST= and PORT=

Syntax

SESSREF=*session-name*

Required Argument

session-name
specifies the name of the CAS session to which you want to connect.

TAG= LIBNAME Statement Option

Specifies a multi-level prefix for a CAS table that is used to identify the table to both the CAS server and SAS.

Valid in: CAS LIBNAME Statement

Categories: Data Access
CAS

Syntax

TAG=*tag-name*

Required Argument

tag-name
specifies a user-defined string to use for constructing table names.

Details

The TAG= option allows a SAS session to access CAS tables that have multi-level names without requiring the global VALIDMEMNAME=EXTEND option. A SAS table is identified by its libref and a table name (for example, mycas.mytab). Table names in SAS Cloud Analytic Services can have more than two levels (for example, lev1.lev2.mytab).

If you use this option, SAS can access only the in-memory tables that are prefixed with the value that you specified.

The value of the engine's TAG option is the CAS table name up to a delimiting period. The CAS server knows the table by the full name, 'lev1.lev2.mytab', but the SAS session knows the table name only as 'mytab'. When the TAG option is used, the CAS engine produces the correct name for both the SAS session and the CAS server session.

Example

This example uses the TAG= option to specify that the tag name, 'lev1.lev2', prefixes to the table name, 'mytab'. The engine uses the table name "lev1.lev2.mytab" in the CAS actions that are sent to the server.

```
libname mycas cas tag="lev1.lev2";
proc print data=mycas.mytab;
run;
```

See Also

"TAG= Data Set Option" in *SAS Data Set Options: Reference*

TIMEOUTSECONDS= LIBNAME Statement Option

Specifies the client inactivity time-out interval in seconds. If SAS stops transferring data to the server, the server starts the inactivity timer for the specified duration and cancels the request if SAS does not resume data transfer.

Category:	CAS
Alias:	TIMEOUT=
Default:	1800 seconds
Range:	30–3600 seconds

Syntax

TIMEOUTSECONDS=*n*

Required Argument

n
specifies the client inactivity time-out interval in seconds

Comparisons

The TIMEOUTSECONDS= LIBNAME statement option overrides the TIMEOUTSECONDS= data set option.

Example: Set a Client Inactivity Time-out

This example shows how you can use the TIMEOUTSECONDS= LIBNAME and Data Set option to set a client inactivity time-out. The output from PROC CAS shows that the TIMEOUTSECONDS= option is sent with success in the addTable action.

```
libname timeout cas timeoutSeconds=45;

options nodscas;

data timeout.t1;
   do i = 0 to 2e8;
  output;
   end;
run;

proc cas; history; run; quit;
```

```
NOTE: 11: action table.addTable / table='T1', caslib='CASUSER(your-user-name)',
vars={ {name='i', length=8, rType='NUMERIC', offset=0,
formattedLength=12, nfl=0, nfd=0} }, recLen=8, timeoutSeconds=30; /* (SUCCESS) */
```

See Also

"TIMEOUTSECONDS= Data Set Option" in *SAS Data Set Options: Reference*

TRANSCODE_FAIL= LIBNAME Statement Option

Specifies how the engine handles transcoding errors.

Valid in: CAS LIBNAME Statement, DATA and PROC steps

Categories: Data Access
CAS

Default: ERROR

Syntax

TRANSCODE_FAIL=ERROR | WARN | SILENT

Required Arguments

ERROR
Writes an error message to the SAS log. Processing stops.

WARN
Writes a warning message to the SAS log. Processing continues.

SILENT
Transcoding errors are ignored. Processing continues.

Comparisons

The TRANSCODE_FAIL= data set option overrides the TRANSCODE_FAIL= LIBNAME statement option.

See Also

"TRANSCODE_FAIL= Data Set Option" in *SAS Data Set Options: Reference*

UUID= LIBNAME Statement Option

Connects the CAS engine to the CAS session that is identified in the UUID.

Valid in: CAS LIBNAME Statement

Categories: Data Access
CAS

Tip: The "SESSREF= LIBNAME Statement Option" on page 624 is the preferred option for connecting to a CAS session.

Example: This example starts a session with the CAS statement and saves the UUID in the MyUUID macro variable. A libref is assigned and connects to that session.

```
cas casauto host="cloud.example.com" port=5570 uuid=MyUUID;
```

```
< Perform some additional tasks. >
libname mycas cas uuid="&MyUUID";

data mycas.female;
    set sashelp.class;
    where sex='F';
run;
```

Syntax

UUID=*"session-uuid"*

Required Argument

"session-uuid"
specifies the universally unique identifier (UUID) of an existing session. You must obtain the UUID from the existing session before you can specify it in this option.

UUIDMAC= LIBNAME Statement Option

Specifies a macro variable that stores a UUID for any session created by the engine.

Valid in: CAS LIBNAME Statement

Categories: Data Access
CAS

Default: If you do not specify this option, the UUID is stored in the _IOCASUUID_ macro variable value.

Syntax

UUIDMAC=*"macro-variable-name"*

Required Argument

"macro-variable-name"
specifies a name or string into which the engine saves the UUID of the current CAS session.

WRITETRANSFERSIZE= LIBNAME Statement Option

Specifies the maximum number of bytes that the CAS engine writes when transferring data to the CAS server.

Valid in: CAS LIBNAME Statement

Categories: Data Access
CAS

Alias: WTS

Default: 512KB

Restriction: This option affects only Write access.

Syntax

WRITETRANSFERSIZE=*integer* | *integer*K | *integer*M | *integer*G

Required Arguments

integer
specifies the maximum write buffer size in bytes.

***integer*K**
specifies the maximum write buffer size in kilobytes.

***integer*M**
specifies the maximum write buffer size in megabytes.

***integer*G**
specifies the maximum write buffer size in gigabytes.

Comparisons

The WRITETRANSFERSIZE= data set option overrides the WRITETRANSFERSIZE= LIBNAME statement option.

See Also

- "DATALIMIT= LIBNAME Statement Option" on page 617
- "DATALIMIT= Data Set Option" in *SAS Data Set Options: Reference*

- "READTRANSFERSIZE= LIBNAME Statement Option" on page 622
- "READTRANSFERSIZE= Data Set Option" in *SAS Data Set Options: Reference*
- "WRITETRANSFERSIZE= LIBNAME Statement Option" on page 629
- "WRITETRANSFERSIZE= Data Set Option" in *SAS Data Set Options: Reference*

11

Data Set Options

About Data Set Options for CAS 631

About Data Set Options for CAS

You can find information about all data set options, including those that you can use with CAS, in *SAS Data Set Options: Reference*. You can use any data set option with CAS that lists the CAS category in the documentation.

12

Macro Variables

Dictionary **633**
CASHOST Macro Variable 633
_CASHOST_PERSONAL_ Macro Variable 635
_CASHOST_SHARED_ Macro Variable 636
CASHOSTCONT Macro Variable 637
CASPORT Macro Variable 637
IOCASUUID Macro Variable 638
SESSREF Macro Variable 639
CAS_ADMIN_GLOBAL_UDFMTS Macro Variable 639
CASSTMTERR Macro Variable 642
SYSVIYARELEASE Automatic Macro Variable 643
SYSVIYAVERSION Automatic Macro Variable 643

Dictionary

CASHOST Macro Variable

Contains the primary controller host name and the backup controller host name, if available, on the CAS server.

Type: Global macro variable

Default: The value of system option CASHOST, if set. Otherwise, not defined.

Range: 1–256 characters

Interactions: When set, this macro variable overrides SAS system option CASHOST.
The CAS statement HOST= option overrides this macro variable.

Notes: When system option CASHOST is set, macro variable _CASHOST_ is set to the value that was specified for system option CASHOST. However, setting macro variable _CASHOST_ does not affect the CASHOST system option value.

When a connection is established, macro variable _CASHOST_ and system option CASHOST are updated to specify the current primary controller host name and, if available, the backup controller host name.

If the primary controller fails, the backup controller becomes the primary controller. Afterward, when a connection is established, macro variable _CASHOST_ and system option CASHOST are updated to specify the backup controller host name only. When the primary server is subsequently restored and a new session connection is established, the CASHOST system option and macro variable _CASHOST_ are updated with the current primary and secondary controller host names.

Tips: To display information about the controllers, run the following code:

```
proc cas;
   builtins.listNodes result = r;
   print r.nodelist.where(role eq "controller");
run;
quit;
```

If you want to delete macro variable _CASHOST_ and allow SAS system option CASHOST to prevail, use the following statement:

```
%symdel _CASHOST_;
```

Syntax

%let _CASHOST_= *primary-controller-host-name*;

%let _CASHOST_= (*primary-controller-host-name*, *backup-controller-host-name*);

Syntax Description

primary-controller-host-name
specifies the primary controller host name.

Note The name can be specified with or without quotation marks.

backup-controller-host-name
specifies the backup controller host name. The name can be specified with or without quotation marks.

Requirements The primary controller host name and the backup host name must be enclosed in parentheses and must be separated by a comma or a space.

If you enclose the host names in quotation marks, each name must be enclosed in its own pair of quotation marks. Do not enclose the entire list in one set of quotation marks.

Note The host name can be specified with or without quotation marks.

Examples

Example 1

Specify the primary controller host name only:

```
%let _CASHOST_ = cloud.example.com;
```

Example 2

Specify the primary controller host name and the backup controller host name, and enclose the names in quotation marks:

```
%let _CASHOST_ = ("cloud.example.com", "cloudbackup.example.com");
```

_CASHOST_PERSONAL_ Macro Variable

Contains the host name of the personal CAS server, if enabled, which is localhost.

Type: Global macro variable

Default: If a personal CAS server is enabled, localhost. Otherwise, this macro variable is not defined.

Range: 1–256 characters

Tip: Use macro function %SYMEXIST to determine whether this macro variable is set before attempting to connect to the personal CAS server. See "%SYMEXIST Macro Function" in *SAS Macro Language: Reference*.

Example

Connect to the personal CAS server:

```
cas mysess_personal host=&_cashost_personal_;
```

Here is an example of the output that is written to the SAS log.

```
NOTE: The session MYSESS_PERSONAL connected successfully to Cloud Analytic Services
      localhost using port 5570. The UUID is session-UUID.
      The user is sasdemo and the active caslib is CASUSER(sasdemo).
NOTE: The SAS option SESSREF was updated with the value MYSESS_PERSONAL.
NOTE: The SAS macro _SESSREF_ was updated with the value MYSESS_PERSONAL.
NOTE: The session is using 0 workers.
```

See Also

- "Personal CAS Server" in *SAS Viya Platform: SAS Cloud Analytic Services*
- "Example 10: Start a Session on a Personal CAS Server" on page 731
- "Example 11: Switch between a Personal CAS Server and the Shared CAS Server" on page 732

_CASHOST_SHARED_ Macro Variable

Contains the host name of the shared CAS server enclosed in quotation marks.

Type: Global macro variable

Default: The value of macro variable _CASHOSTCONT_ on page 637, which contains the host name of the primary controller.

Range: 1–256 characters

Note: The host name in _CASHOST_SHARED_ is enclosed in quotation marks.

Tip: Use this macro variable to connect to the shared CAS server after connecting to the personal CAS server.

Example

Connect to the shared CAS server:

```
cas mysess_shared host=&_cashost_shared_;
```

Here is an example of the output that is written to the SAS log.

```
NOTE: The session MYSESS_SHARED connected successfully to Cloud Analytic Services
      cloud.example.com using port 5570. The UUID is session-UUID.
      The user is sasdemo and the active caslib is CASUSER(sasdemo).
NOTE: The SAS option SESSREF was updated with the value MYSESS_SHARED.
NOTE: The SAS macro _SESSREF_ was updated with the value MYSESS_SHARED.
NOTE: The session is using n workers.
```

See Also

"Example 11: Switch between a Personal CAS Server and the Shared CAS Server" on page 732

CASHOSTCONT Macro Variable

Contains the host name of the primary controller on the CAS server.

Type: Global macro variable

Default: The primary host name specified in system option CASHOST, if set. Otherwise, not defined.

Range: 1–256 characters

Notes: Macro variable _CASHOSTCONT_ stores the host name of the primary controller only.

When system option CASHOST is set, macro variable _CASHOSTCONT_ is set to the primary controller host name that was specified in system option CASHOST. Macro variable _CASHOST_ is also set to the specified primary controller host name and the backup controller host name, if specified.

If the primary controller fails, the backup controller becomes the primary controller. Afterward, when a new connection is established, macro variable _CASHOSTCONT_ is updated to specify the backup controller host name as the current primary controller. Macro variable _CASHOST_ and system option CASHOST are also updated.

Tip: To display information about the controllers, run the following code:

```
proc cas;
   builtins.listNodes result = r;
   print r.nodelist.where(role eq "controller");
run;
quit;
```

See: "_CASHOST_ Macro Variable" on page 633

"CASHOST= System Option" in *SAS System Options: Reference*

Example: Display the current primary controller host name:

```
%put NOTE: The Primary Controller is: &_cashostcont_;
```

CASPORT Macro Variable

Contains the CAS server port.

Type: Global macro variable

Default: The value of system option CASPORT, if set. Otherwise, not defined.

Range: 0–65535

Interactions: When set, this macro variable overrides SAS system option CASPORT.

The CAS statement PORT= option overrides this macro variable.

Notes: When set to 0, CAS selects a port number.

When system option CASPORT is set, macro variable _CASPORT_ is set to the value specified for system option CASPORT. However, setting macro variable _CASPORT_ does not affect the CASPORT system option value.

Tip: If you want to delete macro variable _CASPORT_ and allow SAS system option CASPORT to prevail, use the following statement:

```
%symdel _CASPORT_;
```

Syntax

%let _CASPORT_= *port*;

IOCASUUID Macro Variable

Contains the UUID of the session that is associated with the last CAS LIBNAME engine libref that was created.

Type: Global macro variable

Tip: The current value of _IOCASUUID_ is overwritten each time a CAS LIBNAME engine libref is created. To preserve the UUIDs when multiple CAS LIBNAME engine librefs are created, use the UUIDMAC= option in the LIBNAME statements to specify unique macro variable names. See “UUIDMAC= LIBNAME Statement Option” on page 628.

Example

Create CAS LIBNAME engine libref Mycas and display the UUID of the session with which it is associated:

```
libname mycas cas;
%put Mycas UUID: &_IOCASUUID_;
```

Here is the output.

```
Mycas UUID: 3e7e18e7-a9d5-704d-b919-31249fe0fd20
```

See Also

Chapter 2, “CAS LIBNAME Engine ,” on page 37

SESSREF Macro Variable

Contains the session reference of the active SAS Cloud Analytic Services session.

Type: Global macro variable

Default: Not defined

Interactions: When you use the CAS statement to create a new session, this macro variable is automatically set to the session reference of the new session, which is CASAUTO by default.

When you set SAS system option SESSREF, this macro variable is automatically set to the same value.

CAS_ADMIN_GLOBAL_UDFMTS Macro Variable

Enables a user to enable or disable an automatic request for escalation of administrative authority to Superuser before manipulating global user-defined format libraries using a CAS statement.

Type: Global macro variable

Default: Not defined

Interaction: When defined, this macro variable overrides SAS client-side environment variable CAS_ADMIN_GLOBAL_UDFMTS.

Possible Values

YES| ON| TRUE	specifies that escalation of administrative authority to Superuser is requested before manipulating user-defined format libraries using a CAS statement. If the UDF library is in global scope, env.CAS_ADMIN_GLOBAL_UDFMTS is enabled on the CAS server, and the user does not have administrative authority, an error results.
NO| OFF| FALSE	specifies that escalation of administrative authority to Superuser is not requested before manipulating user-defined format libraries using a CAS statement. If the UDF library is in global scope and env.CAS_ADMIN_GLOBAL_UDFMTS is enabled on the CAS server, an error results.

Details

Macro variable CAS_ADMIN_GLOBAL_UDFMTS enables a user to specify whether escalation of administrative authority to Superuser is requested before manipulating user-defined format (UDF) libraries using a CAS statement. These operations include promoting a UDF library to global scope or dropping a global UDF library. By default, users can perform these operations without administrative authority. However, on the server side, the CAS administrator can specify env.CAS_ADMIN_GLOBAL_UDFMTS=YES in the CAS server configuration to require an administrative-credentials check when manipulating global UDF libraries.

On the client side, an administrator can use client-side environment variable CAS_ADMIN_GLOBAL_UDFMTS or macro variable CAS_ADMIN_GLOBAL_UDFMTS to specify whether escalation of administrative authority to Superuser is requested before manipulating a UDF library using a CAS statement. Client-side environment variable CAS_ADMIN_GLOBAL_UDFMTS takes the same values as macro variable CAS_ADMIN_GLOBAL_UDFMTS, which have the same effect. When client-side environment variable CAS_ADMIN_GLOBAL_UDFMTS is set, macro variable CAS_ADMIN_GLOBAL_UDFMTS can be used to override its current setting.

An escalation request can be made regardless of whether it is required by the CAS server. The CAS server considers the escalation request only when env.CAS_ADMIN_GLOBAL_UDFMTS is enabled in its configuration. If an administrator wants to manipulate global UDF libraries and env.CAS_ADMIN_GLOBAL_UDFMTS is enabled on the CAS server, the administrator must enable CAS_ADMIN_GLOBAL_UDFMTS on the client to request escalation to Superuser as required by the CAS server. If the env.CAS_ADMIN_GLOBAL_UDFMTS setting on the CAS server is unknown, the administrator can enable CAS_ADMIN_GLOBAL_UDFMTS on the client to request escalation to Superuser in case it is required.

When manipulating UDF libraries in session scope that also exist in global scope, an administrator can disable CAS_ADMIN_GLOBAL_UDFMTS on the client side (default) to prevent the accidental manipulation of the same UDF libraries that are in global scope. For example, when a UDF library is dropped, CAS first looks for the UDF library in session scope. If it does not exist in session scope, it then looks in global scope, which might not be desired.

When a user attempts to manipulate a UDF library using a CAS statement, the CAS statement does the following to determine whether to request escalation of administrative authority to Superuser:

1. The CAS statement determines whether macro variable CAS_ADMIN_GLOBAL_UDFMTS is set on the SAS client. If macro variable CAS_ADMIN_GLOBAL_UDFMTS is set, it uses the macro variable value to determine whether to request escalation to Superuser, in case it is needed.
2. If macro variable CAS_ADMIN_GLOBAL_UDFMTS is not set, the CAS statement determines whether environment variable CAS_ADMIN_GLOBAL_UDFMTS is set on the client. If it is set, the CAS statement uses the CAS_ADMIN_GLOBAL_UDFMTS environment variable value to determine whether to request escalation to Superuser, in case it is needed.

3 If both macro variable CAS_ADMIN_GLOBAL_UDFMTS and client environment variable CAS_ADMIN_GLOBAL_UDFMTS are not set, by default, the CAS statement does not request escalation to Superuser.

When CAS_ADMIN_GLOBAL_UDFMTS is enabled on the client, the following occurs:

- When an administrator attempts to manipulate a UDF library using a CAS statement, the CAS statement requests escalation to Superuser before performing the operation. Because the user has administrator authority, the request is granted, and the operation is carried out.
- When a user who does not have administrative authority attempts to manipulate a UDF library using a CAS statement, the CAS statement requests escalation to Superuser before performing the operation, but the request is denied. (The escalation request failure is not logged.) In that case, one of the following then occurs:
  - If the UDF library that is being manipulated is in session scope, escalation to Superuser is not required, and the operation is carried out.
  - If the UDF library is in global scope and env.CAS_ADMIN_GLOBAL_UDFMTS is disabled on the CAS server, escalation to Superuser is not required, and the operation is carried out.
  - If the UDF library is in global scope and env.CAS_ADMIN_GLOBAL_UDFMTS is enabled on the CAS server, the operation fails, and an error is written to the SAS log. (This can occur unintentionally when dropping a UDF library.) In that case, if the operation was intentional, the user must contact the CAS administrator to perform the desired operation.

When CAS_ADMIN_GLOBAL_UDFMTS is disabled on the client, the CAS statement does not request escalation to Superuser when a user manipulates a UDF library. In that case, if env.CAS_ADMIN_GLOBAL_UDFMTS is enabled on the CAS server and the UDF library is in global scope, an error results. If env.CAS_ADMIN_GLOBAL_UDFMTS is disabled on the CAS server, escalation to Superuser is not required and the operation is carried out.

For users who do not have administrative authority, requests for escalation to Superuser always fail. Non-administrative users can use client environment variable CAS_ADMIN_GLOBAL_UDFMTS or macro variable CAS_ADMIN_GLOBAL_UDFMTS to disable the automatic escalation requests, if it is currently enabled on their client. That would eliminate steps in their program execution that would have no affect on the result.

Examples

Example 1

This example assumes that you have administrative authority and are working with UDF library myFmtLib in session scope and global scope. You want to drop

myFmtLib from session scope, but you want to ensure that you do not drop myFmtLib from global scope by accident. You are not sure whether CAS_ADMIN_GLOBAL_UDFMTS is set on the client. To ensure that the automatic request for escalation to Superuser is disabled before UDF library myFmtLib is dropped from session scope:

```
%let CAS_ADMIN_GLOBAL_UDFMTS=no;
cas casauto dropfmtlib fmtlibname=myFmtLib;
```

This is useful when your UDF library exists in both session scope and global scope and you want to ensure that you do not accidentally delete your UDF library from global scope.

Example 2

This example assumes that you have administrative authority and you want to drop UDF library myFmtLib from global scope but you are not sure whether env.CAS_ADMIN_GLOBAL_UDFMTS is enabled on the CAS server:

```
%let CAS_ADMIN_GLOBAL_UDFMTS=yes;
cas casauto dropfmtlib fmtlibname=myFmtLib;
```

Example 3

This example deletes macro variable CAS_ADMIN_GLOBAL_UDFMTS, which allows the client-side CAS_ADMIN_GLOBAL_UDFMTS environment variable setting or the default to prevail:

```
%symdel CAS_ADMIN_GLOBAL_UDFMTS;
```

See Also

" CAS Environment Variables" in *SAS Viya Platform: SAS Cloud Analytic Services*

CASSTMTERR Macro Variable

Contains the status of the last CAS statement that was executed.

Type: Global macro variable

Possible Values

0 The CAS statement completed successfully.

1 The CAS statement failed to execute. An error message is written to the SAS log.

2 The CAS statement completed with a warning. A warning message is written to the SAS log.

Example

Create session Casauto and, if successful, add format library Myfmtlib:

```
cas casauto sessopts=(caslib=casuser);
%if &CASSTMTERR ne 1 %then %do;
   cas casauto addfmtlib fmtlibname=myFmtLib
      table="myFmtLib.sashdat" caslib=casuser;
%end;
```

SYSVIYARELEASE Automatic Macro Variable

Contains the SAS Viya platform cadence release number.

Type: Automatic macro variable (read only)

Details

The SYSVIYARELEASE automatic macro variable contains the SAS Viya platform cadence release. The cadence release string format is as follows:

YYYYmmdd.nnnnnnnnnnnn

Example

Display the SAS Viya platform release:

```
%put SAS Viya platform release: &sysviyarelease;
```

Here is an example of the output.

```
SAS Viya platform release: 20230901.1693566968705
```

SYSVIYAVERSION Automatic Macro Variable

Contains the SAS Viya platform cadence version.

Type: Automatic macro variable (read only)

Details

About SAS Viya Platform Cadence Versions

The SAS Viya platform release cadence consists of Stable and Long-Term Support (LTS) releases. Stable releases typically have a cadence of one month, and LTS releases have a cadence of approximately six months. The cadence version is stored in automatic macro variable SYSVIYAVERSION.

The stable release version string format is:

```
Stable YYYY.nn
```

where *YYYY* is the release year and *nn* is the release month in the release year. Here is the cadence version for the 2025 April Stable release:

```
Stable 2025.04
```

The LTS release cadence version string format is:

```
LTS YYYY.nn
```

where *YYYY* is the release year and *nn* is the release month in the release year. Here is the cadence version for the 2024 November LTS release:

```
LTS 2024.09
```

Example

Display the SAS Viya platform version:

```
%put SAS Viya platform version: &sysviyaversion;
```

Here is an example of the output.

```
SAS Viya platform version: LTS 2025.04
```

13

Session Options

Setting Session Options **646**
About the Session Options 646
Setting Session Options for a New Session 646
Setting Options for Existing Sessions 646

Session Options by Category **648**

Dictionary **650**
APPTAG= Session Option 650
AZURETENANTID= Session Option 651
BIGINTPROCESSING= Session Option 652
CASLIB= Session Option 654
CMPLIB= Session Option 655
CMPOPT= Session Option 657
COLLATE= Session Option 659
DATASTEPFMTERR= Session Option 660
DATASTEPMERGENOBY= Session Option 661
DATASTEPMSGSUMLEVEL= Session Option 661
DATASTEPREPLACETABLE= Session Option 662
DATASTEPVARLENCHK= Session Option 663
DATESTYLE= Session Option 664
DEBUGOPT= Session Option 666
DEFAULTMEMORYFORMAT= Session Option 667
DEFAULTTABLEREPLICATION= Session Option 669
DQLOCALE= Session Option 670
DQQKBREPOLOC= Session Option 670
DQSETUPLOC= Session Option 671
HISTORYLINELENGTH= Session Option 671
INTERVALDS= Session Option 673
LOCALE= Session Option 673
LOGFLUSHTIME= Session Option 674
MAXCORESPERWORKER= Session Option 675
MAXTABLEMEM= Session Option 679
MESSAGELEVEL= Session Option 681
METRICS= Session Option 682
NLDECSEPARATOR= Session Option 684
NWORKERS= Session Option 686
S2FORMATSEARCH= Session Option 687

SESSIONTABLEREDISTUPPOLICY= Session Option 688
SUBSETSESSIONCOPIES= Session Option 689
TIMEOUT= Session Option 690
TIMEZONE= Session Option 690
TRANSFERSESSION= Session Option 692
USEGPU= Session Option 693
YEARCUTOFF= Session Option 693

Setting Session Options

About the Session Options

The session options control various properties of your SAS Cloud Analytic Services session. To list the properties and their current setting for a session, use the LISTSESSOPTS option in a CAS statement. See "LISTSESSOPTS " on page 710. To see the setting for a specific property, use the GETSESSOPT function. See "GETSESSOPT Function" in *SAS Functions and CALL Routines: Reference*.

Setting Session Options for a New Session

How the Session Option Values Are Determined

When you create a new session, the value for each of the session properties is provided by the following sources in descending order of precedence:

- options specified in the SESSOPTS= option in the CAS statement.
- when you set SAS system options CASTIMEOUT and CASNWORKERS in SAS, CASTIMEOUT for session option TIMEOUT and CASNWORKERS for session option NWORKERS.

  Note: The CASTIMEOUT and CASNWORKERS system options have effect only after you set them in SAS. Otherwise, they are ignored.

- configuration parameters specified in the SAS Cloud Analytic Services configuration file. See SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services*.
- command-line options that are used in the server start-up command.
- SAS Cloud Analytic Services system defaults.

Overriding the Default Session Option Values

To override the default session option values for a new session, use the option shown in the following table to complete the desired task.

Task	Option to use:
Override one or more session options for a new session only.	CAS statement SESSOPTS= option
Override the DATESTYLE option value for all subsequently created sessions.	SAS system option DATESTYLE=[1, 2]
Override the NWORKERS option value for all subsequently created sessions.	SAS system option CASNWORKERS=[1, 2]
Override the TIMEOUT option value for all subsequently created sessions.	SAS system option CASTIMEOUT=[1, 2]
Override the YEARCUTOFF option value for all subsequently created sessions.	SAS system option YEARCUTOFF=[1, 2]

1 Specifying this option in the CAS statement SESSOPTS= option overrides the default value for the specified session only.

2 Setting this SAS system option does not affect the existing sessions.

Setting Options for Existing Sessions

For an existing session, use the option shown in the following table to complete the desired task.

Task	Option to use:
Change one or more session options for a specific session.	CAS statement SESSOPTS option **Note:** You can also use the setSessOpt action in the sessionProp action set to change session options for a session. See "Set session option" in *SAS Viya Platform: System Programming Guide*.
Change one or more session options for the active session.[1]	SAS system option CASSESSOPTS= (alias SESSOPTS=)

Task	Option to use:
Specify a caslib for the active session.[1]	SAS system option CASLIB=

1 SAS system option SESSREF stores the name of the currently active session.

Session Options by Category

Category	Language Elements	Description
Action	APPTAG= Session Option (p. 650)	specifies a string that is included in the JSON log line of some CAS log messages on the CAS server.
Action History	HISTORYLINELENGTH= Session Option (p. 671)	specifies the number of bytes to use for reporting action names and action parameters in a line of history.
Authentication	AZURETENANTID= Session Option (p. 651)	specifies the tenant ID for connecting to a Microsoft Azure storage system in a session.
Caslib	CASLIB= Session Option (p. 654)	specifies the caslib name to set as the active caslib.
	MAXTABLEMEM= Session Option (p. 679)	specifies the maximum amount of memory, in bytes, to allocate for a table.
Data Processing	BIGINTPROCESSING= Session Option (p. 652)	specifies whether BIGINT processing is enabled for the CAS session.
	INTERVALDS= Session Option (p. 673)	specifies one or more interval-name=value pairs, where the value is the name of a data set that contains user-defined intervals.
Data Quality	DQLOCALE= Session Option (p. 670)	specifies the default 5-letter SAS Quality Knowledge Base (QKB) ISO locale code to use as the default locale for data quality (DQ) operations.
	DQQKBREPOLOC= Session Option (p. 670)	specifies the name of the caslib where SAS Quality Knowledge Bases are stored.
	DQSETUPLOC= Session Option (p. 671)	specifies the name of the default SAS Quality Knowledge Base (QKB) to use for data quality operations.
DATA Step	DATASTEPFMTERR= Session Option (p. 660)	specifies how SAS reacts when a specified variable format cannot be found.
	DATASTEPMERGENOBY= Session Option (p. 661)	specifies whether a warning message is generated, an error message is generated, or no message is generated when a DATA step merge is executed without a BY statement.

Category	Language Elements	Description
	DATASTEPMSGSUMLEVEL= Session Option (p. 661)	specifies the DATA step message summary level.
	DATASTEPREPLACETABLE = Session Option (p. 662)	specifies whether a DATA step can replace an existing table.
	DATASTEPVARLENCHK= Session Option (p. 663)	specifies whether a warning message is generated, an error message is generated, or no message is generated when truncation occurs during a DATA step append.
Date and Time	TIMEZONE= Session Option (p. 690)	specifies the user local time zone.
	YEARCUTOFF= Session Option (p. 693)	specifies the first year of a 100-year span that is used by date informats and functions to read a 2-digit year.
Debugging	DEBUGOPT= Session Option (p. 666)	specifies the host and port for the SAS Code Debugger in a session.
Files: SAS Files	CMPLIB= Session Option (p. 655)	specifies one or more CMP libraries that contain compiler subroutines to include during program compilation.
Formats	DATESTYLE= Session Option (p. 664)	specifies the sequence of month, day, and year when ANYDTDTE, ANYDTDTM, or ANYDTTME informat data is ambiguous.
	NLDECSEPARATOR= Session Option (p. 684)	specifies whether SAS uses locale-sensitive decimal separators or US English decimal separators (dot) in formatted numeric output.
	S2FORMATSEARCH= Session Option (p. 687)	specifies the format search list for format libraries that have been published to SingleStore for use by the SAS Embedded Process.
Input Control	INTERVALDS= Session Option (p. 673)	specifies one or more interval-name=value pairs, where the value is the name of a data set that contains user-defined intervals.
Localization	LOCALE= Session Option (p. 673)	specifies the locale to use for sorting and formatting.
Log	LOGFLUSHTIME= Session Option (p. 674)	specifies the log flush time, in milliseconds.
	MESSAGELEVEL= Session Option (p. 681)	specifies the log message level.
	METRICS= Session Option (p. 682)	specifies whether to include default detailed performance metrics reports in the SAS log.
Memory	DEFAULTMEMORYFORMAT = Session Option (p. 667)	specifies the default memory format for session tables.
Session	MAXCORESPERWORKER= Session Option (p. 675)	specifies the maximum number of cores per worker for the session.

Category	Language Elements	Description
	NWORKERS= Session Option (p. 686)	specifies the number of worker nodes for a new session.
	SESSIONTABLEREDISTUPPOLICY= Session Option (p. 688)	specifies a session table redistribution policy that is applied when the number of worker pods is scaled up on a running distributed CAS server (MPP).
	SUBSETSESSIONCOPIES= Session Option (p. 689)	specifies the number of backup copies to create of subset session in-memory table blocks for failover.
	TIMEOUT= Session Option (p. 690)	specifies the SAS Cloud Analytic Services session time-out in seconds for a new or existing session.
	TRANSFERSESSION= Session Option (p. 692)	specifies whether this session is transferred to the new server when a CAS server state transfer occurs.
	USEGPU= Session Option (p. 693)	specifies whether work is offloaded to the Graphics Processing Unit (GPU), if it is available.
Sort	COLLATE= Session Option (p. 659)	specifies the collating sequence for sorting.
System Administration: Performance	CMPOPT= Session Option (p. 657)	specifies the type of code generation optimizations to use in the SAS language compiler.
Table Replication	DEFAULTTABLEREPLICATION= Session Option (p. 669)	specifies the default replication factor for new tables for a session.

Dictionary

APPTAG= Session Option

specifies a string that is included in the JSON log line of some CAS log messages on the CAS server.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Action

Default: No prefix

Note: Currently, this option is not used.

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

SAS Viya Logging Facility for SAS Programs and Jobs

Syntax

APPTAG=*"tag-string"*

AZURETENANTID= Session Option

specifies the tenant ID for connecting to a Microsoft Azure storage system in a session.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Authentication

Default: The value of CAS server configuration option cas.AZURETENANTID, if set.

Requirement: Session option AZURETENANTID= or CAS server configuration option cas.AZURETENANTID= is required for a Microsoft Azure storage system if your deployment on the SAS Viya platform is not configured for single sign-on with Azure Active Directory (Azure AD). If your deployment is configured for single sign-on using the OpenID Connect (OIDC) authentication protocol, then session option AZURETENANTID= and CAS server configuration option cas.AZURETENANTID= are not required for authentication. For more information see *SAS Viya Platform: Authentication* and SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services*.

Interaction: This option overrides CAS server configuration option cas.AZURETENANTID only for this session.

Note: This session option might have already been set by your CAS administrator.

See: *SAS Viya Platform: Authentication*

SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

AZURETENANTID=*"tenant-id"*

Examples

Example 1: Set Session Option AZURETENANTID

This example sets session option AZURETENANTID in session Casauto:

```
cas casauto sessopts=(azuretenantid="tenant-id");
```

Example 2: Get the Current Value of Session Option AZURETENANTID

This example gets the current value of session option AZURETENANTID in session Casauto:

```
%put Session Casauto: AZURETENANTID=%sysfunc(getsessopt(casauto,
azuretenantid));
```

The GETSESSOPT arguments are not enclosed in quotation marks when it is invoked by %SYSFUNC. Here is an example of the result.

```
Session Casauto: AZURETENANTID=tenant-id
```

BIGINTPROCESSING= Session Option

specifies whether BIGINT processing is enabled for the CAS session.

Valid in: CAS statement SESSOPTS option, CAS configuration file, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Data Processing

Default: The CAS server option cas.BigIntProcessing value, if set. Otherwise, FALSE.

Restriction: This session option is not honored by all of the available CAS actions. See "Details" on page 653.

Notes: Support for this option was added in 2025.12.

Data type BIGINT is the same as INT64. See "Numeric Data" on page 613.

The BIGINTPROCESSING= session option overrides CAS server option cas.BigIntProcessing for a session.

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

BIGINTPROCESSING=TRUE | FALSE

Parameter Values

TRUE
specifies that integer values are not automatically coerced into DOUBLE values for non-mathematical operations. This preserves full precision of the integer values for operations such as indexing, BY-group processing, and categorical analysis within CAS actions.

Note: For numeric calculations, BIGINT values are coerced into DOUBLE values, which might lead to a loss of precision for values that exceed 15–16 digits. If your program relies on the implicit conversion of BIGINT values into DOUBLE values, specifying TRUE might result in behavioral differences from prior releases.

FALSE
specifies that integer values are coerced into DOUBLE values. This is the default. When BIGINT data is read from a CAS table, integer values are coerced into DOUBLE values within CAS action processing. Setting this option to FALSE preserves compatibility with existing programs or workflows that assume traditional CAS behavior.

Details

The BIGINTPROCESSING= session option controls how some CAS actions handle BIGINT values. In SAS, by default, integer values are coerced into DOUBLE values before most of the calculations are performed in procedures and actions. For large integer values of 15–16 digits or more, this conversion can result in a loss of precision. The BIGINTPROCESSING= session option can be used to bypass the coercion of BIGINT values into DOUBLE values for applicable CAS actions in order to preserve the full precision of BIGINT values.

Currently, session option BIGINTPROCESSING= is honored by the following CAS actions only:

- ds2 actions runDS2 and runModel
- fedSql action execDirect
- table action fetch

When BIGINTPROCESSING= is TRUE, BIGINT processing in these CAS actions bypasses the coercion of BIGINT values into DOUBLE values. When BIGINTPROCESSING= is FALSE (default), these CAS actions use the legacy behavior of coercing BIGINT values into DOUBLE values. For other CAS actions, setting the BIGINTPROCESSING= session option has no effect.

Note: Some actions might already support BIGINT processing natively and do not use the BIGINTPROCESSING= session option.

Session option BIGINTPROCESSING= defaults to the value of CAS server option cas.BigIntProcessing, which defaults to FALSE. Session option

BIGINTPROCESSING= overrides CAS server option cas.BigintProcessing. A default value of FALSE helps preserve the legacy behavior of integers for the CAS actions that honor the BIGINTPROCESS= session option. To enable BIGINT processing for a CAS session, specify BIGINTPROCESSING=TRUE for that session. To enable BIGINT processing for all new CAS sessions, set CAS server option cas.BigintProcessing to TRUE.

Examples

Example 1: Get the Current BIGINTPROCESSING= Setting

This example shows how to use the GETSESSOPT function to get the current BIGINTPROCESSING= session option setting for session Casauto. For BIGINTPROCESSING, the GETSESSOPT function returns 0 if BIGINT processing is disabled or 1 if it is enabled. Here is the SAS code:

```
cas casauto sessopts=(caslib=casuser);
%put NOTE: Session CASAUTO: BIGINTPROCESSING=
%sysfunc(getsessopt(casauto,
   bigintprocessing)).;
```

```
NOTE: Session CASAUTO: BIGINTPROCESSING=0.
```

Example 2: Enable BIGINT Processing in a Session

This example shows how to enable BIGINT processing for session Casauto:

```
cas casauto sessopts=(bigintprocessing=true);
%put NOTE: Session CASAUTO: BIGINTPROCESSING=
%sysfunc(getsessopt(casauto,
   bigintprocessing)).;
```

```
NOTE: Session CASAUTO: BIGINTPROCESSING=1.
```

CASLIB= Session Option

specifies the caslib name to set as the active caslib.

Valid in: CAS statement SESSOPTS option, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Caslib

Syntax

CASLIB=*"caslib-reference-name"*

Parameter Description

"caslib-reference-name"
specifies the name of an existing caslib.

Requirement The specified caslib must already exist.

See "CASLIB Statement" on page 745

CMPLIB= Session Option

specifies one or more CMP libraries that contain compiler subroutines to include during program compilation.

Valid in: CAS statement SESSOPTS option, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Files: SAS Files

Note: Session option CMPLIB= operates independently of SAS system option CMPLIB= in the SAS client. Setting one does not affect the other.

Syntax

Form 1: **CMPLIB=***"caslib-name.cas-table-name <caslib-name.cas-table-name <...>>"*

Form 2: **CMPLIB=**'{ {CASLIB= *"caslib-name"*, NAME= *"cas-table-name"*} <..., {CASLIB= *"caslib-name"*, NAME= *"cas-table-name"*}>}'

Required Arguments

caslib-name
specifies a caslib that contains a CAS table.

cas-table-name
specifies a CAS table that contains compiler functions and subroutines.

Details

IMPORTANT When specifying a CMP library on the SAS client, use form 1 `cmplib="casuser.mycmpfuncs";` to set the session option. When specifying a CMP library on the CAS server, use form 2 `cmplib={ {CASLIB="casuser", NAME="mycmpfuncs"} }` to set the session option.

The CMPLIB= session option is used to specify the name of one or more CMP libraries that contain compiler functions and subroutines. In SAS Cloud Analytic Services, CMP function and subroutine libraries are stored in tables. The CMP libraries are searched in the order in which they are specified in this option. In the case of duplicate functions or subroutines, the last instance that is found in the CMP library search is used.

Examples

Example 1: Set Session Option CMPLIB on the CAS Server

The following examples specify one or more CMP libraries on the CAS server.

Example Code 13.1 *Specify a Single CMP Library:*

```
cas casauto sessopts=(cmplib='{ {CASLIB="casuser",
NAME="mycmpfuncs"} }');
```

Example Code 13.2 *Specify Multiple CMP Libraries:*

```
cas casauto sessopts=(cmplib='{ {CASLIB="ourcmplibs",
NAME="cmpfuncs"}, {CASLIB="casuser", NAME="mycmpfuncs"} }');
```

Example 2: Set Session Option CMPLIB on the SAS Client

The following examples specify one or more CMP libraries on the SAS client.

Example Code 13.3 *Specify a Single CMP Library:*

```
cas casauto sessopts=(cmplib="casuser.mycmpfuncs");
```

Example Code 13.4 *Specify Multiple CMP Libraries:*

```
cas casauto sessopts=(cmplib="ourcmplibs.cmpfuncs casuser.mycmpfuncs");
```

Example 3: Get the Current Value of Session Option CMPLIB

This example writes the current value of session option CMPLIB to the SAS log:

```
%put Session Casauto: CMPLIB=%sysfunc(getsessopt(casauto, cmplib));
```

The GETSESSOPT arguments are not enclosed in quotation marks when it is invoked by %SYSFUNC. Here is an example of the result.

```
Session Casauto: CMPLIB=ourcmplibs.cmpfuncs casuser.mycmpfuncs
```

See Also

- Actions setSessOpt and getSessOpt in "Session Properties Action Set" in *SAS Viya Platform: System Programming Guide*
- "FCMP Action Set" in *SAS Viya Platform: System Programming Guide*

CMPOPT= Session Option

specifies the type of code generation optimizations to use in the SAS language compiler.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: System Administration: Performance

Default: ALL

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

CMPOPT=*"optimization-value <optimization-value <...>>"* | ALL | NONE

Parameter Values

optimization-value <optimization-value <...>>
specifies the type of optimization that the SAS compiler is to use. Specify one or more of the following as a space-delimited list enclosed in quotation marks:

CONSTANTFOLDING
NOCONSTANTFOLDING
specifies to evaluate numeric and character literal expressions at compile time for supported operations. When you specify NOCONSTANTFOLDING, numeric and character literal expressions are evaluated at runtime.

EXTRAMATH
NOEXTRAMATH
specifies whether the compiler is to retain or remove the extra mathematical operations that do not affect the outcome of a statement.

Here are some of the mathematical operations that are either included or excluded in the machine language code.

$x * 1$	$x * -1$
$x \div 1$	$x \div -1$
$x + 0$	x

x - *x* *x* ÷ *x*

- -*x* any operation on two literal constants

Specify EXTRAMATH to retain the extra mathematical operations.

Default NOEXTRAMATH

FUNCDIFFERENCING
NOFUNCDIFFERENCING

specifies whether numeric-differencing derivatives or analytic derivatives are calculated for user-defined functions.

Specify FUNCDIFFERENCING to calculate numeric-differencing derivatives for user-defined functions. Specify NOFUNCDIFFERENCING to calculate analytic derivatives for user-defined functions.

Default NOFUNCDIFFERENCING

GUARDCHECK
NOGUARDCHECK

specifies whether the compiler checks for array boundary problems.

Specify GUARDCHECK to check for array boundary problems.

Default NOGUARDCHECK

Interaction NOGUARDCHECK is set when CMPOPT is set to ALL or NONE.

MISSCHECK
NOMISSCHECK

specifies whether to check for missing values in the data.

Specify MISSCHECK to check for missing data.

Default NOMISSCHECK

Tip If the data contains a significant amount of missing data, specify MISSCHECK to optimize the compilation. Otherwise, specify NOMISSCHECK.

PRECISE
NOPRECISE

specifies whether exceptions are handled at an operation boundary or at a statement boundary.

Specify PRECISE to handle exceptions at the operation boundary. Specify NOPRECISE to handle exceptions at the statement boundary.

Default NOPRECISE

SHORTCIRCUIT
NOSHORTCIRCUIT

specifies whether to optimize the evaluation of expressions that use multiple logical operators.

Note NOSHORTCIRCUIT is set when an expression includes derivatives and more than one logical tree or ternary operations.

Note If you specify a value more than once, then the last value that you specified is used. For example, if you specify values PRECISE, NOEXTRAMATH, and NOPRECISE, in that order, NOPRECISE is used.

Tips CONSTANTFOLDING EXTRAMATH, MISSCHECK, PRECISE, GUARDCHECK, FUNCDIFFERENCING, and SHORTCIRCUIT can be specified in any combination.

All leading, trailing, and embedded blanks are removed.

Example Specify EXTRAMATH, MISSCHECK, and PRECISE:

```
cas casauto sessopts=(cmpopt="extramath misscheck precise");
```

ALL
specifies that the compiler is to optimize the machine language code by using the CONSTANTFOLDING NOEXTRAMATH, NOMISSCHECK, NOPRECISE, NOGUARDCHECK, NOFUNCDIFFERENCING, and SHORTCIRCUIT optimization values.

Restriction ALL cannot be specified with other values.

NONE
specifies that the compiler is not set to optimize the machine language code by using the NOCONSTANTFOLDING EXTRAMATH, MISSCHECK, PRECISE, NOGUARDCHECK, FUNCDIFFERENCING, and NOSHORTCIRCUIT optimization values.

Restriction NONE cannot be specified with other values.

See Also

"CMPOPT= System Option" in *SAS System Options: Reference*

COLLATE= Session Option

specifies the collating sequence for sorting.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Sort

Default: UCA

Syntax

COLLATE=MVA | UCA

Parameter Values

MVA
specifies SAS client collating.

UCA
specifies a locale-appropriate collating sequence.

DATASTEPFMTERR= Session Option

specifies how SAS reacts when a specified variable format cannot be found.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: DATA Step

Alias: FMTERR

Default: TRUE

Restrictions: For DATA steps, this option applies only when the DATA step runs in SAS Cloud Analytic Services. See "DATA Step Processing in CAS" in *SAS Cloud Analytic Services: DATA Step Programming*.

The FMTERR alias is valid in the CAS procedure and in the GETSESSOPT function only. In the CAS statement and in system option SESSOPT, specify DATASTEPFMTERR.

See: *SAS Cloud Analytic Services: User-Defined Formats*

SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

DATASTEPFMTERR=TRUE | FALSE

Parameter Values

TRUE
when SAS cannot find a specified variable format, it writes an error message to the SAS log, and then stops processing. This is the default.

FALSE
when SAS cannot find a specified variable format, it writes a note to the SAS log, substitutes the $w. or BEST12. format for the missing format, and then proceeds with processing.

DATASTEPMERGENOBY= Session Option

specifies whether a warning message is generated, an error message is generated, or no message is generated when a DATA step merge is executed without a BY statement.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: DATA Step

Default: NOWARN

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

DATASTEPMERGENOBY='ERROR' | 'NOWARN' | 'WARN'

Parameter Values

'ERROR'
specifies that the merge does not run and an error message is generated when there is no BY statement.

'NOWARN'
specifies that the merge runs with no warning messages when there is no BY statement. This is the default.

'WARN'
specifies that the merge runs with warning messages when there is no BY statement.

DATASTEPMSGSUMLEVEL= Session Option

specifies the DATA step message summary level.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: DATA Step

Default: ALL

Tip: When the DATA step runs on multiple threads, the same message can be generated on each thread. In that case, use this option to control the summary level of the duplicate messages to help reduce the client log output.

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

DATASTEPMSGSUMLEVEL=ALL | PUT | NONE

Parameter Values

In a DATA step, messages are received from each thread, which can result in a large number of duplicate messages when multiple threads are used. The first occurrence of all messages, including PUT statement messages, are sent to the client when they occur. By default, duplicate messages are summarized, and then sent to the client to reduce client log output. Specify one of the following values to control the level of summarization for duplicate messages:

ALL
summarizes all duplicate messages, including PUT statement messages, and sends them to the client when the DATA step exits. This is the default.

PUT
summarizes all duplicate messages, except PUT statement messages, and sends them to the client when the DATA step exits. All PUT statement messages are not summarized and are sent to the client as they occur.

NONE
does not summarize duplicate messages. All messages, including PUT statement messages, are sent to the client as they occur.

DATASTEPREPLACETABLE= Session Option

specifies whether a DATA step can replace an existing table.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: DATA Step

Default: TRUE

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

DATASTEPREPLACETABLE=TRUE | FALSE

DATASTEPVARLENCHK= Session Option

specifies whether a warning message is generated, an error message is generated, or no message is generated when truncation occurs during a DATA step append.

Valid in: CAS statement SESSOPTS option, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: DATA Step

Default: NOWARN

Restriction: This option applies only when the DATA step is running in SAS Cloud Analytic Services.

Syntax

DATASREPVARLENCHK=NOWARN | WARN | ERROR

Parameter Values

NOWARN
specifies that no check is made for truncation during a DATA step append. This is the default. During a DATA step append, if a string in the source data set is longer than the corresponding string in the destination data set, truncation occurs, and no messages are written to the SAS log.

WARN
specifies that a warning message is written to the SAS log and DATA step execution continues if truncation is detected during a DATA step append.

ERROR
specifies that an error message is written to the SAS log and DATA step execution is terminated if truncation is detected during a DATA step append.

Examples

Example 1: Set Session Option DATASTEPVARLENCHK for Warnings

This example specifies that a warning message is to be written to the SAS log and execution is to continue if truncation occurs during the execution of a DATA step append in session Casauto:

```
cas casauto sessopts=(datastepvarlenchk=warn);
```

Example 2: Set Session Option DATASTEPVARLENCHK for Errors

This example specifies that an error message is to be written to the SAS log and execution is to stop if truncation occurs during the execution of a DATA step append in session Casauto:

```
cas casauto sessopts=(datastepvarlenchk=error);
```

Example 3: Get the Current Value of Session Option DATASTEPVARLENCHK

This example writes the current value of session option DATASTEPVARLENCHK to the SAS log:

```
%put Session Casauto: DATASTEPVARLENCHK=%sysfunc(getsessopt(casauto,
datastepvarlenchk));
```

The GETSESSOPT arguments are not enclosed in quotation marks when it is invoked by %SYSFUNC. Here is an example of the result.

```
Session Casauto: DATASTEPVARLENCHK=ERROR
```

DATESTYLE= Session Option

specifies the sequence of month, day, and year when ANYDTDTE, ANYDTDTM, or ANYDTTME informat data is ambiguous.

Valid in: CAS statement SESSOPTS option, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Formats

Default: The value of SAS system option DATESTYLE=, which is LOCALE by default.

Interaction: This option overrides SAS system option DATESTYLE= for a session.

Syntax

DATESTYLE=MDY | YMD | DMY | LOCALE

Parameter Values

MDY
specifies that SAS set the order as month, day, year.

YMD
specifies that SAS set the order as year, month, day.

DMY
specifies that SAS set the order as day, month, year.

LOCALE
specifies that SAS set the order based on the value that corresponds to the LOCALE= system option value, which can be MDY, YMD, or DMY.

Examples

Example 1: Set Session Option DATESTYLE

This example sets session option DATESTYLE to DMY for session Casauto:

```
cas casauto sessopts=(datestyle=dmy);
```

Example 2: Get the Current Value of Session Option DATESTYLE

This example writes the current value of session option DATESTYLE= for session Casauto to the SAS log:

```
%put Session Casauto: DATESTYLE=
%sysfunc(getsessopt(casauto,datestyle)).;
```

The GETSESSOPT arguments are not enclosed in quotation marks when it is invoked by %SYSFUNC. The default value of session option DATESTYLE= is determined by the value of SAS system option DATESTYLE= at the time the session is created. Here is an example of the result.

```
Session Casauto: DATESTYLE=DMY
```

See Also

Informats:

- "ANYDTDTEw. Informat" in *SAS Formats and Informats: Reference*
- "ANYDTDTMw. Informat" in *SAS Formats and Informats: Reference*

- "ANYDTTMEw. Informat" in *SAS Formats and Informats: Reference*

System options:

- "LOCALE System Option" in *SAS National Language Support (NLS): Reference Guide*
- "DATESTYLE= System Option" in *SAS System Options: Reference*

DEBUGOPT= Session Option

specifies the host and port for the SAS Code Debugger in a session.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Debugging

Default: " "

Notes: This option is not initialized with the value of SAS system option DEBUGOPT. You must set session option DEBUGOPT separately.

This session option might have already been set by your CAS administrator.

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

DEBUGOPT="DEBUGHOST=*host-name* DEBUGPORT=*port*"

Debug Options

DEBUGHOST=*host-name*
specifies the debugger host by name.

Requirement: Specify DEBUGHOST= and DEBUGPORT= separated by a space and enclosed in single or double quotation marks.

DEBUGPORT=*port*
specifies the debugger port.

Examples

Example 1: Set Session Option DEBUGOPT

This example specifies the debugger host name and port:

```
cas casauto sessopts=(debugopt="debughost=cloud.example.com
debugport=5590");
```

Example 2: Synchronize Session Option DEBUGOPT with SAS System Option DEBUGOPT

This example sets session option DEBUGOPT in session Casauto to the current value of SAS system option DEBUGOPT:

```
cas casauto sessopts=(debugopt="%sysfunc(getoption(debugopt))");
```

The GETOPTION argument is not enclosed in quotation marks when it is invoked by %SYSFUNC.

Example 3: Get the Current Value of Session Option DEBUGOPT

This example lists the current DEBUGOPT= setting for session Casauto:

```
%put Session Casauto: DEBUGOPT=%sysfunc(getsessopt(casauto,debugopt));
```

The GETSESSOPT arguments are not enclosed in quotation marks when it is invoked by %SYSFUNC. Here is an example of the result.

```
Session Casauto: DEBUGOPT=DEBUGHOST=CLOUD.EXAMPLE.COM DEBUGPORT=5590
```

Example 4: Clear the Value of Session Option DEBUGOPT

This example clears the debug options to revert to the default:

```
cas casauto sessopts=(debugopt="");
```

See Also

- "DEBUGOPT= System Option" in *SAS System Options: Reference*
- SAS Code Debugger and SAS DATA Step Debugger: User's Guide

DEFAULTMEMORYFORMAT= Session Option

specifies the default memory format for session tables.

Valid in: CAS statement SESSOPTS option, CAS configuration file, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Memory

Default: In order of descending precedence:

1. CAS server option cas.DefaultMemoryFormat=, if set
2. Environment variable CAS_DEFAULT_MEMORY_FORMAT=, if set

3. STANDARD

Notes: Support for this option was added in 2025.11.

This option overrides CAS server option cas.DefaultMemoryFormat= for a session.

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

DEFAULTMEMORYFORMAT=DVR | STANDARD

Parameter Values

DVR
specifies that the duplicate value reduction memory format is used. The DVR memory format can reduce memory consumption and the on-disk size of SASHDAT files when the input data contains a large number of duplicate values.

STANDARD
specifies that the standard memory format is used. The standard memory format is designed for performance. It is the default for this option.

Examples

Example 1: Get the Current DEFAULTMEMORYFORMAT= Setting

This example shows how to use the GETSESSOPT function to get the current DEFAULTMEMORYFORMAT= session option setting for session Casauto:

```
cas casauto sessopts=(caslib=casuser);
%put NOTE: Session CASAUTO: DEFAULTMEMORYFORMAT=
%sysfunc(getsessopt(casauto,
   defaultmemoryformat)).;
```

```
NOTE: Session CASAUTO: DEFAULTMEMORYFORMAT=STANDARD.
```

Example 2: Enable DVR Memory Formatting in a Session

This example shows how to enable DVR memory formatting for session Casauto:

```
cas casauto sessopts=(defaultmemoryformat=dvr);
%put NOTE: Session CASAUTO: DEFAULTMEMORYFORMAT=
%sysfunc(getsessopt(casauto,
   defaultmemoryformat)).;
```

```
NOTE: Session CASAUTO: DEFAULTMEMORYFORMAT=DVR.
```

See Also

"Fine Tuning: Memory Format" in *SAS Viya Platform: System Programming Guide*

DEFAULTTABLEREPLICATION= Session Option

specifies the default replication factor for new tables for a session.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Table Replication

Default: 1

Range: 0–*number_of_workers*

Notes: A value of zero disables default replication for new tables for the session.

You can specify a value that is greater than *number-of-workers*. In that case, *number-of-workers* is used instead.

You can override this session option value by specifying replication= in the casOut parameter in statements that support the casOut parameter.

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

DEFAULTTABLEREPLICATION=*integer*

Examples

Example 1: Set Session Option DEFAULTTABLEREPLICATION

This example sets the default replication factor for new tables to 5 for session Casauto:

```
cas casauto sessopts=(defaulttablereplication=5);
```

Example 2: Disable the Default Replication of New Tables

This example disables the default replication of new tables for session Casauto:

```
cas casauto sessopts=(defaulttablereplication=0);
```

Example 3: Get the Current Value of Session Option DEFAULTTABLEREPLICATION

This example prints the current value of session option DEFAULTTABLEREPLICATION= for session Casauto:

```
%put Session Casauto: DEFAULTTABLEREPLICATION=
%sysfunc(getsessopt(casauto,
   defaulttablereplication));
```

The GETSESSOPT arguments are not enclosed in quotation marks when it is invoked by %SYSFUNC. Here is an example of the result.

```
Session Casauto: DEFAULTTABLEREPLICATION=0.
```

DQLOCALE= Session Option

specifies the default 5-letter SAS Quality Knowledge Base (QKB) ISO locale code to use as the default locale for data quality (DQ) operations.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Data Quality

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

DQLOCALE=*"5-letter-locale-code"*

Details

For a list of the 5-letter QKB ISO locale codes, see QKB Locale ISO Codes.

DQQKBREPOLOC= Session Option

specifies the name of the caslib where SAS Quality Knowledge Bases are stored.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Data Quality

Default: ReferenceData

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

DQQKBREPOLOC=*"caslib-name"*

DQSETUPLOC= Session Option

specifies the name of the default SAS Quality Knowledge Base (QKB) to use for data quality operations.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Data Quality

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

DQSETUPLOC=*"QKB-name"*

HISTORYLINELENGTH= Session Option

specifies the number of bytes to use for reporting action names and action parameters in a line of history.

Valid in: CAS statement SESSOPTS option, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Action History

Defaults: The value of server environment variable CASACTIONHISTORYLEN, if set.
8192 if environment variable CASACTIONHISTORYLEN is not set.

Range: 8192–8388608 (8 megabytes)

Notes: This option takes precedence over server environment variable CASACTIONHISTORYLEN for a session.

If an action request exceeds the history line length, the recorded history line is truncated.

See: CAS statement LISTHISTORY option on page 709

History Action

SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about server environment variable CASACTIONHISTORYLEN.

Syntax

HISTORYLINELENGTH=*number*

Examples

Example 1: Set Session Option HISTORYLINELENGTH for Session Casauto

This example sets the value of session option HISTORYLINELENGTH to 12000 for session Casauto:

```
cas casauto sessopts=(historylinelength=12000);
```

Example 2: Get the Current Value of Session Option HISTORYLINELENGTH

This example writes the current value of session option HISTORYLINELENGTH for session Casauto to the SAS log:

```
%put Session Casauto: HISTORYLINELENGTH=
%sysfunc(getsessopt(casauto,historylinelength));
```

The GETSESSOPT arguments are not enclosed in quotation marks when it is invoked by %SYSFUNC. Here is an example of the result.

```
Session Casauto: HISTORYLINELENGTH=12000
```

Example 3: Set Session Option HISTORYLINELENGTH for the Active Session

This example sets the value of session option HISTORYLINELENGTH to 15000 for the currently active session:

```
options cassessopts=(historylinelength=15000);
```

INTERVALDS= Session Option

specifies one or more interval-name=value pairs, where the value is the name of a data set that contains user-defined intervals.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Categories: Input Control
Data Processing

Interaction: This option overrides system option INTERVALDS for your session.

Tip: To specify interval data sets for all of your sessions, use the INTERVALDS system option instead.

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

INTERVALDS=*"interval-1=CAS-libref.data-table-1 <interval-2=CAS-libref.data-table-2 ...>"*

Parameter Value

Interval specifies the name of an interval. The interval name is used to represent the set of intervals that is specified in data table *CAS-libref.data-table*. *CAS-libref.data-table* specifies a two-level name, where CAS-libref is a CAS engine libref that is associated with the CAS session and data-table is the name of the data table that contains the user-defined intervals.

See Also

"INTERVALDS= System Option" in *SAS System Options: Reference*

LOCALE= Session Option

specifies the locale to use for sorting and formatting.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Localization

Default: en_US

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

LOCALE=*"locale"*

See Also

SAS National Language Support (NLS): Reference Guide

LOGFLUSHTIME= Session Option

specifies the log flush time, in milliseconds.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Log

Default: 100

Range: –1–86400

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

LOGFLUSHTIME=–1 | 0 | *number*

Parameter Values

–1
flushes logs after each action completes.

0
flush logs as they are produced.

number
flushes logs in *number* milliseconds.

MAXCORESPERWORKER= Session Option

specifies the maximum number of cores per worker for the session.

Valid in: CAS statement SESSOPTS option, CAS configuration file, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Session

Default: The value specified for CAS server configuration option MAXCORESPERWORKER, which defaults to 0.

Range: 0–20000

Note: Support for this option was added in 2025.03.

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

MAXCORESPERWORKER=*integer*

Required Argument

integer
specifies the maximum number of cores per worker.

Range 0–20000

Note A value of 0 specifies the number of cores that are available to each worker.

Details

The MAXCORESPERWORKER= enables you to control threading for the session during action execution. When not specified, this option defaults to the value of the CAS server configuration option cas.MAXCORESPERWORKER, which defaults to 0. When MAXCORESPERWORKER=0, the system honors the CAS server container's CGroup core limit in order to reduce performance degradation due to overthreading. When MAXCORESPERWORKER= is set to a nonzero value, the number of cores is limited to that value, up to the container's CGroup core limit.

Reducing the maximum number of cores per worker can help reduce core contention and improve performance. This is especially true when multiple users are running highly concurrent actions on the CAS server. You can use CAS server configuration option cas.MAXCORESPERWORKER to change the default maximum

for all sessions on your CAS server. You can use session option MAXCORESPERWORKER= to change the maximum cores per worker for a single session in order to improve performance for that session only. For a simple example of how to set the MAXCOREWORKER= option for a single session, see "Example 3: Measure Elapsed Time over Varying Values of MAXCORESPERWORKER=" on page 676.

Examples

Example 1: Set Session Option MAXCORESPERWORKER

This example sets MAXCORESPERWORKER to 1 for session Casauto:

```
cas casauto sessopts=(maxcoresperworker=1);
```

Example 2: Get the Current Value of Session Option MAXCORESPERWORKER

This example gets the current value of session option MAXCORESPERWORKER= for session Casauto:

```
%put Session Casauto: MAXCORESPERWORKER=
%sysfunc(getsessopt(casauto,maxcoresperworker));
```

The GETSESSOPT arguments are not enclosed in quotation marks when it is invoked by %SYSFUNC. Here is an example of the result.

```
Session Casauto: MAXCORESPERWORKER=1
```

Example 3: Measure Elapsed Time over Varying Values of MAXCORESPERWORKER=

You might be able to adjust MAXCORESPERWORKER= in order to improve the performance of certain jobs in your session. For example, you might try running an action with different thread counts to see whether changing the default can improve the performance of that action on your system. Here is a simple example that uses the simple.summary action. This example is for demonstration purposes only.

```
/* options cashost="cloud.example.com" casport=5570; */
cas casauto sessopts=(caslib=casuser,                    /* 1 */
   maxcoresperworker=0);
libname casuser cas caslib=casuser;

data casuser.testdata;                                   /* 2 */
   do x=1 to 1000;
      y=rand("uniform", 1, 500);
      output;
   end;
run;

proc cas;                                                /* 3 */
```

```
    session casauto;
    def = .;
    sessionProp.setSessOpt /                                    /* 4 */
          maxcoresperworker = 0;
    simple.summary result = r /
       inputs = {'x', 'y'},
       subset = 'mean', table = 'testdata';
    def = _perf.elapsed_time * 1000;                            /* 5 */
    timings = newtable("Timings",                               /* 6 */
       {"MAXCORESPERWORKER", "Elapsed_Time",
       "Change"}, {"int64", "double", "double"},
       {0, def, .});
    tcounts = {1, 5, 10, 20, 30};                               /* 7 */
    do tcount over tcounts;                                     /* 8 */
       sessionProp.setSessOpt /
          maxcoresperworker = tcount;
       simple.summary result = r /
          inputs = {'x', 'y'},
          subset = 'mean', table = 'testdata';
       delta = (((_perf.elapsed_time * 1000 - def) / def));
       timings = addrow(timings,
          {tcount,_perf.elapsed_time * 1000,
           delta});
      end;
    end;
    saveresult Timings dataout = work.Timings replace;
run;
quit;

cas casauto terminate;                                          /* 9 */

proc print data = Timings label noobs;                          /* 10 */
   label Elapsed_Time = "Elapsed Time (mS)"
      Change = "Change from Default";
   format Elapsed_Time 8.2 change percentn8.2;
run;

proc sql noprint;                                               /* 11 */
    select min(Elapsed_Time)
    into :BarMin from work.Timings;
quit;

%let BarMin=%sysevalf(&BarMin * 0.95);

ods graphics / width = 460px noscale;
title "Run Time Verses Maximum Number of Cores/Worker";
footnote "The default is 0.";
proc sgplot data = Timings;
   vbar MAXCORESPERWORKER / response = Elapsed_Time
      datalabel = Elapsed_Time;
   xaxistable Change / location = inside position = top
      nomissingchar valueattrs = (size=9pt) separator;
   yaxis min=&BarMin;
   label Elapsed_Time = "Elapsed Time (mS)"
      MAXCORESPERWORKER = "MAXCORESPERWORKER=n";
   format Elapsed_Time 8.1 change percentn8.1;
```

```
run;
title;
footnote;
```

1 The CAS statement creates session Casauto and sets session option MAXCORESPERWORKER= to 0. The LIBNAME statement creates CAS libref Casuser.

2 The DATA step creates table testdata, which provides data for this simple test.

3 The PROC CAS statement starts the CAS procedure step.

4 The sessionProp.setSessOpt action statement sets session option MAXCORESPERWORKER=0 (default) for the baseline test. The simple.summary statement runs the action under test.

5 The ellapsed time is multiplied by 1000 to convert it to milliseconds and is stored in variable Def. This value is used as the baseline for comparison in the rest of this program.

6 The baseline value is stored in new table Timings.

7 Variable tcounts stores a list of thread counts to use for MAXCORESPERWORKER= in each test iteration.

8 The DO statement iterates over each value of tcounts. For each iteration, it sets session option MAXCORESPERWORKER= to the tcount variable value, and then runs action simple.summary on the test data. After the execution is complete, it then retrieves the execution elapsed time, multiplies it by 1000 to convert from seconds to milliseconds, and stores the elapsed time for that iteration in table Timings. After all of the timing data is collected, table Timings is saved to data set Work.Timings.

9 The CAS statement terminates session Casauto.

10 The PRINT procedure step prints data set Timings for comparison.

11 Optional: The remaining statements generate a bar chart of the data in data set Timings for visualization. The SQL procedure step gets the minimum value of column Elapsed_Time in data set Timings and assigns it to macro variable BarMin. Because the timing data range is fairly small, this value is used later to adjust the bar chart response axis data range in order to provide a better view of the timing changes. The BarMin value is decreased by 5 percent to ensure that the minimum bar appears in the bar chart. The ODS GRAPHICS statement specifies the graph size and disables scaling in order to preserve the font sizes. The TITLE and FOOTNOTE statements add a title and footnote. The SGPLOT procedure step generates the bar chart.

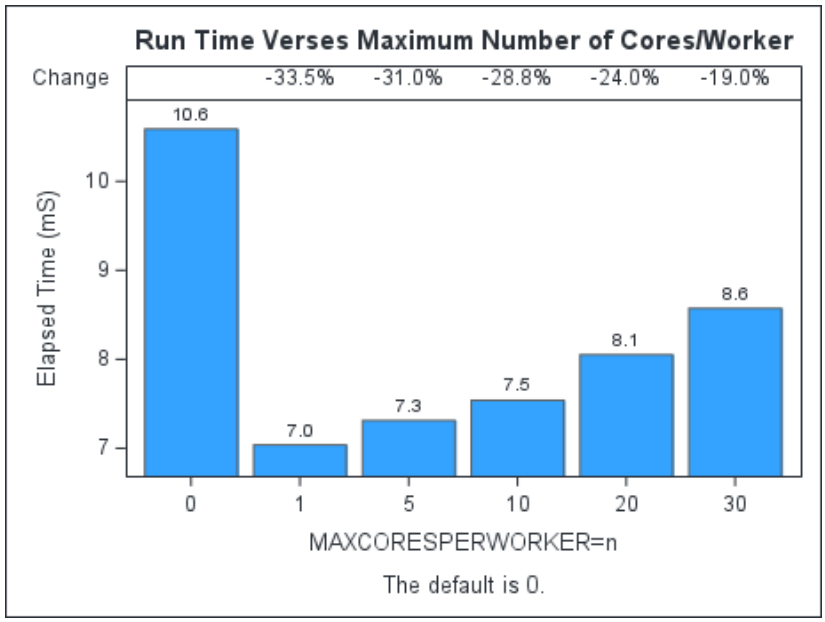
    For more information about ODS Graphics, see *SAS ODS Graphics: Procedures Guide*.

Here are the results.

Output 13.1 *PROC PRINT Result*

MAXCORESPERWORKER	Elapsed Time (mS)	Change from Default
0	10.59	.
1	7.04	-33.54%
5	7.31	-30.95%
10	7.54	-28.80%
20	8.05	-23.96%
30	8.57	-19.04%

Output 13.2 *PROC SGPLOT Result*



The results shown are an example. The actual results can vary significantly, depending on the number of cores that are available and the number of concurrent jobs that are running on your CAS server when the program is executed.

MAXTABLEMEM= Session Option

specifies the maximum amount of memory, in bytes, to allocate for a table.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Caslib

Default: 16M

Note: After this threshold is reached, the server uses temporary files and operating system facilities for memory management.

Tips: You can enclose the value in quotation marks and specify B, K, M, G, or T as a suffix to indicate the units. For example, "8M" specifies eight megabytes.

If you are creating a large table of 5G or greater per node, to improve performance, set MAXTABLEMEM in your active session to 256M or higher.

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

MAXTABLEMEM=*integer* | "*integer*<*units*>"

Required Argument

integer
specifies the total number of bytes to allocate.

Note A value of 0 or less specifies that there is no limit. See "Details" on page 680.

Optional Argument

units
specifies the value units, which can be one of the following:

B bytes

K kilobytes

M megabytes

G gigabytes

T terabytes

Default B (bytes)

Requirement If you specify units, you must enclose the value in quotation marks.

Details

When data is added to the server (loading a table, appending rows, and so on) the server organizes the data in-memory blocks and allocates memory for the blocks. The MAXTABLEMEM= value is used to specify the amount of memory that each thread should allocate before converting to a memory-mapped file. Files are written in the directories that are specified in the CAS_DISK_CACHE environment

variable. See SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services*.

If a value of 0 or less is specified, no limit is imposed. In that case, the data is not converted to a memory-mapped file until backup copies are generated or the table is promoted. Removing the limit can be useful when creating large short-lived output tables. See "Short-lived Output Tables" in *SAS Cloud Analytic Services: Fundamentals*.

Examples

Example 1: Set Session Option MAXTABLEMEM to 256 Megabytes

This example sets MAXTABLEMEM to 256 megabytes for session Casauto:

```
cas casauto sessopts=(maxtablemem="256M");
```

Example 2: Set Session Option MAXTABLEMEM to 32 Megabytes

This example sets MAXTABLEMEM to 32 megabytes for session Casauto:

```
cas casauto sessopts=(maxtablemem=33554432);
```

Example 3: Get the Current Value of Session Option MAXTABLEMEM

This example displays the current value of session option MAXTABLEMEM for session Casauto:

```
%put Session Casauto: MAXTABLEMEM=
%sysfunc(getsessopt(casauto,maxtablemem));
```

The GETSESSOPT arguments are not enclosed in quotation marks when it is invoked by %SYSFUNC. Here is an example of the result.

```
Session Casauto: MAXTABLEMEM=33554432
```

See Also

"Memory" in *SAS Cloud Analytic Services: Fundamentals*

MESSAGELEVEL= Session Option

specifies the log message level.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Log

Default: ALL

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

MESSAGELEVEL=ALL | DEFAULT | ERROR | NONE | NOTE | WARNING

METRICS= Session Option

specifies whether to include default detailed performance metrics reports in the SAS log.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Log

Default: FALSE

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Example: Enable metrics for session Casauto:

```
cas casauto sessopts=(metrics=true);
```

Syntax

METRICS=TRUE | FALSE

Details

Session option METRICS= enables you to display information about the resources that your session consumes as each action in your program is executed. You can use the metrics information to track the resources that your session consumes and make adjustments, if necessary. By default, metrics are disabled for your session. Specify session option METRICS=TRUE to enable default metrics for your session. When enabled, after each action is executed, the metrics are written to the SAS log

as notes. One note is written for each available metric. The following table lists the default metrics that are written to the SAS log when metrics are enabled.

Table 13.1 *Default Metrics*

Metric	Note Written to the SAS Log	
The number of bytes moved.	`NOTE: bytes moved`	*number<units>*
The CPU time in seconds and as a percentage of cluster utilization. Cluster utilization is the sum of the utilization for each core in the cluster and can exceed 100%. For example, cluster utilization for a 96-core cluster where each core is 100% used is 9600%.	`NOTE: cpu time`	*number* `seconds (`*number*`%)`
The data movement time in seconds.	`NOTE: data movement time`	*number* `seconds`
The amount of memory used in bytes and as a percentage of the total available memory.	`NOTE: memory`	*number<units>* `(`*number*`%)`
The amount of time it took to run the action start to finish (real time) in seconds.	`NOTE: real time`	*number* `seconds`
The total available memory.	`NOTE: total memory`	*number<units>*
The total number of nodes and cores in the cluster.	`NOTE: total nodes`	*number* `(`*number* `cores)`

Here is an example of the default metrics that are written to the SAS log when the MDSUMMARY procedure is executed.

```
NOTE: real time               0.126269 seconds
NOTE: cpu time                1.036837 seconds (821.13%)
NOTE: data movement time      0.013819 seconds
NOTE: total nodes             27 (1296 cores)
NOTE: total memory            6.65T
NOTE: memory                  57.25M (0.00%)
NOTE: bytes moved             4.80K
NOTE: The SAS Cloud Analytic Server processed the request in 0.126269 seconds.
NOTE: The data set MYCAS.MPGHW_SUM has 15 observations and 19 variables.
NOTE: PROCEDURE MDSUMMARY used (Total process time):
      real time           0.38 seconds
      cpu time            0.01 seconds
```

NLDECSEPARATOR= Session Option

specifies whether SAS uses locale-sensitive decimal separators or US English decimal separators (dot) in formatted numeric output.

Valid in: CAS statement SESSOPTS option, CAS configuration file, GETSESSOPT function, sessionProp.getSessOpt action name parameter

Category: Formats

Default: TRUE

Restriction: This option applies to only the ISO 8601 family of formats.

Interaction: This option overrides SAS system option NLDECSEPARATOR for a session.

Note: This option is valid starting in 2021.2.6.

See: "NLDECSEPARATOR System Option" in *SAS National Language Support (NLS): Reference Guide*

SAS Formats and Informats: Reference and *SAS National Language Support (NLS): Reference Guide* for information about ISO 8601 formats.

SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

NLDECSEPARATOR=TRUE | FALSE

Parameter Values

TRUE
enables locale-sensitive numeric output for the decimal separator.

FALSE
disables locale-sensitive numeric output for the decimal separator.

Details

The NLDECSEPARATOR= session option enables you to specify whether locale-sensitive decimal separators are used when formatting date, time, and datetime values with an ISO 8601 format. It applies only when using a format in the ISO 8601 family of formats. When TRUE (default), locale-sensitive decimal separators are used in formatted numeric values. When FALSE, the US English locale decimal separator (dot) is used regardless of locale.

Some non-US locales use the comma as the decimal separator instead of the dot when formatting numbers. The English locale time value 15:30:55.665, for example,

might be formatted as 15:30:55,665 in some locales. In that situation, you can use the NLDECSEPARATOR=FALSE session option to force an ISO 8601 format to use the US English locale character (dot) instead of the locale-sensitive character. See Example 3 on page 685.

For information about using ISO 8601 formats, see *SAS Formats and Informats: Reference* and *SAS National Language Support (NLS): Reference Guide*.

Examples

Example 1: Set Session Option NLDECSEPARATOR

This example sets session option NLDECSEPARATOR to FALSE for session Casauto:

```
cas casauto sessopts=(nldecseparator=false);
```

Example 2: Get the Current Value of Session Option NLDECSEPARATOR

This example writes the current value of session option NLDECSEPARATOR for session Casauto to the SAS log:

```
%put Session Casauto: NLDECSEPARATOR=
%sysfunc(getsessopt(casauto,NLDECSEPARATOR));
```

The GETSESSOPT arguments are not enclosed in quotation marks when it is invoked by %SYSFUNC. Here is the result. Here is an example of the result.

```
Session Casauto: NLDECSEPARATOR=0
```

Example 3: The Effect of Session Option NLDECEPARATOR on Format E8601TM in the Norwegian Locale

This example showcases the difference in behavior with the E8601TM format when run with the session option set to TRUE and then set to FALSE in the Norwegian locale. Here is the example for TRUE.

```
cas casauto sessopts=(locale=nn_NO, nldecseparator=true,
   caslib=casuser);
options nonotes nosource;
%put LOCALE = %sysfunc(getsessopt(casauto,locale));
%put NLDECSEPARATOR = %sysfunc(getsessopt(casauto,nldecseparator));

/* NLDECSEPARATOR session option is TRUE by default */
libname casuser cas caslib=casuser;
data casuser.abc;
x = put('15:30:55.665't, e8601tm15.3);
put x=;
run;
options notes source;
```

Here is the result.

```
LOCALE = nn_NO
NLDECSEPARATOR = 1
x=15:30:55,665
```

Because NLDECSEPARATOR=TRUE, the Norwegian locale decimal separator (comma) is used in the formatted output. Here is the example for FALSE.

```
/* Setting NLDECSEPARATOR session option to FALSE */
cas casauto sessopts=(NLDECSEPARATOR=false, caslib=casuser);
options nonotes nosource;
%put LOCALE = %sysfunc(getsessopt(casauto,locale));
%put NLDECSEPARATOR = %sysfunc(getsessopt(casauto,nldecseparator));

libname casuser cas caslib=casuser;
data casuser.abc;
x = put('15:30:55.665't, e8601tm15.3);
put x=;
run;
options notes source;
```

Here is the result.

```
LOCALE = nn_NO
NLDECSEPARATOR = 0
x=15:30:55.665
```

Because NLDECSEPARATOR=FALSE, the US English decimal separator (dot) is used in the formatted output.

NWORKERS= Session Option

specifies the number of worker nodes for a new session.

Valid in: CAS statement SESSOPTS option, CAS configuration file, GETSESSOPT function, sessionProp.getSessOpt action name parameter

Category: Session

Default: In order of descending precedence:

1. SAS system option CASNWORKERS, if you explicitly set it in SAS
2. 0 (all)

Range: 0–5000

Restriction: The number of workers can be set for new sessions only.

Tip: Use this option to start a subset session. See "Example 2: Start a Subset Session" on page 725.

See: "CASNWORKERS= System Option" in *SAS System Options: Reference*

SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

NWORKERS=*number*

S2FORMATSEARCH= Session Option

specifies the format search list for format libraries that have been published to SingleStore for use by the SAS Embedded Process.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Formats

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Example: "Example 2: Publish User-Defined Formats to a SingleStore Database" on page 329

Syntax

S2FORMATSEARCH= *"caslib-1.tablename-1 <... caslib-n.tablename-n>"*

Parameter Value

caslib.tablename
specifies a caslib and table in a SingleStore database. The table contains a previously published format library. The caslib is required for each table. The caslib and table name cannot contain blank spaces. When specifying multiple tables, separate each *caslib.tablename* with a space.

Details

This option is valid for the SAS SpeedyStore data connector. In those use cases, WHERE expressions and computed columns are processed in the SingleStore database using the SAS Embedded Process.

If a WHERE expression or computed column includes a user-defined format, you must use the sessionProp.saveFmtLib action to save the format library as a SingleStore database table. Then you must set the S2FORMATSEARCH= session option or cas.S2FORMATSEARCH= server option to list one or more database table names that contain the formats.

If you specify multiple tables, separate each table name with a space.

SESSIONTABLEREDISTUPPOLICY= Session Option

specifies a session table redistribution policy that is applied when the number of worker pods is scaled up on a running distributed CAS server (MPP).

Valid in: CAS statement SESSOPTS option, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Session

Default: REDIST_UNSPECIFIED

Tips: You can override this policy for a specific caslib by using the CASLIB statement TABLEREDISTUPPOLICY= option or the table.addCaslib action tableRedistUpPolicy parameter. See "CASLIB Statement" on page 745 in this document or "Add caslib" in *SAS Viya Platform: System Programming Guide*.

You can override this policy for new and existing CAS tables by using the tableRedistUpPolicy= action parameter. For a new CAS action output table, use the tableRedistUpPolicy= parameter in the action's output table parameter or parameters, when supported. Refer to the action documentation for more information. For an existing CAS table, use the table.alterTable action tableRedistUpPolicy= parameter. See "Alter table" in *SAS Viya Platform: System Programming Guide*.

See: CAS Node Scaling in *SAS Viya Platform: SAS Cloud Analytic Services* for information about worker pod scaling in CAS.

Syntax

SESSIONTABLEREDISTUPPOLICY=DEFER | NOREDIST | REBALANCE

Parameter Values

DEFER
specifies that the redistribution policy selection is deferred to a higher-level entity.

NOREDIST
specifies that table data in the session is not redistributed when the number of worker pods is increased on a running CAS server.

REBALANCE
specifies that table data in the session is rebalanced when the number of worker pods is increased on a running CAS server.

Examples

Example 1: Set Session Option SESSIONTABLEREDISTUPPOLICY

This example sets the scale-up table data redistribution policy to REBALANCE for session Casauto:

```
cas casauto sessopts=(sessionTableRedistUpPolicy=rebalance);
```

Example 2: Get the Current Value of Session Option SESSIONTABLEREDISTUPPOLICY

This example displays the current table data redistribution policy for session Casauto that is applied when the number of worker pods is increased:

```
%put Session Casauto: SESSIONTABLEREDISTUPPOLICY=
%sysfunc(getsessopt(casauto,
   sessionTableRedistUpPolicy));
```

Here is an example of the response:

```
Session Casauto: SESSIONTABLEREDISTUPPOLICY=REBALANCE
```

SUBSETSESSIONCOPIES= Session Option

specifies the number of backup copies to create of subset session in-memory table blocks for failover.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Session

Default: 0

Range: 0–5000

Notes: When set to 0, no backup copies are made.

A subset session is a session that is started on a subset of the available worker nodes on a CAS server. See “Example 2: Start a Subset Session” on page 725.

Tip: Specify a value of 1 or greater when a subset session needs to cache a global table. The cache contains the number of backup copies specified.

See: SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

SUBSETSESSIONCOPIES=*number*

TIMEOUT= Session Option

specifies the SAS Cloud Analytic Services session time-out in seconds for a new or existing session.

Valid in: CAS statement SESSOPTS option, CAS configuration file, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Session

Default: In order of descending precedence:

1. SAS system option CASTIMEOUT=, if you explicitly set it in SAS to a value greater than 0
2. 60

Range: 0–31536000

Notes: The session time-out starts when the number of connections to the session becomes zero and no actions are executing.

If a connection is established before the time-out expires, the time-out is canceled. Otherwise, the session is automatically terminated when the time-out expires.

When set to 0, the session is terminated immediately when the connection count becomes zero and no actions are executing.

See: "CASTIMEOUT= System Option" in *SAS System Options: Reference*

SAS Cloud Analytic Services: Reference in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS configuration file.

Syntax

TIMEOUT=*number*

TIMEZONE= Session Option

specifies the user local time zone.

Valid in: CAS statement SESSOPTS option, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Date and Time

Syntax

TIMEZONE=*'time-zone-name< time-zone-offset>' | 'time-zone-ID'*

Parameter Values

time-zone-name

specifies a three- or four-character time zone name.

Default The CAS server time zone.

Note The SAS system option TIMEZONE= setting has no effect on the time zone setting for CAS sessions.

See "Specifying Time Zones in SAS" in *SAS National Language Support (NLS): Reference Guide*

"Time Zone Information and Time Zone Names" in *SAS System Options: Reference* for a list of time-zone names and IDs.

CAS configuration file option cas.TIMEZONE in *SAS Viya Platform: SAS Cloud Analytic Services* for information about the CAS server time zone setting.

time-zone-offset

specifies an optional time zone offset from Greenwich, England. The offset is specified in the following form:.

```
<->hh<:mm<:ss>>
```

-

specifies a time zone offset west of Greenwich, England. Time zone offsets east of Greenwich do not require signed notation.

hh

specifies an hour by using one or two digits. If you specify an offset, the hour is required.

mm

specifies minutes by using one or two digits. Minutes are optional.

ss

specifies seconds by using one or two digits. Seconds are optional.

Restriction A time zone offset cannot be used with *time-zone-ID*.

Requirement When a time zone offset is included, the combined value must be enclosed in quotation marks.

Examples

```
cas casauto sessopts=(timeZone="PST");

cas casauto sessopts=(timeZone="GMT-05");
```

time-zone-ID
specifies a region or area value that is defined by SAS. When you specify a time zone ID, the time zone that the CAS session uses is determined by the time zone name and Daylight Saving Time rules.

Requirement: The time zone ID must be enclosed in quotation marks.

See: "Specifying Time Zones in SAS" in *SAS National Language Support (NLS): Reference Guide*

"Time Zone Information and Time Zone Names" in *SAS System Options: Reference* for a list of time-zone IDs.

Example:
```
cas casauto sessopts=(timeZone="America/New_York");
```

TRANSFERSESSION= Session Option

specifies whether this session is transferred to the new server when a CAS server state transfer occurs.

Valid in: CAS statement SESSOPTS option, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Session

Default: TRUE

Example: Disable the transfer of session Casauto to the new server when the CAS server state is transferred to another server:

```
cas casauto sessopts=(transfersession=false);
```

Syntax

TRANSFERSESSION=TRUE | FALSE

Details

When state transfer is enabled in the CAS server, sessions, tables, and the state of the current running CAS server are preserved when the server state is transferred to a new server. This typically occurs during a CAS server software update. Refer to the SAS Viya platform administration documentation for more information. By default, sessions are transferred to the new server and automatically started. Specifying TRANSFERSESSION=FALSE for a session excludes that session from the transfer. In that case, after the transfer is complete, that session no longer exists.

USEGPU= Session Option

specifies whether work is offloaded to the Graphics Processing Unit (GPU), if it is available.

Valid in: CAS statement SESSOPTS option, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Session

Syntax

USEGPU=ALWAYS | NEVER | OPTIMAL

Parameter Values

ALWAYS
specifies that work is always offloaded to the GPU, if it is available.

NEVER
specifies that work is never offloaded to the GPU.

OPTIMAL
specifies that the system determines when work is offloaded to the GPU, if it is available.

YEARCUTOFF= Session Option

specifies the first year of a 100-year span that is used by date informats and functions to read a 2-digit year.

Valid in: CAS statement SESSOPTS option, OPTIONS statement CASSESSOPTS option, GETSESSOPT function, sessionProp.setSessOpt action, sessionProp.getSessOpt action name parameter

Category: Date and Time

Default: The SAS system option YEARCUTOFF= value, which is 1940 by default.

Range: 1582–19900

Interaction: This option overrides SAS system option YEARCUTOFF=.

Note: There is no YEARCUTOFF option for the CAS server.

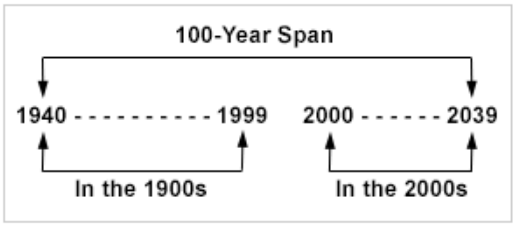
See: "YEARCUTOFF= System Option" in *SAS System Options: Reference*

Syntax

YEARCUTOFF=*number*

Details

The YEARCUTOFF= session option specifies the beginning of a range of years that span two centuries. The span of years is used by various date and datetime informats and functions to determine the correct century when reading dates with 2-digit years. By default, YEARCUTOFF=1940, which creates a span of 1940–2039, as shown in the following image.



A 2-digit year between 40 and 99, inclusive, refers to the first part of the span, 1940–1999, and is converted to 19*nn*. A 2-digit year between 00 and 39, inclusive, refers to the second part, 2000-2039, and is converted to 20*nn*. Here is an example of the 2-digit year to 4-digit year conversion when YEARCUTOFF=1940 (highlighted).

2-Digit Year	4-Digit Year
01JAN36	01JAN2036
01JAN37	01JAN2037
01JAN38	01JAN2038
01JAN39	01JAN2039
01JAN40	01JAN1940
01JAN41	01JAN1941
01JAN42	01JAN1942
01JAN43	01JAN1943
01JAN44	01JAN1944

Note: This option has no effect on existing SAS dates or dates that are read from input data that include a 4-digit year, except years with leading zeros. For example, 0076 with yearcutoff=1990 indicates 2076.

Examples

Example 1: Set Session Option YEARCUTOFF

This example sets session option YEARCUTOFF to 1950 for session Casauto:

```
cas casauto sessopts=(yearcutoff=1950);
```

Example 2: Get the Current Value of Session Option YEARCUTOFF

This example prints the current value of session option YEARCUTOFF for session Casauto:

```
%put Session Casauto: YEARCUTOFF=
%sysfunc(getsessopt(casauto,yearcutoff));
```

The GETSESSOPT arguments are not enclosed in quotation marks when it is invoked by %SYSFUNC. Here is an example of the result.

```
Session Casauto: YEARCUTOFF=1950
```

14

Statements

Dictionary 697
CAS Statement 697
CASLIB Statement 745
LIBNAME Statement: CAS Engine 765

Dictionary

CAS Statement

Starts and manages your SAS Cloud Analytic Services session.

Syntax

Form 1: Create a new session:

CAS <*session-reference*> <HOST=*"Kubernetes-service-name"*> <PORT=*number*> <USER=*user-ID*> <SESSOPTS=(*session-options*)> <AUTHDOMAIN=*authentication-domain* | AUTHINFO=*"authentication-info-file"*> <UUIDMAC=*macro-variable-name*>;

Form 2: Manage session options:

CAS *session-reference* LISTSESSOPTS;

CAS *session-reference* SESSOPTS=(*session-options*);

Form 3: Get session information:

CAS *session-reference* LIST;

CAS _ALL_ LIST;

CAS *session-reference* LISTSESSIONS;

Form 4: Disconnect or reconnect a session

CAS <*session-name*> DISCONNECT | RECONNECT;

CAS <*session-name*> UUID="*session-uuid*";

Form 5: Terminate a session or all sessions:

CAS <*session-name*> TERMINATE;

CAS _ALL_ TERMINATE;

Form 6: Get CAS server information:

CAS *session-reference* LISTABOUT | LISTSERVERSTARTOPTS;

Form 7: Add or drop a format library:

CAS *session-reference* ADDFMTLIB FMTLIBNAME=*format-library-name* <*option(s)*>;

CAS *session-reference* ADDFMTLIB TABLE=*table-name* <CASLIB=*caslib*> <FMTLIBNAME=*format-library-name*> <*option(s)*>;

CAS *session-reference* ADDFMTLIB PATH=*path* <FMTLIBNAME=*format-library-name*> <*option(s)*>;

CAS *session-reference* DROPFMTLIB FMTLIBNAME=*format-library-name* <FMTSEARCHREMOVE>;

Form 8: List formats:

CAS *session-reference* LISTFORMATS <FMTLIBNAME=*format-library-name*> <SCOPE=BOTH | SESSION | GLOBAL> <MEMBERS>;

CAS *session-reference* LISTFMTRANGES FMTNAME=*format-name*;

Form 9: Manage the format search list:

CAS *session-reference* LISTFMTSEARCH;

CAS *session-reference* FMTSEARCH=(*name1*< *name2* ... *nameN*>) <POSITION=APPEND | INSERT | REPLACE>;

CAS *session-reference* FMTSEARCH CLEAR;

Form 10: Save a format library to disk:

CAS *session-reference* SAVEFMTLIB FMTLIBNAME=*format-library-name* <*table-option(s)* | PATH=*path*>;

Form 11: Promote a format library:

CAS *session-reference* PROMOTEFMTLIB <FMTLIBNAME=*format-library-name*> <REPLACE>;

Summary of Optional Arguments

Authentication option

AUTHDOMAIN=*authentication-domain*
: obtains credentials from the credentials service for the SAS Viya platform.

AUTHINFO="*authentication-info-file*"
: specifies an authinfo file or netrc file that includes authentication information.

CAS server information options

LISTABOUT
: writes information about SAS Cloud Analytic Services to the SAS log.

LISTSERVERSTARTOPTS
: lists the SAS Cloud Analytic Services options and their current values.

Connection options

DISCONNECT
: disconnects SAS from the session.

HOST="*Kubernetes-service-name*"
: specifies the name of the CAS service running in the Kubernetes framework for the SAS Viya platform.

PORT=*number*
: specifies the port on which the CAS Kubernetes service listens for client connections.

RECONNECT
: reconnects to a session using a session name.

TERMINATE
: terminates a single connected or disconnected session, or all of the connected and disconnected sessions in your SAS client.

USER=*user-ID*
: specifies the user ID to use for connecting to SAS Cloud Analytic Services.

UUID="*session-uuid*"
: specifies the UUID of an existing session to which you want to connect.

UUIDMAC=*macro-variable-name*
: specifies a SAS macro variable name into which the UUID of the session is stored.

Session information options

LIST
: writes to the SAS log information about a single session or all of the sessions in your SAS client.

LISTHISTORY< *history_count* | _ALL_>
: prints the log of actions that were generated by statements and procedures and submitted to the CAS session.

LISTSESSIONS
: writes to the SAS log information about all of the current user's sessions that are known to SAS Cloud Analytic Services.

Session property options

LISTSESSOPTS
: lists the session options.

SESSOPTS=(*session-options*)

specifies one or more session option settings to apply during or after session start-up.

User-defined format options

ADDFMTLIB FMTLIBNAME=*format-library-name* <*option(s)*>

ADDFMTLIB TABLE=*table-name* <**CASLIB=***caslib* > <**FMTLIBNAME=***format-library-name*> <*option(s)*>

ADDFMTLIB PATH=*path* <**FMTLIBNAME=***format-library-name*> <*option(s)*>
adds a session format library.

DROPFMTLIB FMTLIBNAME=*format-library-name* <**FMTSEARCHREMOVE**>
drops a session-scope or a global-scope format library.

FMTSEARCH=(*name1* <*name2 ... nameN*>) <**POSITION=APPEND | INSERT | REPLACE**>
modifies the format library search list for the session.

FMTSEARCH CLEAR
clears the format library search list for the session.

LISTFMTRANGES FMTNAME=*format-name*
lists the ranges and labels for a format.

LISTFMTSEARCH
displays the format library search list for the session.

LISTFORMATS <**FMTLIBNAME=***format-library-name*> <**SCOPE=BOTH | SESSION | GLOBAL**> <**MEMBERS**>
lists the user-defined format libraries that are known to SAS.

PROMOTEFMTLIB FMTLIBNAME=*format-library-name* <**REPLACE**>
promotes a session-local format library to a global format library.

SAVEFMTLIB FMTLIBNAME=*format-library-name* <*table-option(s)* | PATH=*path*>
saves a session format library to a CAS table or to a file.

Wild-card option

ALL
specifies that the LIST or TERMINATE argument that follows applies to all of the sessions in the SAS client.

Without Arguments

Session CASAUTO is created.

Optional Arguments

ALL
specifies that the LIST or TERMINATE argument that follows applies to all of the sessions in the SAS client.

Requirement This option must be followed by LIST or TERMINATE.

See "LIST" on page 706

"TERMINATE" on page 714

ADDFMTLIB FMTLIBNAME=*format-library-name* <*option(s)*>
ADDFMTLIB TABLE=*table-name* <CASLIB=*caslib* > <FMTLIBNAME=*format-library-name*> <*option(s)*>
ADDFMTLIB PATH=*path* <FMTLIBNAME=*format-library-name*> <*option(s)*>
adds a session format library. By default, the format library is appended to the format library search list.

FMTLIBNAME=*format-library-name*
specifies the name of the format library to add.

Default When specified with TABLE= or PATH=, _FMTLIB*n*, where *n* starts at 1 and is incremented for each format library that is added.

Restriction The format library name cannot exceed 63 characters in length.

Notes If TABLE= or PATH= is not specified, an attempt is made to add *format-library-name* from the active caslib.

If the format library already exists in the session, specify REPLACEFMTLIB to replace the existing format library. Otherwise, an error occurs.

TABLE=*table-name* <CASLIB=*caslib*>
specifies the name of the table where the format library was previously saved using SAVEFMTLIB.

Default the name specified in the FMTLIBNAME= option

Note Do not specify this option with PATH=.

table-name
specifies the name of the table in which the format library is saved.

Note When adding a format library that is stored in a table on disk, you can specify the table name with or without the file extension (.sashdat).

CASLIB=*caslib*
specifies the name of the caslib where the table is saved.

Default the active caslib for the session

Note Do not specify this option with PATH=.

PATH=*path*
specifies the absolute path to the file in which the format information is stored.

Requirement The specified path must be authorized and accessible from the controller.

Notes Do not specify this option with TABLE= or CASLIB=.

The path can be outside of a caslib.

option(s)
specifies one or more of the following options:

POSITION=APPEND | INSERT | REPLACE | NONE
specifies the position of this format library in the format-library search list.

APPEND
appends this format library to the end of the format-library search list.

INSERT
inserts this format library at the beginning of the format-library search list.

REPLACE
replaces the current format-library search list with this format library.

NONE
does not add this format library to the format-library search list.

Default APPEND

PROMOTE
promotes the format library to global scope so that it is available to all sessions.

Note The CAS server can be configured to require administrative Superuser authority to promote a user-defined format library to global scope. By default, it is not required. If administrative Superuser authority is required and you are a CAS administrator, you must enable an automatic request for escalation to Superuser before executing the CAS statement that promotes the format library. See "CAS_ADMIN_GLOBAL_UDFMTS Macro Variable" on page 639. If you do not have administrative authority, you must contact your CAS administrator to promote your user-defined format library.

REPLACEFMTLIB
replaces the format library if it already exists.

Tip When you add a user-defined format library, it persists until it is deleted or the session is terminated. Your CAS administrator can configure CAS to add user-defined format libraries to all sessions and promote them automatically at session start-up. See "Managing User-Defined Formats in the SAS Viya Platform" .

See *SAS Cloud Analytic Services: User-Defined Formats*

Examples "Example 14: Add a Format Library That Is Stored in a SASHDAT File" on page 738

"Example 2: Migrate User-Defined Formats Using the FMTC2ITM Procedure" in *SAS Cloud Analytic Services: User-Defined Formats*

AUTHDOMAIN=*authentication-domain*
obtains credentials from the credentials service for the SAS Viya platform. SAS system option SERVICESBASEURL specifies the root URL for services on the SAS Viya platform. If it is set, SAS first requests credentials for *authentication-domain* from the credentials service.

If credentials are obtained, a note is written to the SAS log identifying the source of the credentials. Otherwise, notes are written to the SAS log indicating that credentials could not be obtained, and a session is not started.

authentication-domain
specifies the name of an authentication domain object registered on the credentials service for the SAS Viya platform that associates user credentials with an identity. The domain name can be specified with or without quotation marks.

Restriction This option is valid only when starting a new session.

Requirement To obtain credentials from the credentials service for the SAS Viya platform, the system option SERVICESBASEURL must specify the base URL for the SAS Viya platform services.

Interaction Option USER= is ignored when this option is specified.

See "Authentication Domain" in *SAS Viya Platform Encryption: Data at Rest*

"SERVICESBASEURL= System Option" in *SAS System Options: Reference*

Example This example requests that the credentials that are stored for identity CASDEMO be used to connect to CAS Kubernetes service sas-cas-server-default-client on port 5570:

```
cas casauto authdomain=casdemo
   host="sas-cas-server-default-client" port=5570;
```

AUTHINFO="*authentication-info-file*"
specifies an authinfo file or netrc file that includes authentication information.

Default SAS system option AUTHINFO=, if set. Otherwise, file **~/.authinfo**.

Restriction This option is valid only when starting a new session.

Note An authinfo file is required when running batch jobs.

See "Create an Authinfo File" in *Client Authentication Using an Authinfo File*

DISCONNECT
disconnects SAS from the session.

Notes The session name is preserved for use with the RECONNECT option for the duration of the SAS session. See "Example 9: Connect to an Existing Session" on page 730.

When you disconnect from the default session, SAS system option SESSREF= continues to reference the disconnected session. To access the server in that case, you must reconnect to the default session, start a new session, or set system option SESSREF= to reference an existing connected session. See "Example 1: Start a Session" on page 723 and "SESSREF= System Option" in *SAS System Options: Reference*.

Tip The session time-out value determines the lifetime in seconds of a disconnected session. The default is 60 seconds. The session time-out starts when the number of connections becomes zero and no actions are running. After the time-out expires, the session is terminated. Before you disconnect, if necessary, use the SESSOPTS= on page 713 option to increase the time-out value.

See "Example 8: Disconnect from a Session" on page 729

DROPFMTLIB FMTLIBNAME=*format-library-name* <FMTSEARCHREMOVE>
drops a session-scope or a global-scope format library.

FMTLIBNAME=*format-library-name*
specifies the name of the format library to drop.

This argument is required.

FMTSEARCHREMOVE
removes the format library from the format search list.

Note The CAS server can be configured to require administrative Superuser authority to drop a global user-defined format library. By default, it is not required. If administrative Superuser authority is required and you are a CAS administrator, you must enable an automatic request for escalation to Superuser before executing the CAS statement that drops the format library. See "CAS_ADMIN_GLOBAL_UDFMTS Macro Variable" on page 639. If you do not have administrative authority, you must contact your CAS administrator to drop your user-defined format library from global scope.

Tip If a session-scope and a global-scope format library with the same name exist, the session-scope format library is dropped first. To drop the global format library in that case, execute the drop command again.

See *SAS Cloud Analytic Services: User-Defined Formats*

Example "Example 21: Drop a Format Library from Session Scope" on page 744

FMTSEARCH=(*name1* <*name2* ... *nameN*>) <POSITION=APPEND | INSERT | REPLACE>
modifies the format library search list for the session. The format library search list determines the order in which the format libraries are searched when a format is requested.

(*name1* <*name2* ... *nameN*>)
specifies a list of one or more format-library names enclosed in parentheses.

Each name is separated by a space.

POSITION=APPEND | INSERT | REPLACE
specifies the position of the format libraries in the format-library search list.

APPEND
appends the format libraries to the end of the format-library search list.

INSERT
inserts the format libraries at the beginning of the format-library search list.

REPLACE
replaces the current format-library search list with the specified format libraries.

Default APPEND

Notes Global format libraries are not automatically added to your format library search list when you start your session. Use this option to add any global format libraries that you want to use.

When a global format library is added to the search list, the global format library is created in session scope.

See "How User-Defined Formats are Located In SAS Cloud Analytic Services" in *SAS Cloud Analytic Services: User-Defined Formats* for information about the format library search list in CAS.

SAS Cloud Analytic Services: User-Defined Formats for information about using user-defined formats in CAS.

Example Add format libraries FMTLIB1 and FMTLIB2 to the beginning of the format library search list for session Casauto:

```
cas casauto fmtsearch=(fmtlib1 fmtlib2) position=insert;
```

Examples "Example 17: Add Global Format Libraries to a Session's Format Search List" on page 740

"Example 18: Change the Format Library Search List" on page 741

FMTSEARCH CLEAR
clears the format library search list for the session.

See *SAS Cloud Analytic Services: User-Defined Formats*

Example Clear the format search list for session Casauto:

```
cas casauto fmtsearch clear;
```

HOST="*Kubernetes-service-name*"
specifies the name of the CAS service running in the Kubernetes framework for the SAS Viya platform. When a new connection is established using the CAS statement, a connect message in the SAS log reflects the name of the CAS Kubernetes service. System option CASHOST= and macro variable _CASHOST_ are also updated with the current service name.

Aliases CASHOST

SERVER

Default Macro variable _CASHOST_, if set. Otherwise, SAS system option CASHOST.

Restriction This option is valid only when starting a new session.

Requirement The name must be enclosed in quotation marks.

Interactions This option overrides macro variable _CASHOST_ and SAS system option CASHOST.

Macro variable _CASHOST_ and SAS system option CASHOST are updated to specify the current CAS Kubernetes service name.

Note If the specified service name is not valid and an attempt is made to start a new session, an error occurs.

Tips To connect to a personal CAS server, specify macro variable reference &_CASHOST_PERSONAL_ in HOST=.

To connect to the shared CAS server, specify macro variable reference &_CASHOST_SHARED_ in HOST=.

See "CASHOST= System Option" in *SAS System Options: Reference*

"_CASHOST_ Macro Variable" on page 633

"_CASHOSTCONT_ Macro Variable" on page 637

"_CASHOST_PERSONAL_ Macro Variable" on page 635

"_CASHOST_SHARED_ Macro Variable" on page 636

Examples Start a session on sas-cas-server-default-client using port 5570:

```
cas casauto host="sas-cas-server-default-client" port=5570;
```

Connect to the personal CAS server:

```
cas casauto host=&_cashost_personal_;
```

Connect to the shared CAS server:

```
cas casauto host=&_cashost_shared_;
```

LIST

writes to the SAS log information about a single session or all of the sessions in your SAS client. The information includes the session reference, the session state, the service name and port of the SAS Cloud Analytic Services Kubernetes service to which it is connected, and the session UUID.

Note The server is not accessed for the information.

Tips Use this option with *session-reference* to list information about a single session, or use it with _ALL_ on page 700 to list information about all of the sessions that are known to your SAS client.

To list information about all of the sessions in all of your SAS clients, use LISTSESSIONS.

See "Example 3: List Information about the Sessions in Your SAS Client" on page 726

LISTABOUT

writes information about SAS Cloud Analytic Services to the SAS log. The information is organized as shown in the following table.

Section	Field	Description
About	CAS	Product name: SAS Cloud Analytic Services.
	Version	Short-form CAS version.
	VersionLong	Long-form CAS version.
	Copyright	Copyright information.
	ServerTime	Current date and time on the CAS server.
System	Host name	CAS Kubernetes service name.
	OS Name	CAS host information.
	OS Family	
	OS Release	
	OS Version	
	Model Number	
	Documentation	CAS documentation URL.
License	site	Site information.
	siteNum	
	expires	License expiration date and time.
	gracePeriod	Grace period in days.

Section	Field	Description
	warningPeriod	Expiration warning period in days.
	CASHostAccountRequired	Indicates whether CAS environment variable CASALLHOSTACCOUNTS is set. Values can be ALL or OPTIONAL. ALL indicates that CASALLHOSTACCOUNTS is set. OPTIONAL indicates that it is not set. See env.CASALLHOSTACCOUNTS in *SAS Viya Platform: SAS Cloud Analytic Services*.

LISTFMTRANGES FMTNAME=*format-name*
lists the ranges and labels for a format.

FMTNAME=*format-name*
specifies the name of the format to list.

The format libraries in the format-library search list are searched in the order specified. The ranges and labels are listed for the first instance of *format-name* that is found.

See *SAS Cloud Analytic Services: User-Defined Formats*

Example "Example 19: List the Ranges in a Format" on page 742

LISTFMTSEARCH
displays the format library search list for the session. The format library search list determines the order in which the format libraries are searched when a format is requested.

See "How User-Defined Formats are Located In SAS Cloud Analytic Services" in *SAS Cloud Analytic Services: User-Defined Formats* for information about the format library search list in CAS.

SAS Cloud Analytic Services: User-Defined Formats for information about using user-defined formats in CAS.

Examples "Example 17: Add Global Format Libraries to a Session's Format Search List" on page 740

"Example 18: Change the Format Library Search List" on page 741

LISTFORMATS <FMTLIBNAME=*format-library-name*> <SCOPE=BOTH | SESSION | GLOBAL> <MEMBERS>

lists the user-defined format libraries that are known to SAS.

FMTLIBNAME=*format-library-name*

specifies the name of the format library.

If FMTLIBNAME= is not specified, all of the format libraries that are known to SAS are listed.

SCOPE=BOTH | SESSION | GLOBAL

specifies the scope.

BOTH

lists both SESSION and GLOBAL format libraries that are known to SAS.

SESSION

lists the format libraries that are known to SAS.

GLOBAL

lists the format libraries that are known globally to all SAS sessions.

Default BOTH

MEMBERS

lists the names of the members in each format library.

See *SAS Cloud Analytic Services: User-Defined Formats*

Example "Example 16: List the Session-Scope Formats in a Session" on page 739

LISTHISTORY< *history_count* | _ALL_>

prints the log of actions that were generated by statements and procedures and submitted to the CAS session. For information about actions, see "Running Actions with CASL" in *SAS Cloud Analytic Services: CASL Programmer's Guide*.

history_count

specifies the number of the most recent actions that are to be listed.

Range 0–1999999999

Note 0 is equivalent to _ALL_

ALL

lists all of the actions that have been executed in the current session.

Default Lists the last 10 actions that were executed in the current session.

Examples Print the last 10 actions that were executed in the current session:

```
cas casauto listhistory;
```

Print the last 5 actions that were executed in the current session:

```
cas casauto listhistory 5;
```

Print all of the actions that were executed in the current session:

```
cas casauto listhistory _all_;
```

LISTSERVERSTARTOPTS
lists the SAS Cloud Analytic Services options and their current values. For each option, a note containing information about the option is written to the SAS log. The information includes the option name, option value type, current option value, value range (when applicable), and value source, as shown in the following example.

```
NOTE: Name = logFlushTime
         Type = INT RANGE
         Value = 100
         Minimum = -1
         Maximum = 86400
         Source = default
```

Alias LISTSSO

Restriction You must have administration privileges to use this option.

LISTSESSIONS
writes to the SAS log information about all of the current user's sessions that are known to SAS Cloud Analytic Services. The information includes a session name, the session UUID, the session state, the method that was used for user authentication, and the user ID. The session name consists of the session display name and the session creation date and time in the following format:

session-display-name:*Weekday Month dd hh*:*mm*:*ss yyyy*

Note: On a SAS client, by default, the session display name is the same as the session reference. See "*session-reference*" on page 715. The session display name can be changed by using the session.sessionName action. See "Change session name" in *SAS Viya Platform: System Programming Guide*.

Requirement A connection to a session is required to use the LISTSESSIONS option.

See "Example 4: List Information about All of Your Sessions" on page 727

Example Use session Casauto to list all of your sessions.

```
cas casauto listsessions;
```

For each session that is found, a note containing information about that session is written to the SAS log. Here is an example:

```
NOTE: SessionName = CASAUTO:Fri Jun 2 10:56:07 2023
         UUID = 735542cb-aed4-c44c-970b-0d426b154676
         State = Connected
         Authentication = Active Directory
         Userid = sasdemo
```

LISTSESSOPTS
lists the session options.

See "Session Options" on page 645 for a list of the session options.

"Example 5: List the Properties for a Session" on page 728

Example List the session option settings for session Casauto:

```
cas casauto LISTSESSOPTS;
```

PORT=*number*
specifies the port on which the CAS Kubernetes service listens for client connections.

Alias CASPORT

Default Macro variable _CASPORT_, if set. Otherwise, SAS system option CASPORT.

Range 1-65535

Restriction This option is valid only when starting a new session.

Interaction This option overrides macro variable _CASPORT_ and SAS system option CASPORT.

See "_CASPORT_ Macro Variable" on page 637

"CASPORT= System Option" in *SAS System Options: Reference*

PROMOTEFMTLIB FMTLIBNAME=*format-library-name* <REPLACE>
promotes a session-local format library to a global format library.

FMTLIBNAME=*format-library-name*
specifies the name of the format library.

This option is required.

REPLACE
replaces the format library if it is already promoted.

Requirement *Format-library-name* must be in the session format search list. Use the LISTFMTSEARCH option to see the current format search list.Use the FMTSEARCH on page 704 option to update the search list, if needed.

Note The CAS server can be configured to require administrative Superuser authority to promote a user-defined format library to global scope. By default, it is not required. If administrative Superuser authority is required and you are a CAS administrator, you must enable an automatic request for escalation to Superuser before executing the CAS statement that promotes the format library. See "CAS_ADMIN_GLOBAL_UDFMTS Macro Variable" on page 639. If you do not have administrative authority, you must contact your CAS administrator to promote your user-defined format library.

Tips When you promote a user-defined format, it persists as a global format library until it is deleted or the CAS server is restarted. Your CAS administrator can configure CAS to add user-defined format libraries to all sessions and promote them automatically at session

start-up. See "Managing User-Defined Formats in the SAS Viya Platform" .

Global format libraries are not automatically included in the format search list for new sessions. Use FMTSEARCH on page 704 to add any global format libraries that you want to use to your new session's format search list.

See *SAS Cloud Analytic Services: User-Defined Formats*

Example "Example 20: Promote a Format Library to Global Scope" on page 743

RECONNECT

reconnects to a session using a session name.

Note If the target session has terminated, an error message is written to the SAS log, and the session is automatically deleted from the SAS client. In that case, to connect to the server, you must create a new session.

Tip If the session is not known to SAS, connect using the session UUID. Do not specify the RECONNECT keyword in that case. See "UUID="*session-uuid*"" on page 715.

See "Example 9: Connect to an Existing Session" on page 730

SAVEFMTLIB FMTLIBNAME=*format-library-name* <*table-option(s)*> | PATH=*path*>

saves a session format library to a CAS table or to a file.

FMTLIBNAME=*format-library-name*

specifies the name of the format library to save.

This option is required.

table-option(s)

specifies one or more of the following options:

CASLIB=*caslib*

specifies the caslib in which the table is stored.

Default the active caslib for the session

Requirement You must have Write access to the specified caslib.

Note Do not specify this option with PATH=.

Tip Contact your SAS Cloud Analytic Services system administrator if you want to store your format library in the FORMATS caslib.

TABLE=*table-name*

specifies the name of the table in which the format library is saved.

Default the name specified in the FMTLIBNAME= option

Notes Do not specify this option with PATH=.

When saving a format library to a table on disk, you can specify the table name with or without the file extension (.sashdat).

If the table already exists in the caslib, specify REPLACE to replace the existing table. Otherwise, an error occurs.

Do not change the name of the format library file after you save it. If you change the file name, the format library that it contains cannot be loaded.

PROMOTE
promotes the format library to global scope so that it is available to all sessions.

Notes This option is ignored when PATH= is specified.

The CAS server can be configured to require administrative Superuser authority to promote a user-defined format library to global scope. By default, it is not required. If administrative Superuser authority is required and you are a CAS administrator, you must enable an automatic request for escalation to Superuser before executing the CAS statement that promotes the format library. See "CAS_ADMIN_GLOBAL_UDFMTS Macro Variable" on page 639. If you do not have administrative authority, you must contact your CAS administrator to promote your user-defined format library.

REPLACE
replaces the table in the caslib if it already exists.

Alias REPLACETABLE

Note This option is ignored when PATH= is specified.

PATH=*path*
specifies the absolute path to the file in which the format library is to be saved.

Requirement The specified path must be authorized and accessible from the controller.

Notes Do not specify this option with TABLE= or CASLIB=.

The path can be outside of a caslib.

See *SAS Cloud Analytic Services: User-Defined Formats*

Example "Example 13: Create a Format Library and Save It to a SASHDAT File" on page 735

SESSOPTS=(*session-options*)
specifies one or more session option settings to apply during or after session start-up.

See "Session Options" on page 645 for a list of the options that you can specify for *session-options*.

"Example 7: Change a Session Property Value" on page 729

"Program: Start a Session with Custom Properties" on page 724

TERMINATE

terminates a single connected or disconnected session, or all of the connected and disconnected sessions in your SAS client.

Alias CLEAR

Notes When you terminate the active session, SAS system option SESSREF= continues to reference the terminated session. A note is written to the SAS log indicating that the session identified by SAS system option SESSREF= was terminated. To access the server in that case, you must start a new session or set system option SESSREF= to an existing connected session. See "Example 1: Start a Session" on page 723 and "SESSREF= System Option" in *SAS System Options: Reference*.

If you terminate a session that you disconnected using DISCONNECT, the disconnected session is removed from your client's session list, but it continues to run on the server until its time-out expires. To terminate the session in that case, see "Terminating a Disconnected Session" on page 734.

Tip Use this option with *session-reference* to terminate a single connected or disconnected session, or use it with _ALL_ to terminate all of the currently connected and disconnected sessions in your SAS client.

See "Example 12: Terminate a Session" on page 733

USER=*user-ID*

specifies the user ID to use for connecting to SAS Cloud Analytic Services.

Alias CASUSER

Default SAS system option CASUSER=, if set.

Restriction This option is valid only when starting a new session.

Requirement *User-ID* must match a user ID in your authinfo file.

Interaction This option overrides SAS system option CASUSER=.

Note When you use SAS Data and AI Studio (known as SAS Studio prior to 2026.06), the user credentials that you used to sign on are used to authenticate your connection to CAS. The USER= option is not needed in that case. The USER= option or SAS system option CASUSER= is used when submitting code to CAS from the command line, in batch mode.

See "CASUSER= System Option" in *SAS System Options: Reference*

SAS Viya Platform Administration: Authentication

UUID="*session-uuid*"
specifies the UUID of an existing session to which you want to connect.

Restriction Only one named session can be associated with a given session UUID.

Requirement *session-uuid* must be 36 characters in length and must be enclosed in quotation marks.

Note If you attempt to connect to a session that is busy, an error message is written to the SAS log, and the connection is not established. The error message indicates that the request cannot be submitted because the target session is busy running actions.

Tips You can view the UUID for all of your sessions by using the LISTSESSIONS option in a CAS statement for your currently active session.

If the target session is busy, use the actionstatus action in the Session action set to determine when the target session is free.

See "LIST" on page 706

"Program: Connect to an Existing Session Using the Session Name and UUID" on page 731

UUIDMAC=*macro-variable-name*
specifies a SAS macro variable name into which the UUID of the session is stored.

Tip The UUIDMAC= option is useful if you want subsequent SAS program steps to connect to the session by specifying the UUID.

See "Program: Start a Session and Store the UUID in a Macro Variable" on page 725

Session Reference

session-reference
specifies a valid SAS name that is less than 256 characters. This name is used to reference the session when needed such as in CAS statements, in the CAS procedure SESSION statement, and in SESSREF= options.

Default CASAUTO

Notes Do not enclose *session-reference* in quotation marks.

If session *session-reference* does not exist, the session is started, and macro variable _SESSREF_ and SAS system option SESSREF are set to *session-reference*. Also, macro variable _CASHOST_ and SAS system option CASHOST are updated to specify the current CAS Kubernetes service name.

The SAS Cloud Analytic Services system administrator can limit the number of concurrent sessions on the CAS server. See "Maximum Number of Sessions on the Server" on page 717.

The session display name, which is displayed in the LISTSESSIONS output, in the session.listSessions output, and in the session.sessionId action output, might differ from *session-reference*. By default, the session display name is *session-reference*, but it can be changed by using the session.sessionName action. See "Change session name" in *SAS Viya Platform: System Programming Guide*.

Details

What Can I Do with the CAS Statement?

You can do the following tasks with the CAS statement:

- list information about a specific SAS Cloud Analytic Services session or all of your sessions
- list the properties of a session
- manage format libraries in a session
- change one or more session properties
- disconnect a session
- connect to an existing session
- start a session
- terminate a session

Starting Your Initial Session

You must start a session in order to connect to SAS Cloud Analytic Services. After you start your session, you can use it to complete your tasks. If you are using SAS Data and AI Studio (known as SAS Studio prior to 2026.06), a session might have been started for you when you signed on. To see whether a session has already been started, run the following statement:

```
cas _all_ list;
```

If a session has been started, you should see a message similar to the following in the SAS log.

```
NOTE: Session CASAUTO is ACTIVE using port 5570 and host sas-cas-server-default-
client for
user sasdemo. The session UUID is session-id.
```

The note in the SAS log indicates the name and port of the CAS Kubernetes service to which the session is connected and the session UUID. If a session has not been started, no messages are written to the SAS log. In that case, use a CAS statement to start a session. For examples, see "Example 1: Start a Session" on page 723.

After your session has been started, you can use the CAS statement to perform management tasks on your session. Some examples are listing or changing properties, managing format libraries, and so on.

CAS Statement Status

When you execute a CAS statement for the first time in your SAS session, global macro variable CASSTMTERR is created. It is set to 0 if the CAS statement was successful, 1 if an error occurred, or 2 if a warning was issued. The CASSTMTERR macro variable is updated each time you execute a CAS statement. You can use the CASSTMTERR macro variable in your SAS program to test the status of your last CAS statement and proceed accordingly. For an example, see "Example 8: Disconnect from a Session" on page 729.

Maximum Number of Sessions on the Server

The maximum number of concurrent sessions is determined by SAS Cloud Analytic Services configuration option MAXSESSIONS. The default is 5000. Administrative users are exempt from this limit. When the limit is reached, non-administrative users are denied access until the number of sessions drops below the maximum. In that case, an error message is written to the SAS log indicating that the limit is reached, and the requested session is not started. For information about configuration option MAXSESSIONS, see Configuration File Options in *SAS Viya Platform: SAS Cloud Analytic Services*.

Managing User-Defined Formats Using the CAS Statement

The CAS statement provides several options that enable you to perform common tasks related to user-defined format libraries. The following table maps common user-defined format library tasks to the applicable CAS statement options and examples.

Table 14.1 *Performing Common Tasks for User-Defined Formats Using the CAS Statement*

Task	What to Use	Example(s)
Migrate your user-defined formats stored in SAS format catalogs to CAS.	The IFTC2ITM procedure and the CAS statement LOADFORMATS option	"Migrate User-Defined Formats from SAS to CAS" in *SAS Cloud Analytic Services: User-Defined Formats*
Add a user-defined format library stored in a SASHDAT file to your session.	The CAS statement ADDFMTLIB on page 701 option	"Example 14: Add a Format Library That Is Stored in a SASHDAT File" (p. 738)
Promote a session-scope user-defined format library in your session to global scope.	The CAS statement PROMOTEFMTLIB on page 711 option	"Example 20: Promote a Format Library to Global Scope" (p. 743)

Task	What to Use	Example(s)
List the format libraries and formats that are available in your session.	The CAS statement LISTFORMATS on page 709 option	"Example 16: List the Session-Scope Formats in a Session" (p. 739)
List the ranges in a user-defined format in your session.	The CAS statement LISTFMTRANGES on page 708 option	"Example 19: List the Ranges in a Format" (p. 742)
Display the user-defined format library search list for your session.	The CAS statement LISTFMTSEARCH on page 708 option	"Example 17: Add Global Format Libraries to a Session's Format Search List" (p. 740) "Example 18: Change the Format Library Search List" (p. 741)
Modify your session's user-defined format library search list.	The CAS statement FMTSEARCH on page 704 option	"Example 17: Add Global Format Libraries to a Session's Format Search List" (p. 740) "Example 18: Change the Format Library Search List" (p. 741)
Save a format library in your session to a SASHDAT file.	The CAS statement SAVEFMTLIB on page 712 option	"Example 13: Create a Format Library and Save It to a SASHDAT File" (p. 735)
Drop a session-scope or global-scope user-defined format library.	The CAS statement DROPFMTLIB on page 704 option	"Example 21: Drop a Format Library from Session Scope" (p. 744)

For information about user-defined formats in SAS Cloud Analytic Services, see *SAS Cloud Analytic Services: User-Defined Formats*.

Troubleshooting Session Errors

What Session-Related Errors Does This Section Cover?

This section covers the following CAS session-related errors:

- A Connection to a Cloud Analytic Services Session Could Not Be Made
- Could Not Find Netrc or Authinfo File
- Missing Session Information
- Request to Connect Failed for UUID
- Server Has Met the Maximum Number of Concurrent Sessions
- Session Cannot Be Resolved

- Session Is Not Recognized
- Session Connection for a Session Is Not Active
- The Session is Busy Running Actions on page 722
- Unable to Connect to Cloud Analytic Services

A Connection to a Cloud Analytic Services Session Could Not Be Made

```
WARNING: Session session-reference is disconnected.

ERROR: A connection to the Cloud Analytic Services session
session-reference could not be made. Make sure that the session name is
correctly specified and that it is an active session.
```

Note: The warning message might not appear.

Cause	Solution
A reference to session *session-reference* was made in the CASUTIL procedure SESSREF= option but the specified session is disconnected, does not exist, or is not known to SAS.	■ If a warning message indicates that *session-reference* is disconnected, reconnect to session *session-reference*. See "Example 9: Connect to an Existing Session" on page 730. ■ Verify that session *session-reference* exists. See "Example 4: List Information about All of Your Sessions" on page 727. If session *session-reference* does not exist, specify a different session or start session *session-reference*. See "Example 1: Start a Session" on page 723.

Could Not Find Netrc or Authinfo File

```
ERROR: Could not find netrc or authinfo file.
```

Cause	Solution
Your netrc or authinfo file that is required for authentication could not be found.	Verify that a valid authinfo file exists for your user ID and that the file permissions grant Read and Write access to you only. See "Create an Authinfo File" in *Client Authentication Using an Authinfo File*.

Missing Session Information

```
WARNING: Session session-reference is disconnected.

ERROR: Missing session information.
```

Note: The warning message might not appear.

Cause	Solution
A reference to session *session-reference* was made in a LIBNAME statement, but the specified session is disconnected, does not exist, or is not known to SAS.	■ If a warning message indicates that *session-reference* is disconnected, reconnect to session *session-reference*. See "Example 9: Connect to an Existing Session" on page 730. ■ Verify that session *session-reference* exists. See "Example 4: List Information about All of Your Sessions" on page 727. If session *session-reference* does not exist, specify a different session or start session *session-reference*. See "Example 1: Start a Session" on page 723.

Request to Connect Failed for UUID

```
ERROR: Request to CONNECT failed for UUID session-uuid.
Failure occurs when a disconnected CAS session exceeds the timeout
value and terminates, when the specified UUID is not correct or
designates a session that is already connected, or when
authentication fails.

ERROR: Connection failed. Server returned: Authentication failed: Access denied.
```

Note: The authentication error message might not appear.

Cause	Solution
An attempt was made to connect to a session using its name and UUID, but the session was not found or a connection could not be established.	■ If authentication failed, verify that your credentials are correct. For information about authentication in SAS Cloud Analytic Services, see SAS Viya Platform Administration: Authentication. ■ Verify that session *session-uuid* exists. See "Example 4: List Information about All of Your Sessions" on page 727. If session *session-uuid* does not exist, specify a different session UUID or start a new session. See "Example 1: Start a Session" on page 723.

Server Has Met the Maximum Number of Concurrent Sessions

The following error messages appear:

```
ERROR: Unable to connect to Cloud Analytic Services Kubernetes-service-name on
port port-number

ERROR: Connection failed. The server has met the maximum number of concurrent
sessions. Contact your site administrator.
```

Cause	Solution
The maximum number of concurrent sessions specified by the SAS Cloud Analytic Services MAXSESSIONS configuration parameter has been reached.	Contact your SAS Cloud Analytic Services system administrator. For more information, see "Maximum Number of Sessions on the Server" on page 717.

Session Cannot Be Resolved

```
WARNING: Session session-reference is disconnected.

ERROR:  Session reference 'session-reference' cannot be resolved
```

Note: The warning message might not appear.

Cause	Solution
A reference to session *session-reference* was made in the CAS procedure SESSION statement, but the specified session is disconnected, does not exist, or is not known to SAS.	■ If a warning message indicates that *session-reference* is disconnected, reconnect to session *session-reference*. See "Example 9: Connect to an Existing Session" on page 730. ■ Verify that session *session-reference* exists. See "Example 4: List Information about All of Your Sessions" on page 727. If session *session-reference* does not exist, specify a different session or start session *session-reference*. See "Example 1: Start a Session" on page 723.

Session Is Not Recognized

```
WARNING: Session session-reference is disconnected.

ERROR: Request failed. Session session-reference not recognized.
```

Note: The warning message might not appear.

Cause	Solution
A request was made on session *session-reference*, but session *session-reference* is disconnected, does not exist, or is not known to SAS.	■ If a warning message indicates that *session-reference* is disconnected, reconnect to session *session-reference*. See "Example 9: Connect to an Existing Session" on page 730. ■ Verify that session *session-reference* exists. See "Example 4: List Information about All of Your Sessions" on page 727. If session *session-reference* does not exist, specify a different session or start session *session-reference*. See "Example 1: Start a Session" on page 723.

Session Connection for a Session Is Not Active

An error message similar to the following appears:

```
ERROR: Request to LISTSESSOPTS failed. The session
connection for session-reference is not active.
```

Cause	Solution
A request such as LISTSESSOPTS or LISTFORMATS was made on session *session-reference*, but session *session-reference* is disconnected.	Reconnect to session *session-reference*. See "Example 9: Connect to an Existing Session" on page 730.

The Session Is Busy Running Actions

The following error message appears:

```
ERROR: The request cannot be submitted to session session-reference. The session
is busy running actions.
```

Table 14.2 *Error Cause and Solution*

Cause	Solution
A CAS statement was submitted for a session, but the session is currently busy running an action asynchronously. One or	Wait for asynchronous actions to complete, and then run your CAS statement again. For an existing session, use SAS function SESSBUSY to determine when the session is free. Here is an example: `%put Session Casauto is busy: %sysfunc(sessbusy(casauto));`

Cause	Solution
more additional error messages might appear.	In this example, function SESSBUSY returns 1 if session Casauto is currently busy or 0 if it is free. If you received this error while attempting to connect to another session using the session's UUID, use the actionstatus action in the Session action set to determine when the target session is free.

Unable to Connect to Cloud Analytic Services

One of the following error messages appears:

```
ERROR: Unable to connect to Cloud Analytic Services Kubernetes-service-name
on port port-number Verify connection parameters and retry.

ERROR: Connection failed. Server returned: Authentication
failed: Access denied.
```

Cause	Solution
The specified *Kubernetes-service-name* or *port-number* is invalid, or authentication failed.	■ If authentication failed, verify that your credentials are correct. For information about authentication in SAS Cloud Analytic Services, see SAS Viya Platform Administration: Authentication. ■ Verify that the specified *Kubernetes-service-name* and *port-number* are correct. ■ If the connection parameters are correct, verify that *Kubernetes-service-name* is available.

Examples

Example 1: Start a Session

Program: Start a Session with Default Properties

This example starts a session that is named Casauto with default session properties. If necessary, set system options CASHOST= and CASPORT= to a CAS Kubernetes service name and port that are valid for your site.

```
/* options cashost="sas-cas-server-default-client" casport=5570; */
cas casauto;
```

SAS Log

Notes similar to the following are written to the SAS log:

```
NOTE: The session CASAUTO connected successfully to Cloud Analytic Services  1
      sas-cas-server-default-client using port 5570. The UUID is session-UUID.
      The user is sasdemo and the default CASLIB is
      CASUSER(sasdemo).
NOTE: The SAS option SESSREF was updated with the value CASAUTO.
NOTE: The SAS macro _SESSREF_ was updated with the value CASAUTO.
NOTE: The session is using nnn workers.  2
```

1 This note in the SAS log indicates the name and port number of the CAS Kubernetes service to which the session is connected, and the session UUID.

2 This note indicates the number of worker nodes that are available to your session.

Program: Start a Session with Custom Properties

This example starts a session that is named Casauto. Session options are specified to set the active caslib to CASUSER and to enable metrics reporting:

```
/* options cashost="sas-cas-server-default-client" casport=5570; */
cas casauto sessopts=(caslib=casuser metrics=True);
```

SAS Log

Notes similar to the following are written to the SAS log:

```
NOTE: The session CASAUTO connected successfully to Cloud Analytic Services
      sas-cas-server-default-client using port 5570. The UUID is session-UUID.
      The user is sasdemo and the default CASLIB is
      CASUSER(sasdemo).
NOTE: The SAS option SESSREF was updated with the value CASAUTO.
NOTE: The SAS macro _SESSREF_ was updated with the value CASAUTO.
NOTE: The session is using nnn workers.
NOTE: 'CASUSER(sasdemo)' is now the active caslib.
NOTE: Action 'setsessopt' used (Total process time):
NOTE:       real time               0.181410 seconds
NOTE:       cpu time                0.446927 seconds (246.36%)
NOTE:       total nodes             142 (4536 cores)
NOTE:       total memory            35.47T
NOTE:       memory                  13.09M (0.00%)
NOTE: The CAS statement request to update one or more session options
      for session CASAUTO completed.
```

Program: Start a Session and Store the UUID in a Macro Variable

This example starts a session that is named Casauto and stores the session UUID in macro variable CASAUTO_UUID for later use:

```
/* options cashost="sas-cas-server-default-client" casport=5570; */
cas casauto uuidmac=casauto_uuid;
%put Session casauto UUID: &casauto_uuid;
```

SAS Log

The %PUT statement writes the session UUID to the SAS log. Here is an example.

```
Session casauto UUID: 55c7425b-e383-794a-830a-055731b4e211
```

Additional Information

Here are some best practices to follow when creating your own session:

- Always specify the name of your session in CAS statements that provide a SESSREF= option such as the CASLIB and LIBNAME statements. This helps ensure that the caslib or library reference is associated with the correct session. See “CASLIB Statement” on page 745 and “LIBNAME Statement: CAS Engine” on page 765.
- If you need to disconnect from your session, be sure to set the TIMEOUT property for the session to an appropriate value before you disconnect. If no other client connections exist and no actions are running when you disconnect, your session is automatically terminated if you do not reconnect within the time-out period. See TIMEOUT= on page 690.
- To help conserve system resources, always terminate your session when you are finished with it.

Example 2: Start a Subset Session

Program: Start a Subset Session

This example starts a session that is named Casauto on only four worker nodes instead of all of the available nodes on the CAS server. A session that is started on

a subset of the available worker nodes is called a subset session. It is useful when all of the worker nodes are not needed.

```
/* options cashost="sas-cas-server-default-client" casport=5570; */
cas casauto sessopts=(nworkers=4);
```

SAS Log

Notes similar to the following are written to the SAS log:

```
NOTE: The session CASAUTO connected successfully to Cloud Analytic Services
      sas-cas-server-default-client using port 5570. The UUID is session-UUID.
      The user is sasdemo and the default CASLIB is
      CASUSER(sasdemo).
NOTE: The SAS option SESSREF was updated with the value CASAUTO.
NOTE: The SAS macro _SESSREF_ was updated with the value CASAUTO.
NOTE: The session is using 4 workers.
```

Example 3: List Information about the Sessions in Your SAS Client

Program: List a Specific Session in Your SAS Client

This example lists information about SAS client session Casauto:

```
cas casauto list;
```

SAS Log

A note similar to the following is written to the SAS log:

```
NOTE: Session CASAUTO is ACTIVE using port 5570 and host
sas-cas-server-default-client for user sasdemo. The session UUID is
session-UUID.
```

Program: List All of the Sessions in Your SAS Client

This example lists information about all of the sessions that are in a SAS client. In this example, the sessions are Casauto, Mysess1, and Mysess2:

```
cas _all_ list;
```

SAS Log

A note is written to the SAS log for each of the three sessions as shown in the following example:

```
NOTE: Session CASAUTO is ACTIVE using port 5570 and host
sas-cas-server-default-client for user sasdemo. The session UUID is
session-UUID.
NOTE: Session MYSESS1 is ACTIVE using port 5570 and host
sas-cas-server-default-client for user sasdemo. The session UUID is
session-UUID.
NOTE: Session MYSESS2 is ACTIVE using port 5570 and host
sas-cas-server-default-client for user sasdemo. The session UUID is
session-UUID.
```

Example 4: List Information about All of Your Sessions

This example uses existing session Casauto to list information about all of the current user's sessions that are known to SAS Cloud Analytic Services. You must be connected to an existing session in order to use the LISTSESSIONS option.

```
cas casauto listsessions;
```

SAS Log

For each of the current user's sessions, a note is written to the SAS log that contains information about that session. Here is an example that shows information for sessions Casauto, Mysess1, and Mysess2.

```
NOTE: SessionName = CASAUTO:Mon Feb  8 12:49:00 2016
         UUID= session-UUID
         State = Connected
         Authentication = Active Directory
         Userid = sasdemo
NOTE: SessionName = MYSESS1:Mon Feb  8 12:49:03 2016
         UUID= session-UUID
         State = Connected
         Authentication = Active Directory
         Userid = sasdemo
NOTE: SessionName = MYSESS2:Mon Feb  8 12:49:05 2016
         UUID= session-UUID
         State = Connected
         Authentication = Active Directory
         Userid = sasdemo
NOTE: Request to LISTSESSIONS completed for session CASAUTO.
```

Example 5: List the Properties for a Session

Program

This example lists the properties for session Casauto:

```
cas casauto listsessopts;
```

SAS Log

A note containing property information is written to the SAS log for each session property. Here is a partial example.

```
NOTE: Name = appTag
         UsageType = Session
         Type = String
         Value =
         Default Value =
         Group = Action
         Description = specifies the string to prefix to log messages.
NOTE: Name = azureTenantId
         UsageType = Session
         Type = String
         Value =
         Default Value =
         Group = Session
         Description = specifies the tenant ID for Azure storage.
...
```

Additional Information

For information about the session properties, see “Session Options” on page 645. The GETSESSOPT function enables you to get the value of a single property. See “GETSESSOPT Function” in *SAS Functions and CALL Routines: Reference*.

Example 6: Display a Session Property Value

Program

This example displays the current TIMEOUT property value for session Casauto:

```
%put TIMEOUT for session CASAUTO: %sysfunc(getsessopt(casauto,
timeout)) seconds.;
```

SAS Log

The following is written to the SAS log:

```
TIMEOUT for session CASAUTO: 60 seconds.
```

Example 7: Change a Session Property Value

Program

This example changes the time-out for session Casauto to 60 minutes:

```
cas casauto sessopts=(timeout=3600);
```

Additional Information

For information about the session properties that you can change, see "Session Options" on page 645. To change one or more properties for all of your generated SAS Cloud Analytic Services sessions, specify the property settings in the CASSESSOPTS= option in an OPTIONS statement. See "CASSESSOPTS= System Option" in *SAS System Options: Reference*.

Example 8: Disconnect from a Session

Before You Disconnect from Your Session

If no other client connection exists and no actions are running when you disconnect, you must reconnect to the session before the connection time-out expires (60 seconds by default). Otherwise, the session is automatically terminated. Before you disconnect from a session, be sure to set session option TIMEOUT= for that session to an appropriate value. Use the RECONNECT option to reconnect to the session. See "Example 9: Connect to an Existing Session" on page 730.

Program

This example sets the time-out for session Casauto to 90 minutes, and then, if the time-out was successfully set, disconnects from session Casauto:

```
cas casauto sessopts=(timeout=5400);
%if &CASSTMTERR eq 0 %then %do;
   cas casauto disconnect;
%end;
```

SAS Log

The following notes are written to the SAS log:

```
NOTE: The CAS server request to update one or more session
      options for session CASAUTO completed.
NOTE: Request to DISCONNECT completed for session CASAUTO.
```

Disconnecting from the Default Session

If you disconnect from the default session, you must do one of the following in order to access the server:

- Reconnect to the default session as shown in the following example:

  ```
  cas casauto reconnect;
  ```

- Issue a CAS statement to create a new default session. When you create a new session, system option SESSREF= is automatically set to reference the new session. See “Example 1: Start a Session” on page 723.
- If you have one or more additional sessions available on your client that are connected to the server, set system option SESSREF= to one of the other connected sessions as shown in the following example:

  ```
  options sessref=session-reference;
  ```

  See “SESSREF= System Option” in *SAS System Options: Reference*.

Example 9: Connect to an Existing Session

Program: Reconnect to an Existing Session Using the Session Name

If you want to reconnect to a session that you started or connected to previously in your current SAS session, you must specify the session name to reconnect. This example reconnects to session Casauto, which was started previously in the current SAS session:

```
cas casauto reconnect;
```

SAS Log

The following note is written to the SAS log:

```
NOTE: Request to RECONNECT completed for session CASAUTO.
```

Program: Connect to an Existing Session Using the Session Name and UUID

If you want to connect to a session that you started in a different SAS session, you must specify the session name and UUID to connect. If you do not know the UUID of the session, use the LISTSESSIONS= option as described in "Example 4: List Information about All of Your Sessions" on page 727. This example uses session name Mysess and the UUID option to connect to session Mysess, which was started in a different SAS session.

```
cas mysess uuid="ca683ddf-fe18-3c48-a04e-45718220976d";
```

SAS Log

The following notes are written to the SAS log.

```
NOTE: The session MYSESS connected successfully to Cloud Analytic
      Services sas-cas-server-default-client using port 5570. The UUID is
      ca683ddf-fe18-3c48-a04e-45718220976d. The user is sasdemo and the
      default CASLIB is CASUSER(sasdemo).
NOTE: The SAS option SESSREF was updated with the value MYSESS.
NOTE: The SAS macro _SESSREF_ was updated with the value MYSESS.
NOTE: The session is using nnn workers.
```

Example 10: Start a Session on a Personal CAS Server

This example starts a session on a personal CAS server, sets system option CASHOST to the value of macro variable _CASHOST_PERSONAL_, and then executes a CAS statement:

```
options cashost=&_cashost_personal_;
cas mysess_personal;
```

Alternatively, specify macro variable reference &_CASHOST_PERSONAL_ in the HOST= option in your CAS statement:

```
cas mysess_personal host=&_cashost_personal_;
```

In either case, a connection with the personal CAS server is established.

```
NOTE: The session MYSESS_PERSONAL connected successfully to Cloud Analytic Services
      localhost using port 5570. The UUID is session-UUID.
      The user is sasdemo and the active caslib is CASUSER(sasdemo).
NOTE: The SAS option SESSREF was updated with the value MYSESS_PERSONAL.
NOTE: The SAS macro _SESSREF_ was updated with the value MYSESS_PERSONAL.
NOTE: The session is using 0 workers.
```

Additional Information

If a personal CAS server is not configured, macro variable _CASHOST_PERSONAL_ is not created. You can use macro function %SYMEXIST to determine whether it exists as follows:

```
%put %symexist(_cashost_personal_);
```

Do not include the ampersand (&) in the macro variable name. %SYMEXIST returns 1 if macro variable _CASHOST_PERSONAL_ exists, 0 otherwise. You can use %SYMEXIST in your SAS code to establish a connection in either case as shown in the following example:

```
%if %symexist(_cashost_personal_) %then %do;
   /* Connect to the personal CAS server. */
   %put NOTE: Connecting to personal CAS server &_cashost_personal_..;
   cas mysess host=&_cashost_personal_;
%end;
%else %do;
   /* Connect to the shared CAS server. */
   %put NOTE: Connecting to shared CAS server &_cashost_shared_..;
   cas mysess host=&_cashost_shared_;
%end;
```

See "%SYMEXIST Macro Function" in *SAS Macro Language: Reference*.

See Also

- "Personal CAS Server" in *SAS Viya Platform: SAS Cloud Analytic Services*
- "_CASHOST_PERSONAL_ Macro Variable" on page 635

Example 11: Switch between a Personal CAS Server and the Shared CAS Server

This example switches between the personal CAS server and the shared CAS server, creates a session on each server, and then uses the appropriate session to access the desired server.

```
/* Create a session on the shared CAS server. */
```

```
cas mysess_shared host=&_cashost_shared_;

/* Create a session on the personal CAS server. */
cas mysess_personal host=&_cashost_personal_;
```

To see information about the sessions, list all of the sessions in your SAS client:

```
cas _all_ list;
```

```
NOTE: Session MYSESS_SHARED is ACTIVE using port 5570 and host sas-cas-server-
default-client for user
      sasdemo. The session UUID is session-UUID.
NOTE: Session MYSESS_PERSONAL is ACTIVE using port 5570 and host localhost for user
      sasdemo. The session UUID is session-UUID.
```

To switch the servers in a CAS procedure step, for example, use the SESSION statement to specify the desired server as shown in the following example:

```
/* List the nodes on the shared and personal servers. */
proc cas;
   session mysess_shared;
   builtins.listnodes;
run;
   session mysess_personal;
   builtins.listnodes;
run;
quit;
```

See Also

- "Personal CAS Server" in *SAS Viya Platform: SAS Cloud Analytic Services*
- "_CASHOST_PERSONAL_ Macro Variable" on page 635
- "_CASHOST_SHARED_ Macro Variable" on page 636
- *SAS Cloud Analytic Services: CASL Reference* for information about the CAS procedure.
- *SAS Viya Platform: System Programming Guide* for information about the Builtins action set.

Example 12: Terminate a Session

Program

This example terminates session Casauto:

```
cas casauto terminate;
```

SAS Log

The following notes are written to the SAS log:

```
NOTE: Deletion of the session CASAUTO was successful.
NOTE: The default CAS session CASAUTO identified by SAS option SESSREF=
      was terminated. Use the OPTIONS statement to set the SESSREF= option
      to an active session.
NOTE: Request to TERMINATE completed for session CASAUTO.
```

Terminating the Default Session

If you terminate the default session, a note similar to the following is written to the SAS log.

```
NOTE: The default CAS session CASAUTO identified by SAS option SESSREF=
      was terminated. Use the OPTIONS statement to set the SESSREF= option
      to an active session.
```

In that case, to connect to the server, do one of the following:

- Issue a CAS statement to create a new default session. When you create a new session, system option SESSREF= is automatically set to reference the new session. See "Example 1: Start a Session" on page 723.
- If you have one or more additional sessions available on your client that are connected to the server, set system option SESSREF= to one of the other connected sessions as shown in the following example:

  ```
  options sessref=session-reference;
  ```

  See "SESSREF= System Option" in *SAS System Options: Reference*.

Terminating a Disconnected Session

If you terminate a session that you disconnected using DISCONNECT, the session is removed from your client session list, but it continues to run on the server until its time-out expires. In that case, to terminate the session on the server, do the following:

1. Use LISTSESSIONS in the active session to get the disconnected session's name and UUID.

   ```
   cas active-session-reference listsessions;
   ```

2. Connect to the disconnected session using its name and UUID.

   ```
   cas disconnected-session-reference uuid="disconnected-session-uuid";
   ```

3 Terminate the previously disconnected session.

```
cas disconnected-session-reference terminate;
```

Example 13: Create a Format Library and Save It to a SASHDAT File

Program

This example creates format library MyFmtLib, populates it with character format $Codes, numeric format Response, and numeric range format MPGrating, and then stores MyFmtLib in file myFmtLib.sashdat in caslib CASUSER. All of the formats use the default locale. For an example of how to create locale-specific formats, see "Add Locale-Specific Format Libraries" in *SAS Cloud Analytic Services: User-Defined Formats*.

```
/* options cashost="sas-cas-server-default-client" casport=5570; */
cas casauto;                                          /* 1 */

proc format sessref=casauto casfmtlib="myFmtLib";  /* 2 */
   value $codes
      "A" = "Alpha"
      "B" = "Beta"
      "C" = "Charlie"
      "D" = "Delta";
run;

proc format sessref=casauto casfmtlib="myFmtLib";  /* 3 */
   value response
      1 = "Yes"
      2 = "No"
      3 = "Undecided"
      4 = "No response";
run;

proc format sessref=casauto casfmtlib="myFmtLib";  /* 4 */
   value MPGrating
      34  -  HIGH = "Excellent"
      24  -< 34   = "Good"
      19  -< 24   = "Fair"
      LOW -< 19   = "Poor";
run;

cas casauto savefmtlib fmtlibname=myFmtLib         /* 5 */
   table="myFmtLib.sashdat" caslib=casuser replace;
```

1 If you have not already done so, start your CAS session.

2 Use the FORMAT procedure to create character format $Codes. Option CASFMTLIB= specifies the name of the format library in CAS. The FORMAT procedure stores character format $Codes in the default SAS format catalog Work.Formats and in the CAS in-memory format library MyFmtLib.

Note: Use the FORMAT procedure option LIBRARY= to store the format in a different catalog in SAS. For an example, see "Add and Save a User-Defined Format" in *SAS Cloud Analytic Services: User-Defined Formats*.

Note: Use the FORMAT procedure to add existing SAS format catalogs to your CAS session. See "Migrate User-Defined Formats from SAS to CAS" in *SAS Cloud Analytic Services: User-Defined Formats*.

3 Create numeric format Response. Numeric format Response is stored in the SAS format catalog Work.Formats and in the CAS in-memory format library MyFmtLib.

4 Create range format MPGrating. Numeric range format MPGrating is stored in the SAS format catalog Work.Formats and in the CAS in-memory format library MyFmtLib.

5 Save the CAS in-memory format library MyFmtLib to file myFmtLib.sashdat in caslib CASUSER for later use. The TABLE= option specifies the name of the file on disk. You can specify the name with or without the .sashdat file extension. If TABLE= is not specified, the name specified in FMTLIBNAME with the .sashdat file extension is used as the file name. The REPLACE option replaces the table file if it already exists in CASUSER. If the table already exists and you do not specify REPLACE, an error occurs.

SAS Log

```
1          cas casauto;
   ...
2
3          proc format sessref=casauto casfmtlib="myFmtLib";
NOTE: Both CAS based formats and catalog-based formats will be written.
      The CAS based formats will be written to the session CASAUTO.
4             value $codes
5                "A" = "Alpha"
6                "B" = "Beta"
7                "C" = "Charlie"
8                "D" = "Delta";
NOTE: Format library MYFMTLIB added. Format search update using
      parameter APPEND completed.
NOTE: Format $CODES has been output.
9          run;
   ...
10
11         proc format sessref=casauto casfmtlib="myFmtLib";
NOTE: Both CAS based formats and catalog-based formats will be written.
      The CAS based formats will be written to the session CASAUTO.
12            value response
13               1 = "Yes"
14               2 = "No"
15               3 = "Undecided"
16               4 = "No response";
NOTE: Format RESPONSE has been output.
17         run;
   ...
18
19         proc format sessref=casauto casfmtlib="myFmtLib";
NOTE: Both CAS based formats and catalog-based formats will be written.
      The CAS based formats will be written to the session CASAUTO.
20            value MPGrating
21               34  -  HIGH = "Excellent"
22               24  -< 34   = "Good"
23               19  -< 24   = "Fair"
24               LOW -< 19   = "Poor";
NOTE: Format MPGRATING has been output.
25         run;
   ...
26
27         cas casauto savefmtlib fmtlibname=myFmtLib
28            table="myFmtLib.sashdat" caslib=casuser replace;
NOTE: Cloud Analytic Services saved the file myFmtLib to CASUSER in
      caslib CASUSER(sasdemo).
NOTE: The format library MYFMTLIB save to myFmtLib.sashdat
      completed successfully.
NOTE: Request to SAVEFMTLIB MYFMTLIB completed for session CASAUTO.
```

Example 14: Add a Format Library That Is Stored in a SASHDAT File

Program

This example adds to session Casauto format library MyFmtLib, which was stored in file myFmtLib.sashdat in "Example 13: Create a Format Library and Save It to a SASHDAT File" on page 735.

```
/* options cashost="sas-cas-server-default-client" casport=5570; */
cas casauto;                                    /* 1 */

cas casauto addfmtlib fmtlibname=myFmtLib      /* 2 */
   table="myFmtLib.sashdat" caslib=casuser replacefmtlib;

cas casauto listformats fmtlibname=myFmtLib;  /* 3 */
```

1 If you have not already done so, start your CAS session.

2 Use the FMTLIBNAME option to specify a name for the format library in session Casauto. The TABLE= option specifies the name of the format library file in caslib CASUSER, which is myFmtLib.sashdat. The name can be specified with or without the .sashdat file extension. If you do not specify TABLE=, the file name is assumed to be the name specified in FMTLIBNAME= with the .sashdat file extension. The REPLACEFMTLIB option replaces format library MyFmtLib if it already exists in session Casauto. If MyFmtLib already exists in session Casauto and you do not specify REPLACEFMTLIB, an error occurs.

   By default, format library MyFmtLib is placed at the end of the format library search list (APPEND). Use the POSITION= option to specify a different position in the search list.

3 Use LISTFORMATS to verify that format library MyFmtLib was added to session Casauto.

SAS Log

```
1          cas casauto;
   ...
2
3          cas casauto addfmtlib fmtlibname=myFmtLib
4             table="myFmtLib.sashdat" caslib=casuser replacefmtlib;
NOTE: Format library MYFMTLIB added. Format search update using
      parameter APPEND completed.
NOTE: Request to ADDFMTLIB MYFMTLIB completed for session CASAUTO.
5
6          cas casauto listformats fmtlibname=myFmtLib;
NOTE: Fmtlib = MYFMTLIB
         Scope = Session
         Fmtsearch = YES
NOTE: Request to LISTFORMAT completed for session CASAUTO.
```

Example 15: List the Format Libraries That Are Available to a Session

Program

This example lists all of the format libraries that are available to session Casauto:

```
cas casauto listformats;
```

SAS Log

Here is an example of what is written to the SAS log.

```
     cas casauto listformats;
NOTE: Fmtlib = COMPANYFORMATS
         Scope = Global
         Fmtsearch = NA
NOTE: Fmtlib = DEPTFORMATS
         Scope = Global
         Fmtsearch = NA
NOTE: Fmtlib = MYFMTLIB
         Scope = Session
         Fmtsearch = YES
NOTE: Request to LISTFORMAT completed for session CASAUTO.
```

Example 16: List the Session-Scope Formats in a Session

Program

This example lists the session-scope format libraries and their formats that are available in session Casauto:

```
cas casauto listformats scope=session members;
```

SAS Log

Here is an example of what is written to the SAS log.

```
     cas casauto listformats scope=session members;
NOTE: Fmtlib = MYFMTLIB
         Scope = Session
         Fmtsearch = YES
         Format = $codes
         Format = mpgrating
         Format = response
NOTE: Request to LISTFORMAT completed for session CASAUTO.
```

Example 17: Add Global Format Libraries to a Session's Format Search List

Program

This example adds global format libraries CompanyFormats and DeptFormats to the format search list for session Casauto. This example assumes that format libraries CompanyFormats and DeptFormats exist in global scope, and that format library MyFmtLib exists in session scope. (See "Example 14: Add a Format Library That Is Stored in a SASHDAT File" on page 738.)

```
cas casauto listformats;                    /* 1 */
cas casauto                                 /* 2 */
   fmtsearch=(companyFormats deptFormats)
   position=insert;
cas casauto listfmtsearch;                  /* 3 */
cas casauto listformats;                    /* 4 */
```

1 Use LISTFORMATS to list all of the format libraries that are available. See "SAS Log" on page 741. A note is written to the SAS log for each format library. Format libraries CompanyFormats and DeptFormats are available in global scope, but their formats currently are not available to session Casauto.

2 Insert format libraries CompanyFormats and DeptFormats at the beginning of the search list. In addition to adding CompanyFormats and DeptFormats to the format search list, this creates format libraries CompanyFormats and DeptFormats in session scope.

3 Use LISTFMTSEARCH to verify that CompanyFormats and DeptFormats are now included in the format search list at the beginning.

4 Use LISTFORMATS to verify that format libraries CompanyFormats and DeptFormats are now available in session scope. Because format libraries CompanyFormats and DeptFormats are included in the format search list and are now in session scope, their formats are available for use in session Casauto.

SAS Log

```
4    cas casauto listformats;
NOTE: Fmtlib = COMPANYFORMATS
         Scope = Global
         Fmtsearch = NA
NOTE: Fmtlib = DEPTFORMATS
         Scope = Global
         Fmtsearch = NA
NOTE: Fmtlib = MYFMTLIB
         Scope = Session
         Fmtsearch = YES
NOTE: Request to LISTFORMAT completed for session CASAUTO.
4  !                                /* 1 */
5    cas casauto                    /* 2 */
6      fmtsearch=(companyFormats deptFormats)
7      position=insert;
NOTE: Request to FMTSEARCH completed for session CASAUTO.
8    cas casauto listfmtsearch;
NOTE: FmtLibName = COMPANYFORMATS
         Scope = Both
NOTE: FmtLibName = DEPTFORMATS
         Scope = Both
NOTE: FmtLibName = MYFMTLIB
         Scope = Session
NOTE: Request to LISTFMTSEARCH completed for session CASAUTO.
8  !                                /* 3 */
9    cas casauto listformats;
NOTE: Fmtlib = COMPANYFORMATS
         Scope = Global
         Fmtsearch = NA
NOTE: Fmtlib = COMPANYFORMATS
         Scope = Session
         Fmtsearch = YES
NOTE: Fmtlib = DEPTFORMATS
         Scope = Global
         Fmtsearch = NA
NOTE: Fmtlib = DEPTFORMATS
         Scope = Session
         Fmtsearch = YES
NOTE: Fmtlib = MYFMTLIB
         Scope = Session
         Fmtsearch = YES
NOTE: Request to LISTFORMAT completed for session CASAUTO.
```

Example 18: Change the Format Library Search List

Program

This example changes the format library search list shown in “Example 17: Add Global Format Libraries to a Session’s Format Search List” on page 740 to CompanyFormats, MyFmtLib, DeptFormats so that MyFmtLib can override formats in DeptFormats.

```
cas casauto
   fmtsearch=(companyformats, myFmtLib, deptformats)  /* 1 */
   position=replace;
```

```
cas casauto listfmtsearch;                                    /* 2 */
```

1 Change the order to CompanyFormats, MyFmtLib, DeptFormats. POSITION=REPLACE replaces the current search list with the new search list.

2 Use LISTFMTSEARCH to verify the new search list.

SAS Log

```
     cas casauto
              fmtsearch=(companyformats, myFmtLib, deptformats)
              position=replace;
NOTE: Request to FMTSEARCH completed for session CASAUTO.

     cas casauto listfmtsearch;
NOTE: FmtLibName = COMPANYFORMATS
         Scope = Session
NOTE: FmtLibName = MYFMTLIB
         Scope = Session
NOTE: FmtLibName = DEPTFORMATS
         Scope = Session
NOTE: Request to LISTFMTSEARCH completed for session CASAUTO.
```

Example 19: List the Ranges in a Format

Program

This example lists the ranges in formats Response and MPGrating in format library MyFmtLib, which was added to session Casauto in "Example 14: Add a Format Library That Is Stored in a SASHDAT File" on page 738.

```
cas casauto listfmtranges fmtname=response;
cas casauto listfmtranges fmtname=MPGrating;
```

SAS Log

```
cas casauto listfmtranges fmtname=response;
Format Name         Range
RESPONSE            1=Yes
                    2=No
                    3=Undecided
                    4=No response

NOTE: Request to LISTFMTRANGES RESPONSE completed for session CASAUTO.
cas casauto listfmtranges fmtname=MPGrating;
Format Name         Range
MPGRATING           LOW-<19=Poor
                    19-<24=Fair
                    24-<34=Good
                    34-HIGH=Excellent
```

Example 20: Promote a Format Library to Global Scope

Note: The CAS server can be configured to require administrative Superuser authority to promote a user-defined format library to global scope. By default, it is not required. If administrative Superuser authority is required and you are a CAS administrator, you must enable an automatic request for escalation to Superuser before executing the CAS statement that promotes the format library. See "CAS_ADMIN_GLOBAL_UDFMTS Macro Variable" on page 639. If you do not have administrative authority, you must contact your CAS administrator to promote your user-defined format library.

This example promotes a format library that is in session scope to global scope.

Program

Promote format library MyFmtLib, which was added in "Example 14: Add a Format Library That Is Stored in a SASHDAT File" on page 738, to global scope.

```
cas casauto promotefmtlib fmtlibname=myFmtLib;  /* 1 */

cas casauto listformats;                        /* 2 */
```

1 Promote format library MyFmtLib to global scope.

Note: If the format library that you are promoting already exists in global scope, specify REPLACE to replace it. Otherwise, an error occurs.

2 Use LISTFORMATS to verify that format library MyFmtLib now exists in global scope.

SAS Log

```
     cas casauto promotefmtlib fmtlibname=myFmtLib;
NOTE: Request to PROMOTEFMTLIB MYFMTLIB completed for
      session CASAUTO.

     cas casauto listformats;
NOTE: Fmtlib = MYFMTLIB
        Scope = Global
        Fmtsearch = NA
NOTE: Fmtlib = MYFMTLIB
        Scope = Session
        Fmtsearch = YES

NOTE: Request to LISTFORMAT completed for session CASAUTO.
```

Example 21: Drop a Format Library from Session Scope

Note: The CAS server can be configured to require administrative Superuser authority to drop a global user-defined format library. By default, it is not required. If administrative Superuser authority is required and you are a CAS administrator, you must enable an automatic request for escalation to Superuser before executing the CAS statement that drops the format library. See "CAS_ADMIN_GLOBAL_UDFMTS Macro Variable" on page 639. If you do not have administrative authority, you must contact your CAS administrator to drop your user-defined format library from global scope.

This example drops a format library that is in global scope.

Program

Drop format library MyFmtLib, which was promoted in "Example 20: Promote a Format Library to Global Scope" on page 743, from session scope.

```
cas casauto dropfmtlib fmtlibname=myFmtLib;  /* 1 */

cas casauto listformats;                     /* 2 */
```

1 Drop format library MyFmtLib from session scope. Format library MyFmtLib remains in the Casauto session format library search list.

2 Use LISTFORMATS to verify that format library MyFmtLib is no longer in session scope.

SAS Log

```
     cas casauto dropfmtlib fmtlibname=myFmtLib;
NOTE: Request to DROPFMTLIB MYFMTLIB completed for
      session CASAUTO.

     cas casauto listformats;
NOTE: Fmtlib = MYFMTLIB
         Scope = Global
         Fmtsearch = NA
NOTE: Request to LISTFORMAT completed for session CASAUTO.
```

See Also

SAS Cloud Analytic Services: User-Defined Formats

CASLIB Statement

Adds and manages caslibs in a SAS Cloud Analytic Services session.

Restriction: You can add caslibs only if you are authorized to do so. See "Adjust Caslib Management Privileges" in *SAS Environment Manager: User's Guide*.

See: For more examples of accessing data, see *SAS Cloud Analytic Services: User's Guide*.

For conceptual information about caslibs, see "Caslibs" in *SAS Cloud Analytic Services: Fundamentals*.

Data connectors are the software that is used with a caslib to read server-based data sources like databases and Hive. For more information about data connectors, see "Data Connectors" on page 167. For platform file types see "Platform File Types" on page 153.

Syntax

Form 1: **CASLIB** *caslib-reference-name* < SESSREF=*session-reference* > <DATASOURCE=(SRCTYPE="*type*" <*data-source-options*>) > < PATH="*directory-path*" ><*options*>;

Form 2: **CASLIB** *caslib-reference-name* LIST < SESSREF=*session-reference* > ;

Form 3: **CASLIB** _ALL_ ASSIGN | _ALL_ LIST < SESSREF=*session-reference* >;

Required Argument

caslib-reference-name

specifies the name of the caslib. Names of session caslibs must be unique within the session. Names of global caslibs must be unique across all sessions within a server.

Range 1 to 256 characters

Tips Caslib names are case insensitive. If a session-scope caslib and a global-scope caslib have the same name, the session caslib is searched first.

ALL is a valid name with the LIST option and the ASSIGN option.

A SAS library that is associated with a caslib can reference only library member names that conform to SAS library naming conventions.For naming rules see "Words and Names" in *SAS Programmer's Guide: Essentials*.

Optional Arguments

ASSIGN

used with the _ALL_ option to assign SAS librefs for existing caslibs so that they are visible in the Libraries tree in SAS Data and AI Studio (known as SAS Studio prior to 2026.06).

Restriction The CASLIB _ALL_ ASSIGN statement assigns a libref only for caslibs that follow SAS libref naming rules. For example, if the caslib has more than eight characters, then no libref is assigned for that caslib.

Requirement The ASSIGN option must be used with the _ALL_ option.

Tip When caslibs are dropped, the corresponding SAS library is not automatically cleared.

Examples The following statement assigns SAS librefs for all existing caslibs:

```
caslib _all_ assign;
```

The following statement assigns SAS librefs for all existing caslibs in the session named MySess:

```
caslib _all_ assign sessref=mysess;
```

ALL

specifies that the ASSIGN, or LIST argument applies to all currently added caslibs.

Example "Program 2: List All Caslibs" on page 756

CREATEDIRECTORY

creates a subdirectory in the caslib's data source. The path, up to the last directory, must already exist.

Alias CREATEDIR

Restriction Directory creation is available only for global caslibs.

Example The following CASLIB statement creates a new directory named Programs in the **/examples/caslibs/** directory:

```
caslib TestDir path="/examples/caslibs/programs"
              datasource=(srctype="path") createdir global;
```

DATASOURCE= (SRCTYPE="*type*", <*data-source-options* >, <ENCRYPTIONPASSWORD="*string*">, <ENCRYPTIONDOMAIN="*string*">)
specifies data source options to use when connecting to a data source. The SRCTYPE="*type*" option specifies the data source type. Data sources can be either databases or path-based. The *data-source-options* syntax depends on the data source.

ENCRYPTIONDOMAIN="*string*"
specifies the name for a collection of data that is stored with a common encryption password.

Restrictions The ENCRYPTIONPASSWORD= option can be specified only with the DATASOURCE= option.

When you add a path-based caslib that supports the encryptionDomain and encryptionPassword parameters, you can specify one or the other. Use the encryptionDomain parameter to retrieve credentials automatically from an encryption domain that is managed with SAS Environment Manager.

ENCRYPTIONPASSWORD="*string*"
specifies a password for encrypting or decrypting stored data.

Restrictions The ENCRYPTIONPASSWORD= option can be specified only with the DATASOURCE= option.

When you add a path-based caslib that supports the encryptionDomain and encryptionPassword parameters, you can specify one or the other. Use the encryptionDomain parameter to retrieve credentials automatically from an encryption domain that is managed with SAS Environment Manager.

Note You can add a caslib without specifying the DATASOURCE option. The following example adds a path caslib using the PATH option: `caslib mylib path="/local/data";`. This CASLIB statement is functionally equivalent to the following CASLIB statement: `caslib mylib path="/local/data" datasource=(srctype="path");`

See "Path-Based Data Source Types and Options" on page 751, "Data Connectors", "Platform Data Sources"

Examples "Example 1: Add a Global Caslib" on page 754

"Example 5: Encrypt Tables in a Caslib" on page 762

DESCRIPTION="*description*"
specifies a description of the data source.

Alias DESC=

DROP
drops a session scope caslib.

Alias CLEAR

Example
```
caslib caslibA drop;
```
```
NOTE: Cloud Analytic Services removed the caslib 'CASLIBA'.
NOTE: Action to DROP caslib CASLIBA completed for session CASAUTO.
```

GLOBAL
adds the caslib so that it has global scope. For a global-scope caslib, you can set access controls so that it is accessible from all sessions and can be a way to share data. Other connections to the server that get their own sessions have access to the caslib, subject to access controls. If you do not specify GLOBAL, the caslib is created with session scope. You must also grant access to the caslib in the CAS Server Monitor. You can add caslibs only if you are authorized to do so. See "Adjust Caslib Management Privileges" in *SAS Environment Manager: User's Guide*.

See For information about caslibs scope, see "Caslib Scope" in *SAS Cloud Analytic Services: Fundamentals*.

LIBREF=
creates a libref and associates it with a caslib. The libref can then be used to access data in SAS Cloud Analytic Services.

Example The following CASLIB statement creates the caslib Casdata and binds it to the libref Mycas.
```
caslib casdata path='/cas/mycasdata/' libref=mycas;
```

LIST
displays caslib names and their specifications. To display all caslibs, specify _ALL_ for the *caslib-reference-name* .

Tip Specify *caslib-reference-name* to list the settings for a single caslib. Specify _ALL_ to list all caslibs and their caslib settings. The specifications displayed by the LIST _ALL_ option are a subset. For full specification information, use the LIST option with a specific caslib specified.

Example "Example 2: List Caslib Settings" on page 755

NOTACTIVE
specifies that the caslib being added does not become the active caslib for the session.

PATH="*directory-path*"
specifies the fully qualified path to a directory to use as a data source.

Example The following CASLIB statement adds a caslib that accesses a path-based directory:
```
caslib mylib path="/local/data" description="Local data" ;
```

SESSION
adds the caslib so that it is session-scope. Other connections to the server that get their own sessions do not have access to the caslib. The lifetime of the

caslib is the lifetime of the session. When you add caslibs, SESSION is the default.

Alias LOCAL

See "Caslibs Scope" on page 750

SESSREF=*session-reference*
specifies the name of the session to associate the caslib with. By default, the most recently started session is used.

SUBDIRS
specifies that subdirectories of the specified PATH= directory can be accessed with the caslib.

TABLEREDISTUPPOLICY = NOREDIST | REBALANCE
specifies how tables loaded in the caslib are redistributed when a server scale-up occurs.

NOREDIST
specifies that tables loaded in the caslib will not be redistributed when a server scale-up occurs

REBALANCE
specifies that tables loaded in the caslib will be balanced across the available nodes or pods when a server scale-up occurs.

Default NOREDIST

Details

What are Caslibs?

Caslibs are the mechanism for accessing data with SAS Cloud Analytic Services (CAS). They provide a volatile, in-memory space to hold tables, access controls, and data source information. Caslibs provide a way to organize in-memory tables and an associated data source. They also provide a way to apply access controls to data. A table within a caslib is a temporary, in-memory copy of the original data. All operations in SAS Cloud Analytic Services that use data are performed on tables within a caslib. Use the SAVE statement in the CASUTIL procedure to permanently save tables to a data source. For more information about the CASUTIL procedure, see Chapter 3, "CASUTIL Procedure," on page 57.

What Can I Do with the CASLIB Statement

You can do the following tasks with the CASLIB statement:

- add a caslib with access to files from the data source and access to in-memory tables that are read from the data source
- specify the options to use when connecting to a data source
- list the caslibs that are available to your session

- view the caslib settings for one or more caslibs

For conceptual information about caslibs, see "Caslibs" in *SAS Cloud Analytic Services: Fundamentals*.

Caslibs Scope

A caslib can have session scope or global scope. Session-scope caslibs make data available to the session that added the caslib. By default, when you add a caslib with the CASLIB statement, the caslib is session scope. You cannot change the scope of a caslib once it has been added.

Global-scope caslibs make data available to all sessions. By default, your personal caslib is global scope, but restricted to the sessions that you start. You can promote tables to global-scope caslibs only. Global-scope caslibs are useful for data sources that all programmers need to access or in cases when you want to share data with other users. An administrator might restrict your ability to add a global-scope caslib. Use the GLOBAL CASLIB statement option to add a global-scope caslib. You can add caslibs only if you are authorized to do so. See "Adjust Caslib Management Privileges" in *SAS Environment Manager: User's Guide*.

Session-scope caslibs are useful for ad hoc data analysis and in cases where you do not want to share data with other users.

For more information about caslib's scope, see "Caslib Scope" in *SAS Cloud Analytic Services: Fundamentals*.

You can also hide caslibs. A hidden caslib is omitted from most lists of caslibs. Tables in a hidden caslib are omitted from most lists of tables. See "Adjust Caslib Management Privileges" in *SAS Environment Manager: User's Guide*.

The Active Caslib for a CAS Session

When you start a session, your personal caslib is added by default. Initially, it is the active caslib if your server is configured with personal caslibs. If not, then the first defined global caslib is the active caslib. The active caslib can change as caslibs are added and dropped. When you use the CASLIB statement to add a caslib, that caslib becomes the active caslib. The active caslib is used as the default data source if you do not override it. You can override the active caslib in the CASUTIL procedure or as a data set option for a CAS engine libref. Because the active caslib is used as a default data source, only one caslib can be active at a time. If you use another CASLIB statement to add a caslib, the previous caslib becomes inactive, and the new caslib becomes active. To add a caslib without making it the active caslib, use the NOTACTIVE CASLIB statement option.

For a CAS session, session option CASLIB is set to the name of the active caslib. You can use the GETSESSOPT function to get the current value of session option CASLIB. For example, the following %PUT statement writes a note to the SAS log that shows the active caslib for session Casauto:

```
%put NOTE: Session Casauto active caslib is
 %sysfunc(getsessopt(casauto, caslib)).;
```

```
NOTE: Session Casauto active caslib is CASUSER(casdemo).
```

You can change the active caslib for a CAS session by using the CASLIB= session option in a CAS statement. For example, the following CAS statement sets the active caslib for session Casauto to CaslibB:

```
cas casauto sessopts=(caslib=caslibB);
```

For information about the CASLIB= session option, see "CASLIB= Session Option" on page 654.

Path-Based Data Source Types and Options

The following table shows the syntax, supported file types (if applicable), and an example for path-based data sources. For the data source options syntax for databases such as Hadoop and Oracle, see "Data Connectors" on page 167. In the following table, for details about parameters like fileType, see "Table Action Set" in *SAS Viya Platform: System Programming Guide*. For links to more information about these platforms and file types, see Data Sources Listed Alphabetically.

Table 14.3 *Path-Based Data Source Types and Options*

SRCTYPE= Type	Option Syntax	Example	Supported File Types

ADLS
 	
(Azure Data Lake Storage Gen2)
	DATASOURCE=(SRCTYPE="ADLS" ACCOUNTNAME="*string* " FILESYSTEM="*file-system-name*" APPLICATIONID="*string*" < DNSSUFFIX="*network-storage-host-name*" > < TIMEOUT=*seconds* >) <*caslib-statement-options* > SUBDIRS=TRUE | FALSE ;	`cas casauto sessopts=(azureTenantId='b1c24e5c-3625-4593-a430-9552373a0c2f');`
		
		`caslib azds`
		`datasource=(`
		`srctype="adls"`
		`accountname="myacount"`
		`filesystem="dir"`
		`applicationId="b1fc955d5c-e0e2-45b3-a3cc-a1cf54120f"`
		`)`
		`subdirs;`
			■ CSV
			■ Multiple-CSVs
			■ ORC [6]
			■ PARQUET [4]

DNFS	DATASOURCE=(SRCTYPE=DNFS, < ENCRYPTIONDOMAIN="*string*" >) PATH="*file-path*";	`caslib mycsvs`
		`datasource=(srctype="dnfs")`
		`path="/data/Mycsvfiles/";`
			■ AUDIO [5]
			■ CSV [3]
			■ Multiple CSVs
			■ Compressed CSVs (.GZ[8] and .ZIP)
			■ DOCUMENT [5]
			■ IMAGE [5]
			■ PARQUET [4]
			■ SASHDAT [7]
			■ VIDEO [5]

SRCTYPE= Type	Option Syntax	Example	Supported File Types
Google Cloud Storage	DATASOURCE=(SRCTYPE="GCS" BUCKET="*string*" GCSAUTHFILE="*string*" <*caslib-statement-options*>;	`caslib gcs datasource=( srctype="gcs" bucket="mybucket" gcsauthfile="myGCSauthfile" );`	■ CSV **Note:** Google Cloud Storage supports only single-file CSVs. ■ PARQUET
PATH	DATASOURCE=(SRCTYPE=PATH, < ENCRYPTIONDOMAIN="*string*" >) PATH="*file-path*";	`caslib mycsvs datasource=(srctype="path", encryptiondomain=*string* ) path="/data/Myxlsxfiles/";`	■ AUDIO [5] ■ CSV [3] ■ Multiple CSVs ■ Compressed CSVs (.GZ[8] and .ZIP) ■ DOCUMENT [5] ■ DTA [2] ■ IMAGE [5] ■ JMP [2, 5] ■ ORC [6] ■ PARQUET [4] ■ SASHDAT ■ SAS7BDAT (specify BASESAS for the fileType parameter) [1] ■ SAV (specify SPSS for the fileType parameter) [2, 5] ■ VIDEO [5] ■ XLS [2] ■ XLSX (specify EXCEL for the fileType parameter) [2]
Amazon S3	DATASOURCE=(SRCTYPE="S3" REGION="*region*"	`caslib mys3 datasource=(srctype="s3" accesskeyid="LASKDJFLKJE9EXAMPLE" secretaccesskey="kJflk/M8ofa/bPMPLE"`	■ AUDIO [5] ■ CSV [3]

SRCTYPE= Type	Option Syntax	Example	Supported File Types
	BUCKET="*name*" <*option(s)* >);	region="EU_Ireland" bucket="mybucket" objectpath="/asr/projB" usessl=true);	■ Compressed CSVs (.GZ8) ■ DOCUMENT [5] ■ IMAGE [5] ■ PARQUET [4] ■ SASHDAT ■ VIDEO [5] ■ XLS [2] ■ XLSX (FIle type XLSX is read-only. Specify EXCEL for the fileType parameter)[2]

1 Requires the SAS Data Connector to SAS Data Sets.

2 Requires the SAS Data Connector to PC Files.

3 The CSV file type is used to identify any delimited file. For example, to read a semicolon delimited file with a .txt suffix, you can specify CSV as the file type and specify the delimiter.

4 Apache Parquet files are supported for single-machine and distributed servers.

5 These file types are read-only.

6 Apache ORC files are supported for single-machine and distributed servers.

7 For a SASHDAT file in DNFS, treat the file as read-only unless you set the backing store to CASDISKCACHE. See "Controlling Use of CAS_DISK_CACHE" in *SAS Viya Platform: System Programming Guide*.

8 One CSV file per .GZ archive is supported. .GZ archives that contain multiple CSV files are not supported. The multiFile parameter must be set to FALSE when loading a CSV from a .GZ archive.

Troubleshooting

Here is a message that you might encounter. In most cases, messages can be experienced from any programming interface. For messages that are produced in SAS Data and AI Studio (known as SAS Studio prior to 2026.06) only, an indication is provided.

You Do Not Have Permission to Create a Caslib

ERROR: You do not have permission to create global caslibs or ERROR: You do not have permission to create session caslibs.

You can add caslibs only if you are authorized to do so. See "Adjust Caslib Management Privileges" in *SAS Environment Manager: User's Guide*.

Examples

Example 1: Add a Global Caslib

Program

The following example adds the caslibs Vapublic and Hadooplib. They each contain connection information to the data sources. After running this program, Hadooplib is the active caslib.

```
/*working example*/
caslib Hadooplib desc="Hadoop Caslib"
               datasource=(srctype="hadoop", /*2*/
               dataTransferMode="parallel",
               hadoopjarpath="/data/cdh54/sdm/lib",
               hadoopConfigDir="/data/cdh54/sdm/conf",
               username="hiveuser",
               server="hive01.example.com",
               schema="default") global;

/*If not already done, create session Casauto.*/
/*Specify a host and port that are valid for your site.*/
/*options cashost="cloud.example.com" casport=5570;*/
/*cas
casauto;*/

caslib Vapublic path="/vapublic"
                          datasource=(srctype="dnfs")
global ;        /*1*/

caslib Hadooplib desc="Hadoop Caslib"

datasource=(srctype="hadoop",                          /*2*/
               dataTransferMode="parallel",
               hadoopjarpath="Hadoo-jar-file-path",
               hadoopConfigDir="Hadoop-config-files-path",
               username="user-id",
               server="Hadoop-server-hostname",
               schema="schema-name") global;
```

1 The first CASLIB statement adds a global-scope caslib named Vapublic. The DATASOURCE= option and the PATH= option provide connection information to the Vapublic directory. The GLOBAL option enables you to promote tables to the caslib. You must also set the caslib to Global in CAS Server Monitor. Vapublic is now the active caslib.

2 The second CASLIB statement adds a global-scope caslib named Hadooplib, which provides access to a Hadoop database. The DATASOURCE= option

specifies the option to use when connecting to the database. The GLOBAL option enables you to promote tables to the caslib. Hadooplib is now the active caslib.

SAS Log

The notes in the SAS log verify that the caslibs Vapublic and Hadooplib were added. Note that Hadooplib is the active caslib, because it was added last.

```
 28         caslib Vapublic path="/vapublic"
 29                                  datasource=(srctype="dnfs") global ;
 NOTE: 'VAPUBLIC' is now the active caslib.
 NOTE: Cloud Analytic Services added the caslib 'VAPUBLIC'.
 NOTE: Action to ADD caslib VAPUBLIC completed for session CASAUTO.
 30
 31
 32         caslib Hadooplib desc="Hadoop Caslib"
 33
datasource=(srctype="hadoop",                                        /*2*/
 34                        dataTransferMode="parallel",
 35                        hadoopjarpath="Hadoo-jar-file-path",
 36                        hadoopConfigDir="Hadoop-config-files-path",
 37                        username="user-id",
 38                        server="Hadoop-server-hostname",
 39                        schema="schema-name") global;
 NOTE: 'HADOOPLIB' is now the active caslib.
 NOTE: Cloud Analytic Services added the caslib 'HADOOPLIB'.
 NOTE: Action to ADD caslib HADOOPLIB completed for session CASAUTO.
```

Additional Information

- For information about specifying Hadoop data source options, see "Hadoop Data Connector" on page 212.
- You can add caslibs only if you are authorized to do so. See "Adjust Caslib Management Privileges" in *SAS Environment Manager: User's Guide*.

Example 2: List Caslib Settings

You can list the caslib properties for one or more caslibs with the LIST option.

Program 1: List the Settings for a Specific Caslib

The CASLIB statement with the LIST option and a caslib name displays the caslib settings for a specific caslib.

```
caslib Hadooplib list;
```

SAS Log Showing Data Connector Settings

The settings for uid, server, hadoopConfigDir, hadoopJarPath, and schema appear only when listing the caslib explicitly. The settings that are displayed correspond to data connector options and are specific to the data connector.

```
NOTE: Session = CASAUTO Name = HADOOPLIB
         Type = hadoop
         Description = Hadoop Caslib
         Path =
         Definition = dataTransferMode = 'Parallel'
         Subdirs = No
         Local = No
         Active = Yes
         Personal = No
         Definition = uid = 'hiveuser'
         Definition = server = 'hive01.example.com'
         Definition = hadoopConfigDir = '/data/cdh54/sdm/conf'
         Definition = hadoopJarPath = '/data/cdh54/sdm/lib'
         Definition = schema = 'default'
```

Program 2: List All Caslibs

The CASLIB statement with the LIST option and the _ALL_ option displays all of the caslibs that are available and the caslib settings for each one.

```
caslib _all_ list;
```

SAS Log Showing Settings for All Caslibs

Caslibs 1 and 2 are the personal caslibs that were automatically created when the session started. Caslibs 3 and 4 were added explicitly in the code above. Caslib 5 was created by another session, but has global scope, so it is available to all sessions.

```
NOTE: Session = CASAUTO Name = CASUSER(casdemo)                    1
         Type = PATH
         Description = Personal File System Caslib
         Path = /u/casdemo/
         Definition =
         Subdirs = Yes
         Local = No
         Active = No
         Personal = Yes
NOTE: Session = CASAUTO Name = HADOOPLIB                           2
         Type = hadoop
         Description = 'Hadoop Caslib'
         Path =
         Definition = ui
         Subdirs = No
         Local = No
         Active = No
         Personal = No
NOTE: Session = CASAUTO Name = VAPUBLIC                            3
         Type = DNFS
         Description =
         Path = /vapublic/
         Definition =
         Subdirs = No
         Local = No
         Active = No
         Personal = No
NOTE: Session = CASAUTO Name = Formats                             4
         Type = PATH
         Description = Format Caslib
         Path = /casdemo/formats/
         Definition =
         Subdirs = Yes
         Local = No
         Active = No
```

Additional Information

When you list caslibs with the `CASLIB _ALL_ LIST;` statement, the following caslib setting information is displayed for all caslibs:

Type=string	indicates the caslib type specified by the SRCTYPE= option.
Description=string	displays the description specified by the DESCRIPTION= option.
Path=string	displays the path specified in the PATH= option.
Definition=string	displays the data source options specified by the DATASOURCE= option.
Subdirs=Yes | No	indicates whether the caslib can access subdirectories. Subdirs= is specified by the SUBDIRS option.
Local=Yes | No	indicates whether the caslib is session scope. **`Local=yes`** indicates session scope. Local= corresponds to the SESSION option in the CASLIB statement.

Active=Yes | No indicates whether the caslib is the active caslib.

Personal=Yes | No indicates whether the caslib is a personal caslib or not.

"SAS Log Showing Data Connector Settings" on page 756. The settings that are displayed correspond to data connector options and are specific to the data connector. For the data source options syntax for databases such as Hadoop and Oracle, see "Data Connectors" on page 167.

- You can also use the CASUTIL procedure to view information about caslibs. For documentation about the CASUTIL procedure, see Chapter 3, "CASUTIL Procedure," on page 57.
- For the Hadoop connect string syntax, see "Hadoop Data Connector" on page 212.

Example 3: Load and Save a Table

Program

The following program adds the caslib Myvapublic, which provides a place to copy an in-memory version of the Cars data set. The in-memory table, named carsWght, is then saved to the DNFS data source.

```
caslib Myvapublic path="/vapublic"

datasource=(srctype="dnfs");              /*1*/

libname mycas cas
caslib=myvapublic;                                /*2*/

data
mycas.carsWght;                                                    /*3*/
   set sashelp.cars;
   if weight<5500 then delete;
   keep make model type weight MPG_City;
run;

proc casutil incaslib="Myvapublic"
outcaslib="Myvapublic";         /*4*/
   list tables incaslib="Myvapublic";
   save casdata="CarsWght";
run;
```

1 The first CASLIB statement adds a global-scope caslib named Myvapublic. The DATASOURCE= option and the PATH= option provide connection information to the Myvapublic directory. Myvapublic is now the active caslib.

2 The LIBNAME statement assigns the libref Mycas. To run a DATA step in CAS, you must specify the CAS engine LIBNAME statement and use the CAS engine libref with both the input and output table names. The CASLIB= option binds the libref to the Myvapublic caslib.

3 This DATA step creates an in-memory table named CarsWght in the Myvapublic caslib. There is no on-disk representation and it does not persist in the Myvapublic directory unless you save it.

4 The CASUTIL procedure saves the table to Myvapublic. The INCASLIB= option specifies the caslib that contains the file, and the OUTCASLIB= option specifies the caslib that the file is being made available to. Use the PROC CASUTIL LIST statement to make sure that the table exists in the Myvapublic caslib.

SAS Log

The note in the SAS log shows that the table was saved.

```
60         save casdata="CarsWght";
NOTE: Cloud Analytic Services saved the file CarsWght.sashdat to DNFS in caslib
Myvapublic.
```

Results

The following results of the PROC CASUTIL LIST TABLES statement show the tables that are in the Myvapublic caslib.

Caslib Information	
Library	MYVAPUBLIC
Source Type	HDFS
Path	/vapublic/
Session local	No
Active	Yes
Personal	No
Hidden	No
Transient	No

The SAS System

The CASUTIL Procedure

Table Information for Caslib MYVAPUBLIC										
Table Name	Number of Rows	Number of Columns	Indexed Columns	NLS encoding	Created	Last Modified	Promoted Table	Repeated Table	View	Compressed
CARSWGHT	7	5	0	utf-8	2018-05-22T17:27:03-04:00	2018-05-22T17:27:03-04:00	No	No	No	No

Additional Information

- Caslibs provide a way to organize in-memory tables and an associated data source. They also provide a way to apply access controls to data. Once the caslib is dropped, the in-memory tables are dropped, too. Files in the caslib's data source are not removed or modified. To add tables to a data source permanently, use the SAVE statement in PROC CASUTIL.
- Tables that are saved from a caslib are saved in SASHDAT format by default.
- For information about using the DATA step in CAS, see *SAS Cloud Analytic Services: DATA Step Programming*.
- For documentation about the CASUTIL procedure syntax, see Chapter 3, "CASUTIL Procedure," on page 57.
- For more examples of using the CASUTIL procedure to access and save data, see Chapter 1, "Accessing Data with SAS Cloud Analytic Services," on page 1.

Example 4: Copy Data from One Data Source to Another

```
caslib ldbeta datasource=(srctype="path")
                    path="path-to-directory"
                    description="imported
files";                 /*1*/

proc casutil incaslib="LDbeta"
outcaslib="hps";                    /*2*/
   contents
casdata="cars.csv";                                        /*3*/
   load casdata="cars.csv" casout="cars";                          /
*4*/
   list tables
incaslib="hps";                                          /*5*/
   save casdata="cars" incaslib="hps";                             /
*6*/
run;
```

1 The CASLIB statement adds a session-scope caslib named Ldbeta. The DATASOURCE= option provides connection information to a path-based directory.

2 The CASUTIL procedure loads and saves the table to the data source. The INCASLIB= option specifies the caslib that contains the file, and the OUTCASLIB= option specifies the caslib that the file is being made available to. HPS is a global caslib that is available to all users.

3 The CONTENTS statement reads the on-disk file, Cars.csv, and displays the table metadata. This enables you to learn if the file has column names in the first row and the data types.

4 The LOAD CASDATA= statement reads the CSV file into memory and explicitly names the table Cars. The table is now available for analytics.

5 The LIST TABLES statement confirms that the in-memory table named Cars is available in the HPS caslib.

6 The SAVE statement saves the table as a SASHDAT file so that it can be loaded from the HPS caslib in the future rather than imported from the CSV file.

Results

The following results show table metadata displayed by the PROC CASUTIL CONTENTS statement.

Output 14.1 *Table Metadata*

CAS File Information

Name	Permission	Owner	Group	Encryption Method	File Size	Last Modified (UTC)
cars.csv	-rw-r--r--	sasdemo	users		41.2KB	09SEP2016:23:27:05

Column Information for cars.csv in Caslib LDBETA

Column	Type	Length	Formatted Length	Format Width	Format Decimal
Make	varchar	0	16	0	0
Model	varchar	0	16	0	0
Type	varchar	0	16	0	0
Origin	varchar	0	16	0	0
DriveTrain	varchar	0	16	0	0
MSRP	double	8	12	0	0
Invoice	double	8	12	0	0
EngineSize	double	8	12	0	0
Cylinders	double	8	12	0	0
Horsepower	double	8	12	0	0
MPG_City	double	8	12	0	0
MPG_Highway	double	8	12	0	0
Weight	double	8	12	0	0
Wheelbase	double	8	12	0	0
Length	double	8	12	0	0

The following results show the files available in the HPS caslib, displayed by the PROC CASUTIL LIST statement.

Output 14.2 *Partial Display of Files Available in the HPS Caslib*

Table Information for Caslib HPS(sasdemo)

Table Name	Number of Rows	Number of Columns	Indexed Columns	NLS encoding	Created	Last Modified	Promoted Table	Repeated Table	View	Source Name	Source Caslib	Compressed
CARS	428	15	0	utf-8	2018-05-23T13:42:05-04:00	2018-05-23T13:42:05-04:00	No	No	No	cars.csv	LDBETA	No

Additional Information

- Caslibs provide a way to organize in-memory tables and an associated data source. They also provide a way to apply access controls to data. After a caslib is dropped, the in-memory tables are dropped, too. Files in the caslib's data source are not removed or modified in any way. To add tables to a data source permanently, use the SAVE statement in PROC CASUTIL.
- Tables that are saved from a caslib are saved in SASHDAT format by default.
- For information about using the DATA step in CAS, see *SAS Cloud Analytic Services: DATA Step Programming*.
- For documentation about the CASUTIL procedure syntax, see Chapter 3, "CASUTIL Procedure," on page 57.
- For more examples of using the CASUTIL procedure to access and save data, see Chapter 1, "Accessing Data with SAS Cloud Analytic Services," on page 1.

Example 5: Encrypt Tables in a Caslib

```
caslib Encr datasource=(srctype="path", encryptionPassword="your-
password")
                        path="your-file-
path";                       /*1*/

libname mycas
cas;                                                    /*2*/

proc casutil;
   load data=sashelp.cars groupby=("make")
outcaslib="Encr";         /*3*/
   contents
casdata="cars";                                        /*4*/
   list tables
incaslib="encr";                                      /*5*/

proc mdsummary
data=mycas.cars;                                      /*6*/
   var msrp invoice;
   output out=mycas.mdsumstatEncr;
   groupby make;
run;

options obs=15;
proc print
data=mycas.mdsumstatEncr;                                /*7*/
   var Make _Column_ _NObs_ _Mean_ _Max_ _Min_ _Std_;
   title "Summary of MSRP and Invoice, Grouped by Make";
run;

proc casutil incaslib="Encr"
outcaslib="Encr";                        /*8*/
```

```
   save
casdata="mdsumstatEncr";                                    /* 9 */
   list files
incaslib="encr";                                          /* 10 */
run;
```

1 The ENCRYPTIONPASSWORD= option in the CASLIB statement specifies a password for encrypting or decrypting tables.

2 Assign a CAS engine libref with the LIBNAME statement. To run PROC MDSUMMARY and PROC PRINT in CAS, you must specify the CAS engine LIBNAME statement and use the CAS engine libref with both the input and output table names.

3 The LOAD DATA= statement reads the file into memory. The table is now available for analytics.

4 The CONTENTS statement reads the on-disk file, Cars, and displays the table metadata. This enables you to determine whether the file has column names in the first row and the data types.

5 The LIST TABLES statement confirms that the in-memory table named Cars is available in the Encr caslib.

6 The MDSUMMARY procedure computes the descriptive statistics and groups them by Make.

7 The PRINT procedure prints the output.

8 The CASUTIL procedure saves the table to the data source. The INCASLIB= option specifies the caslib that contains the file, and the OUTCASLIB= option specifies the caslib that the file is being made available to.

9 The SAVE statement stored the data as UTF-8 when it created the SASHDAT file.

10 The LIST FILES statement confirms that the in-memory table named MdsumstatEncr is saved in the data source that the Encr caslib is associated with.

Results: LIST Statement

The following partial results of the PROC CASUTIL LIST statement show that the caslib is encrypted and a password is specified.

Output 14.3 *ENCR Caslib Information*

Caslib Information	
Library	ENCR
Source Type	PATH
Path	/cloud.com/test/
EncryptionPassword	*****
Session local	Yes
Active	Yes
Personal	No
Hidden	No

Results: CONTENTS Statement

Output 14.4 *Column Information for the Cars Data Set*

Column Information for CARS in Caslib ENCR							
Column	Label	Type	Length	Format Name	Formatted Length	Format Width	Format Decimal
Make		char	13		13	0	0
Model		char	40		40	0	0
Type		char	8		8	0	0
Origin		char	6		6	0	0
DriveTrain		char	5		5	0	0
MSRP		double	8	DOLLAR	8	8	0
Invoice		double	8	DOLLAR	8	8	0
EngineSize	Engine Size (L)	double	8		12	0	0
Cylinders		double	8		12	0	0
Horsepower		double	8		12	0	0
MPG_City	MPG (City)	double	8		12	0	0
MPG_Highway	MPG (Highway)	double	8		12	0	0
Weight	Weight (LBS)	double	8		12	0	0
Wheelbase	Wheelbase (IN)	double	8		12	0	0
Length	Length (IN)	double	8		12	0	0

Results: LIST Statement Information for the Data Source

Output 14.5 *List of Files in the Data Source*

CAS File Information						
Name	Permission	Owner	Group	Encryption Method	File Size	Last Modified
mdsumstatEncr.sashdat	-rwxr-xr-x	sasdemo	unix	AES	19.6KB	06Apr2016:20:50:26

Additional Information

- Caslibs provide a way to organize in-memory tables and an associated data source. They also provide a way to apply access controls to data. After a caslib is dropped, the in-memory tables are dropped too. Files in the caslib's data source are not removed or modified in any way. To add tables to a data source permanently, use the SAVE statement in PROC CASUTIL.
- Tables that are saved from a caslib are saved in SASHDAT format by default.
- For information about using the DATA step in CAS, see *SAS Cloud Analytic Services: DATA Step Programming*.
- For documentation about the CASUTIL procedure syntax, see Chapter 3, "CASUTIL Procedure," on page 57.
- For more examples of using the CASUTIL procedure to access and save data, see Chapter 1, "Accessing Data with SAS Cloud Analytic Services," on page 1.
- For documentation about the MDSUMMARY procedure, see Chapter 4, "MDSUMMARY Procedure," on page 105.

LIBNAME Statement: CAS Engine

Associates a SAS libref with in-memory tables on the CAS server.

Category: Data Access

Data type: CHAR, VARCHAR, and DOUBLE

Tip: If you do not use the active caslib, these connections are valid: HOST= and PORT= (together), SESSREF=, UUID=, or UUIDMAC=.

Syntax

Form 1: **LIBNAME** *libref* CAS <*options*>;

Form 2: **LIBNAME** *libref* CAS **HOST=** *"Kubernetes-service-name"* **PORT=***network-port-number* <*options*>;

Form 3: **LIBNAME** *libref* CAS **SESSREF=***session-reference* <*options*>;

Required Arguments

CAS
specifies the CAS LIBNAME engine.

libref
specifies a valid SAS name that serves as a shortcut name to associate with the tables on the CAS server. The name must conform with SAS naming conventions. It can be up to eight bytes long.

LIBNAME Options Supported by the CAS Engine

The following table contains a list of optional LIBNAME statement options supported by the CAS LIBNAME engine.

Table 14.4 *Optional CAS LIBNAME Statement Arguments*

Option	Description	Default
CASLIB= on page 615	Specifies the name of the caslib to use for engine operations in the LIBNAME statement.	The active caslib
COMPRESS= on page 616	Requests that the CAS table is compressed.	NO
DATALIMIT= on page 617	Specifies the maximum number of bytes of data that can be transferred between the CAS server and SAS during a single-table read.	100M
HOST= on page 618	Specifies the host name of the primary controller.	The host name is determined by the session unless the SESSREF= option and the PORT= option are specified.
MAXTABLEMEM= on page 621	Specifies the maximum amount of memory, in bytes, to allocate for an in-memory table.	16M
NCHARMULTIPLIER= on page 619	Specifies a multiplication factor that is used to increase the number of bytes for a fixed character variable when data is transcoded to the UTF-8	1 for SBCS environments 1.5 for DBCS environments

Option	Description	Default
	encoding in order to run in the CAS server.	
PORT= on page 620	Specifies the network port number on which the controller listens for client connections.	The port is determined by session unless the SESSREF= option and the PORT= option are specified.
READTRANSFERSIZE= on page 622	Specifies the maximum number of bytes that the CAS engine reads when transferring data from the CAS server.	500MB
SESSREF= on page 624	Specifies the name of the CAS session to which you want to connect.	
TAG= on page 624	Specifies a multi-level prefix for a CAS table that is used to identify the table to both the CAS server and SAS.	
TRANSCODE_FAIL= on page 626	Specifies how the engine handles transcoding errors.	ERROR
UUID= on page 627	Connects the CAS engine to the CAS session that is identified in the UUID.	
UUIDMAC= on page 628	Specifies a macro variable that connects a subsequent SAS step to a session that was created or modified by the CAS LIBNAME engine.	If you do not specify this option, the UUID is stored in the _IOCASUUID_ macro variable value.
WRITETRANSFERSIZE= on page 629	Specifies the maximum number of bytes that the CAS engine writes when transferring data to the CAS server.	512KB

Example: Execute a DATA Step

Once you have assigned a CAS engine libref, you can run a DATA step as you would for any other data source. The DATA step reads data and loads it into a CAS table.

In the example below, the DATA step reads data from a table in a Base engine library and writes data to a CAS table using the CAS engine library.

```
cas casauto host="cloud.example.com" port=5570;    /*1*/

libname mycas cas;                                 /*2*/


 data mycas.cars;
    set sashelp.cars;                              /*3*/
run;

data mycas.somecars2;
   set mycas.cars;                                 /*4*/
   by make model;
   keep make model weight origin;
run;


proc print data=mycas.cars2;                       /*5*/
run;
```

1 Start a CAS session.

2 Assign a libref.

3 Use the engine to transfer data from SAS to the CAS server.

4 Run the DATA step in CAS to subset data.

5 Print the in-memory table Cars2.

See Also

- CASLIB statement,
- COMPRESS= LIBNAME option