SAS® Cloud Analytic Services: Language Reference
# Contents

_Early Adopter Software_ ........................................................................................................... v
_About This Book_ ................................................................................................................... vii

**Chapter 1 • CAS Statement** ................................................................. 1
  Dictionary ............................................................................................................. 1

**Chapter 2 • CAS LIBNAME Statement** ........................................ 19
  What You Can Do with the CAS LIBNAME Engine ...................... 19
  Dictionary ........................................................................................................ 20

**Chapter 3 • CASLIB Statement** .................................................. 35
  Dictionary ........................................................................................................ 35

**Chapter 4 • CASUTIL Procedure** ............................................. 53
  Overview: CASUTIL Procedure ............................................................... 53
  Syntax: CASUTIL Procedure ................................................................. 54
  Enclose Values in Quotation Marks .................................................. 65
  Limitations and Restrictions ............................................................... 65
  Results: CASUTIL Procedure ............................................................... 65
  Examples: CASUTIL Procedure .......................................................... 67

**Chapter 5 • MDSUMMARY Procedure** ..................................... 75
  Overview: MDSUMMARY Procedure .................................................. 75
  Syntax: MDSUMMARY Procedure ......................................................... 75
  PROC MDSUMMARY Output Data Sets ....................................... 77
  Results: MDSUMMARY Procedure ..................................................... 78
  Examples: MDSUMMARY Procedure ............................................... 79

**Chapter 6 • Platform Data Sources** ............................................ 93
  Data Redundancy ....................................................................................... 93
  Dictionary .................................................................................................. 94

**Chapter 7 • Platform File Types** ................................................ 101
  Dictionary .................................................................................................. 101

**Chapter 8 • Data Connectors** ....................................................... 107
  Working with SAS Data Connectors ................................................. 107
  Quick Reference for Data Connector Syntax ........................................... 107

**Chapter 9 • Data Connector Reference** .......................................... 109
  Dictionary .................................................................................................. 109

**Chapter 10 • Data Types** ................................................................. 129
  SAS Cloud Analytic Services Data Types ........................................... 129

**Chapter 11 • Functions** ................................................................. 131
  Dictionary ................................................................................................. 131
Chapter 12 • Macro Variables ................................................................. 135
Dictionary ...................................................................................... 135

Chapter 13 • Session Options ............................................................. 137
Setting Session Options ................................................................. 137
Session Options by Category ......................................................... 139
Dictionary ...................................................................................... 140

Chapter 14 • System Options ............................................................. 151
Dictionary ...................................................................................... 151
Early Adopter Software

THIS DOCUMENTATION FOR AN EARLY ADOPTER PRODUCT IS A PRELIMINARY DRAFT AND IS PROVIDED BY SAS INSTITUTE INC. ("SAS") ON AN "AS IS" BASIS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR PURPOSE. SAS does not warrant that this documentation is complete, accurate, or similar to that which may be released to the general public, or that any such documentation will be released. The company shall not be liable whatsoever for any damages arising out of the use of this documentation, including any direct, indirect, or consequential damages. SAS reserves the right to alter or abandon use of this documentation at any time.
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.
Chapter 1
CAS Statement

Dictionary

CAS Statement
Starts and manages your SAS Cloud Analytic Services session.

Syntax
Form 1: CAS;
Form 2: CAS session-name <option(s)>;

Summary of Optional Arguments

Authentication option
AUTHINFO="authentication-info-file"
specifies an authinfo or netrc file that includes authentication information.

CAS server information options
LISTABOUT
lists information about SAS Cloud Analytic Services.
LISTSERVERSTARTOPTS
lists the SAS Cloud Analytic Services server options and their current values.

Connection options
DISCONNECT
disconnects SAS from the session.
HOST="host-name"
specifies the machine name for the control node of the server.
PORT=number
specifies the port on which the control node of the SAS Cloud Analytic Services server listens for client connections.
RECONNECT
reconnects to a session using a session name.

TERMINATE
terminates the SAS Cloud Analytic Services session.

TIMEOUT=seconds
specifies the session time-out in seconds.

USER=user-ID
specifies the user ID to use for connecting to the SAS Cloud Analytic Services server.

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
lists information about a session in your SAS client.

LISTSESSIONS
lists information about all of the current user's sessions that are known to the SAS Cloud Analytic Services server.

Session property options

LISTSESSOPTS
lists the session options.

SESSOPTS=(session-option(s))
specifies one or more session options as option=value pairs separated by a space and enclosed in parentheses.

User-defined format options

ADDFMTLIB FMTLIBNAME=format-library-name TABLE=table-name
<CASLIB=calib> <POSITION=APPEND | INSERT | REPLACE | NONE> <PROMOTE> <REPLACEFMTLIB>

ADDFMTLIB FMTLIBNAME=format-library-name PATH=path
<POSITION=APPEND | INSERT | REPLACE | NONE> <PROMOTE> <REPLACEFMTLIB>
adds a session format library.

DROPFMTLIB FMTLIBNAME=format-library-name <FMTSEARCHREMOVE>
drops a session-local or a global format library.

FMTSEARCH=(name1 <name2…nameN> ) <POSITION=APPEND | INSERT | REPLACE>

FMTSEARCH CLEAR
modifies or clears the format library search order for the session.

LISTFMTRANGES FMTNAME=format-name
lists the ranges for a format.

LISTFMTSEARCH
 lists the format library search order 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=table-name
<CASLIB=caslib> <REPLACEFMTLIB>
SAVEFMTLIB FMTLIBNAME=format-library-name PATH=path
```

saves a session format library to a CAS table or to a file.

**Without Arguments**

When specified without arguments, a session is created using the name that is specified by system option CASNAME (alias SESSREF), which is CASAUTO by default. See “CASNAME= System Option” on page 155. Default values are used for all of the session options. See Chapter 13, “Session Options,” on page 137.

**Required Argument**

- `session-name`
  - specifies a valid SAS name that is less than 256 characters.

  **Note** If session `session-name` does not exist, it is created, and macro variable `_SESSREF_` and SAS system option CASNAME (alias SESSREF) are set to `session-name`. If session `session-name` already exists, the options specified are applied to existing session `session-name`.

**Optional Arguments**

ADD FMTLIB FMTLIBNAME=format-library-name TABLE=table-name
<CASLIB=caslib> <POSITION=APPEND | INSERT | REPLACE | NONE> <PROMOTE> <REPLACEFMTLIB>

ADD FMTLIB FMTLIBNAME=format-library-name PATH=path
<POSITION=APPEND | INSERT | REPLACE | NONE> <PROMOTE> <REPLACEFMTLIB>

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 save. This option is required.

- `TABLE=table-name`
  - specifies the name of the table where the format library is saved.

  **Note** Do not specify this option with PATH=.

- `CASLIB=caslib`
  - specifies the name of the caslib where the table is saved.

  **Note** This option is ignored when PATH= is specified.

- `PATH=path`
  - specifies the name of the file where the format information is saved.

  **Note** Do not specify this option with TABLE=.

- `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.
**DISCONNECT**

disconnects SAS from the session.

**Note**
The session name is preserved for use with the RECONNECT option.

**Tip**
The TIMEOUT= value determines the lifetime of a disconnected session. The session time-out starts when the number of connections becomes zero and no actions are executing. After the time-out value expires, the session is terminated.

**See**
“Example 6: Disconnect from a Session” on page 15

**DROPFMTLIB FMTLIBNAME=** *format-library-name*  <FMTSEARCHREMOVE>
drops a session-local or a global format library.

**FMTLIBNAME=** *format-library-name*
specifies the name of the format library to drop. This option is required.

**FMTSEARCHREMOVE**
removes the format library from the format search order.

**Tip**
If a session-local and a global format library exist, the local format library is dropped first. To drop the global format library in that case, execute the drop command again.

**FMTSEARCH=** *(name1 <name2...nameN>)  <POSITION=APPEND | INSERT | REPLACE>*

**FMTSEARCH CLEAR**
modifies or clears the format library search order for the session.

*(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

CLEAR clears the format library search list.

Examples
Add format libraries FMTLIB1 and FMTLIB2 to the beginning of the format library search order for session Casauto:
cas casauto fmtsearch=(fmtlib1 fmtlib2) position=insert;
Clear the format search list:
cas casauto fmtsearch clear;

HOST="host-name"
specifies the machine name for the control node of the server.

Default: Macro variable _CASHOST_, if set. Otherwise, SAS system option CASHOST.

Interaction: This option overrides macro variable _CASHOST_ and SAS system option CASHOST.

See "_CASHOST_ Macro Variable" on page 135
"CASHOST= System Option" on page 153

LIST lists information about a session in your SAS client. The information includes the session name, the session state, the host and port of the SAS Cloud Analytic Services server to which it is connected, and the session UUID.

Note: The server is not accessed for the information.

Tips: Use _ALL_ instead of session-name to list information about all of the sessions in your SAS client.
To list information about all of the sessions in all of your SAS clients, use LISTSESSIONS.

See "Example 2: List Information about the Sessions in Your SAS Client" on page 13

LISTABOUT lists information about SAS Cloud Analytic Services. The information is written to the SAS log and is organized as shown in the following table.
<table>
<thead>
<tr>
<th>Section</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>About</td>
<td>CAS</td>
<td>Product name: SAS Cloud Analytic Services.</td>
</tr>
<tr>
<td></td>
<td>Version</td>
<td>Short-form SAS Cloud Analytic Services version.</td>
</tr>
<tr>
<td></td>
<td>Copyright</td>
<td>Copyright information.</td>
</tr>
<tr>
<td>System</td>
<td>Hostname</td>
<td>SAS Cloud Analytic Services host name.</td>
</tr>
<tr>
<td></td>
<td>OS Name</td>
<td>SAS Cloud Analytic Services host information.</td>
</tr>
<tr>
<td></td>
<td>OS Family</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OS Release</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OS Version</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Model Number</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Documentation</td>
<td>SAS Cloud Analytic Services documentation URL.</td>
</tr>
<tr>
<td>License</td>
<td>Site</td>
<td>Site information.</td>
</tr>
<tr>
<td></td>
<td>SiteNum</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Expires</td>
<td>License expiration date and time.</td>
</tr>
<tr>
<td></td>
<td>GracePeriod</td>
<td>Grace period in days.</td>
</tr>
<tr>
<td></td>
<td>WarningPeriod</td>
<td>Expiration warning period in days.</td>
</tr>
<tr>
<td></td>
<td>Setinit</td>
<td>Path to the SETINIT file.</td>
</tr>
</tbody>
</table>

**LISTFMTRANGES FMTNAME=**format-name

lists the ranges for a format.

FMTNAME=**format-name**
specifies the name of the format to list. This option is required. The format libraries are searched in the format-library search order. The ranges are listed for the first instance of **format-name** that is found.

Example

List the ranges in format MYFORMAT:
LISTFMTRANGES fmtname=myformat;

LISTFMTRANGES
lists the format library search order for the session.

Example
List the format library search order for session Casauto:
cas casauto listfmtranges;

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 SESSION

MEMBERS
lists the names of the members in each format library.

Example
List the global and session-local formats and their members for session Casauto:
cas casauto listformats scope=both members;

LISTSERVERSTARTOPTS
lists the SAS Cloud Analytic Services server 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 = errors
Type = INT RANGE
Value = 20
Minimum = 0
Maximum = 2147483647
Source = default

Alias LISTSSO

Restriction You must have administration privileges to use this option.

LISTSESSIONS
lists information about all of the current user's sessions that are known to the SAS Cloud Analytic Services server. The information includes the session name, the session UUID, the session state, the method that was used for user authentication, and the user ID.
A connection to a session is required to use the LISTSESSIONS option.

See
“Example 3: List Information about All of Your Sessions” on page 14

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.

LISTSESSOPTS
lists the session options.

See
Chapter 13, “Session Options,” on page 137 for a list of the session options.
“Example 4: List the Properties for a Session” on page 14

Example
List the session option settings for session Casauto:
cas casauto LISTSESSOPTS;

PORT=number
specifies the port on which the control node of the SAS Cloud Analytic Services server listens for client connections.

Default
Macro variable _CASPORT_, if set. Otherwise, SAS system option CASPORT.

Range
1-65535

Interaction
This option overrides macro variable _CASPORT_ and SAS system option CASPORT.

See
“_CASPORT_ Macro Variable” on page 135
“CASPORT= System Option” on page 156

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.

Example
Promote session-local format library MYFMTLIB in session Casauto:
cas casauto promote fmtlibname=myfmtlib;

RECONNECT
reconnects to a session using a session name.

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 10.

See
“Example 7: Connect to an Existing Session” on page 16
SAVFMTLIB FMTLIBNAME=format-library-name TABLE=table-name
<CASLIB=caslib> <REPLACETABLE>
SAVFMTLIB FMTLIBNAME=format-library-name 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=table-name
specifies the name of the table in which the format library is saved.
  Note Do not specify this option with PATH=.
CASLIB=caslib
specifies the caslib in which the table is stored.
  Note This option is ignored when PATH= is specified.
REPLACETABLE
replaces the table if it already exists.
  Note This option is ignored when PATH= is specified.
PATH=path
specifies the name of the file to which the format library is to be saved.
  Note Do not specify this option with TABLE=.
SESSOPTS=(session-option(s))
specifies one or more session options as option=value pairs separated by a space and enclosed in parentheses.
  See Chapter 13, “Session Options,” on page 137 for a list of the options that you can specify for session-option(s).
  “Example 5: Change a Property for a Session” on page 15
  “Program: Create a Session with Custom Properties” on page 12
TERMINATE
terminates the SAS Cloud Analytic Services session.
  Alias CLEAR
  Note When you terminate the last session that was created, SAS system option CASNAME (alias SESSREF) and macro variable _SESSREF_ are not updated. They continue to reference the terminated session until a new session is created or they are manually set. See “CASNAME= System Option” on page 155.
  Tip Use _ALL_ instead of session-name to terminate all of the sessions in your SAS client.
  See “Example 8: Terminate a Session” on page 17
TIMEOUT=seconds
specifies the session time-out in seconds. The session time-out starts when the number of connections becomes zero and no actions are executing.
### USER=**user-ID**

**specifies the user ID to use for connecting to the SAS Cloud Analytic Services server.**

<table>
<thead>
<tr>
<th>Alias</th>
<th>CASUSER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>SAS system option CASUSER=, if set.</td>
</tr>
<tr>
<td>Interaction</td>
<td>This option overrides SAS system option CASUSER=.</td>
</tr>
<tr>
<td>Note</td>
<td>When you use SAS Studio, 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.</td>
</tr>
<tr>
<td>See</td>
<td>“CASUSER= System Option” on page 158</td>
</tr>
</tbody>
</table>

### UUID="**session-uuid**"

**specifies the UUID of an existing session to which you want to connect.**

<table>
<thead>
<tr>
<th>Requirements</th>
<th>session-uuid must be 36 characters in length and must be enclosed in quotation marks.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>You must also specify the HOST= and PORT= options to connect to a session.</td>
</tr>
<tr>
<td>Tip</td>
<td>You can view the UUID for a session with the LIST command option.</td>
</tr>
<tr>
<td>See</td>
<td>“LIST” on page 5</td>
</tr>
</tbody>
</table>

---

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

<table>
<thead>
<tr>
<th>Range</th>
<th>0–31536000</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Restriction</th>
<th>This option applies only to a new session. To change the time-out for an existing session, use the TIMEOUT= session option.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction</td>
<td>This option overrides the CASTIMEOUT= system option.</td>
</tr>
<tr>
<td>Notes</td>
<td>When set to 0, the session is terminated immediately when the connection count 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.</td>
</tr>
</tbody>
</table>

See

- “TIMEOUT= Session Option” on page 148
- “CASTIMEOUT= System Option” on page 157
**Program: Connect to an Existing Session Using the Session Name and UUID**

**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 steps to connect to the session by specifying the UUID.

**See** “Program: Create a Session and Store the UUID in a Macro Variable” on page 13

**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
- create a session
- terminate a session

**Creating Your Initial Session**

After you sign in to SAS Studio, you must create a session in order to connect to the SAS Cloud Analytic Services server. After you create your session, you can use it to complete your tasks. Code snippet **New Session** in SAS Studio provides the SAS code that is needed to create a session. Alternatively, you can submit your own CAS statement to create a customized session. See “Example 1: Create a Session” on page 12 for examples. Use the CAS statement to perform management tasks on your session, such as 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 or to 2 if an error occurred. 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 6: Disconnect from a Session” on page 15.

**Managing Sessions with the CAS Server Monitor**

You can use the CAS Server Monitor to perform the following session tasks:

- list information about your sessions
- cancel the action that is currently running in a session
- cancel a session
• terminate a session

Note: Some tasks in the CAS Server Monitor might require administration privileges.

Use the GETCASURL function to get the CAS Server Monitor URL. See “GETCASURL Function” on page 131.

For information about the CAS Server Monitor, see “About CAS Server Monitor” in SAS Viya Administration: Getting Started.

Examples

Example 1: Create a Session

Program: Create a Session with Default Properties

This example creates session Casauto with default session properties:

```sas
   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 cloud.example.com using port 5570. The UUID is session-UUID.
       The user is casdemo and the default CASLIB is CASUSERHDFS(casdemo).
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.
```

Program: Create a Session with Custom Properties

This example creates session Casauto with a 60-minute time-out, caslib Casuser, and with metrics enabled:

```sas
   cas casauto sessopts=(timeout=3600 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 cloud.example.com using port 5570. The UUID is session-UUID.
       The user is casdemo and the default CASLIB is CASUSER(casdemo).
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(casdemo)' is now the active caslib.
NOTE: Action 'setsessopt' used (Total process time):
       real time              0.181410 seconds
       cpu time               0.446927 seconds (246.36%)
       total nodes           142 (4536 cores)
       total memory          35.47T
       memory                 13.09M (0.00%)
NOTE: The CAS server request to update one or more session options for session CASAUTO completed.
```
Program: Create a Session and Store the UUID in a Macro Variable

This example creates session Casauto and stores the session UUID in macro variable CASAUTO_UUID for later use:

```
cas casauto uuidmac=casauto_uuid;
%put Session casauto UUID: &casauto_uuid;
```

SAS Log

The %PUT statements 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. See “CASLIB Statement” on page 35 and “CAS LIBNAME Statement” on page 20.
- 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, your session is automatically terminated if a connection is not established within the time-out period. See TIMEOUT= on page 148.
- Always terminate your session when you are finished with it.

Example 2: 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 cloud.example.com for user casdemo. 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, which are Casauto, Mysess1, and Mysess2, in this example:

```
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:

```plaintext
NOTE: Session CASAUTO is ACTIVE using port 5570 and host cloud.example.com for user casdemo. The session UUID is session-UUID.
NOTE: Session MYSESS1 is ACTIVE using port 5570 and host cloud.example.com for user casdemo. The session UUID is session-UUID.
NOTE: Session MYSESS2 is ACTIVE using port 5570 and host cloud.example.com for user casdemo. The session UUID is session-UUID.
```

Example 3: 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 the SAS Cloud Analytic Services server. You must be connected to an existing session in order to use the LISTSESSIONS option.

```sas
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.

```plaintext
NOTE: SessionName = CASAUTO:Mon Feb  8 12:49:00 2016
   UUID= session-UUID
   State = Connected
   Authentication = Active Directory
   Userid = casdemo
NOTE: SessionName = MYSESS1:Mon Feb  8 12:49:03 2016
   UUID= session-UUID
   State = Connected
   Authentication = Active Directory
   Userid = casdemo
NOTE: SessionName = MYSESS2:Mon Feb  8 12:49:05 2016
   UUID= session-UUID
   State = Connected
   Authentication = Active Directory
   Userid = casdemo
NOTE: Request to LISTSESSIONS completed for session CASAUTO.
```

Example 4: List the Properties for a Session Program

This example lists the properties for session Casauto:

```sas
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
     Min = 0
     Max = 0
     Description = specifies the string to prefix to log messages.

NOTE: Name = caslib
     UsageType = Session
     Type = String
     Value = CASUSER(stengl)
     Default Value =
     Group = Caslib
     Min = 0
     Max = 0
     Description = specifies the caslib name to set as the active caslib.
```

Additional Information

For information about the session properties, see Chapter 13, “Session Options,” on page 137. The GETSESSOPT function enables you to get the value of a single property. See “GETSESSOPT Function” on page 131.

**Example 5: Change a Property for a Session**

Program

Change the time-out to 60 minutes for session Casauto:

```
cas casauto sessopts=(timeout=3600);
```

Additional Information

For information about the session properties that you can change, see Chapter 13, “Session Options,” on page 137. To change one or more properties for all of your generated SAS Cloud Analytic Services sessions, specify the property settings in an OPTIONS statement. See “CAS Statement” on page 1.

**Example 6: Disconnect from a Session**

Before You Disconnect

If no other client connection exists 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 7: Connect to an Existing Session” on page 16.
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.
```

Example 7: 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 created 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 created 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 created 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 3: List Information about All of Your Sessions” on page 14. This example uses session name Mysess and the UUID option to connect to session Mysess, which was created 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 cloud.example.com using port 5570. The UUID is ca683ddf-fe18-3c48-a04e-45718220976d. The user is casdemo and the default CASLIB is CASUSER(casdemo).
NOTE: The session is using nnn workers.
```
**Example 8: Terminate a Session**

**Program**

Terminate session Casauto:

```plaintext
cas casauto terminate;
```

**SAS Log**

The following notes are written to the SAS log:

- NOTE: Deletion of the session CASAUTO was successful.
- NOTE: Request to TERMINATE completed for session CASAUTO.
What You Can Do with the CAS LIBNAME Engine

You can use the CAS LIBNAME engine to work with SAS Cloud Analytic Services sessions, create a new session, connect to an existing session, or reference SAS Cloud Analytic Services libraries.

You can also use it to work with SAS Cloud Analytic Services tables to add, delete, replace, or append to them, or you can run SAS procedures on them.

When you use this engine with a library in SAS Cloud Analytic Services, the active caslib is used unless you specify the CASLIB= LIBNAME option. You can override both the active caslib and the CASLIB= LIBNAME option with the CASLIB= data set option.

You can also specify options on the SAS Cloud Analytic Services LIBNAME statement or in a DATA step.
Specify a data set option in parentheses after the data set name. When you specify a value for a data set option that has a corresponding CAS LIBNAME statement option (such as PROMOTE=), the data set option value takes precedence over the value for the CAS LIBNAME statement option.

---

**Dictionary**

**CAS LIBNAME Statement**

Associates a SAS libref with tables on the SAS Cloud Analytic Services server.

- **Valid in:** CAS LIBNAME engine
- **Category:** Data Access
- **Default:** none
- **Requirement:** For a valid connection, you must specify only one of these mutually exclusive methods:
  - host=
  - sessref=
  - uuid or uuidmac
- **Data type:** Supports varying-length data types
- **See:** CASLIB statement, COMPRESS= LIBNAME option

---

**Syntax**

Form 1:  
```plaintext
libname libref cas (host="controller-host-name"
port="network-port-number") | options;
```

Form 2:  
```plaintext
libname sessref=session-name | options;
```

Form 3:  
```plaintext
libname libref cas (uuid=identifier | uuidmac=macro_variable) | options;
```

Form 4:  
```plaintext
libname libref cas;
```

**Required Arguments**

- **libref**
  - specifies a valid SAS name that serves as a shortcut name to associate with the tables on the SAS Cloud Analytic Services server. The name must conform with SAS naming conventions. It can be up to eight bytes long and is the handle to the SAS Cloud Analytic Services library or session.

- **cas**
  - specifies the CAS LIBNAME engine.

- **host**
  - specifies the machine for the SAS Cloud Analytic Services server control node. It is recommended that you specify a host. Otherwise, the engine checks for a value in the CASHOST= system option and then in the _CASHOST_ macro variable. If neither of these contain a value, it uses the server on the local machine (host="localhost").
Alias server=

port=network-port-number

specifies an integer for the port where the SAS Cloud Analytic Services control node listens for client connections. It is recommended that you specify a port. Otherwise, the engine checks for a value in the CASHOST= system option and then in the _CASHOST_ macro variable.

Optional Arguments

caslib=caslib

specifies the name of a caslib to bind to the libref. By default, engine operations with the libref use the active caslib from your session. The active caslib can change as a result of adding or dropping caslibs. Use this option to specify an alternative caslib. When you specify this option, an engine operation using the libref uses the caslib that you specify, regardless of the active caslib for the session. You can override this option with the CASLIB= data set option.

sessref=session-name

specifies the name of the SAS Cloud Analytic Services session to which you want to connect. Using this option is a preferred alternative to HOST= and PORT=.

uuid=identifier

specifies a universally unique identifier (UUID) as a name or a quoted string. If you specify this option, the engine connects to the SAS Cloud Analytic Services session as identified in the UUID.

Tip

Most people prefer to use sessref= instead of uuid=.

Example

This example starts a SAS Cloud Analytic Services session through the CAS statement, saves the UUID in the MyUUID macro variable, connects the CAS LIBNAME to that session, and loads the FOO table into it.

```
cas mysess port=19999 host="rdcgrd001" UUIDMAC=MyUUID;
< Perform some additional tasks. >

libname mycas cas host="rdcgrd001" port=19999 uuid="&MyUUID";

data mycas.foo;
  < some DATA step code >
run;
```

uuidmac=macro-variable

specifies a SAS macro variable as a name or as a quoted string. If you specify this option, the engine saves the UUID of the last SAS Cloud Analytic Services session that it created. This option is useful if you want a subsequent SAS step to connect to a session that the CAS LIBNAME engine created or modified. If you do not specify this option, the UUID is stored in the _IOCASUUID_ macro variable.

Details

If you use the WHERE clause or statement with the CAS LIBNAME engine, it is resolved on the SAS Cloud Analytic Services server if it is able to handle it. Otherwise, records are pulled back to the SAS client. In this case, a note is written to the log only if SAS Cloud Analytic Services processed the WHERE clause or statement and you
specified MSGLEVEL=i. For more information about WHERE processing, see SAS System Options: Reference.

**CASLIB= LIBNAME Option**

Specifies the name of the caslib to use for engine operations on the LIBNAME statement.

- **Valid in:** CAS LIBNAME statement
- **Category:** Data Access
- **Default:** the active caslib
- **Tips:** If you specify both the CASLIB= LIBNAME and data set option, the data set option has precedence.
  
  To override the active caslib or the CASLIB= LIBNAME option, use the CASLIB= data set option.

- **See:** CASLIB= statement, CASLIB= data set option, CASLIB= system option

**Syntax**

```
CASLIB=caslib
```

**COMPRESS= LIBNAME Option**

Requests that the table to be created in SAS Cloud Analytic Services is compressed.

- **Valid in:** CAS LIBNAME statement
- **Category:** Data Access
- **Default:** NO
- **Restrictions:** The CAS LIBNAME engine does not support the COMPRESS= system option.

  This option is ignored when tables are opened for input.

- **See:** COMPRESS= data set option

**Syntax**

```
COMPRESS=<YES | NO>
```

**DATALIMIT= LIBNAME Option**

Specifies the maximum total amount of bytes that can be read from a file.

- **Valid in:** CAS LIBNAME statement
- **Category:** Data Access
- **Default:** 100MB
- **Restriction:** This option affects only read access.
Syntax

DATALIMIT=(integer) | K | M | G | "ALL"

Optional Arguments

**integer**
- specifies the total number of bytes to read.

**K**
- specifies the total number of kilobytes to read.

**M**
- specifies the total number of megabytes to read.

**G**
- specifies the total number of gigabytes to read.

"ALL"
- specifies that the entire file can be read, no matter how large it is.

READTRANSFERSIZE= LIBNAME Option

Specifies the maximum data transfer size in bytes that can be used when reading a table from SAS Cloud Analytic Services.

**Valid in:** CAS LIBNAME statement

**Category:** Data Access

**Alias:** RTS

**Default:** 500MB

**Restriction:** This option affects only read access.

**Interaction:** READTRANSFERSIZE= is the maximum amount of data that is transferred with each read request that is made to SAS Cloud Analytic Services. 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 READTRANSFERR= option.

**See:** DATALIMIT= LIBNAME option, DATALIMIT= data set option, READTRANSFERSIZE= data set option, WRITETRANSFERSIZE= LIBNAME option, WRITETRANSFERSIZE= data set option

**Examples:**
- 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 but larger than READTRANSFERSIZE=, this error results:
ERROR: The maximum allowed bytes of data have been fetched from SAS Cloud Analytic Services.

Table size=2GB, DATALIMIT="ALL", READTRANSFERSIZE=500MB. The entire table is handled in two read requests because the table size is equal to DATALIMIT= and greater than READTRANSFERSIZE=.

Syntax

READTRANSFERSIZE=<integer | K | M | G>

Optional Arguments

integer
    specifies the total number of bytes to read.

K
    specifies the total number of kilobytes to read.

M
    specifies the total number of megabytes to read.

G
    specifies the total number of gigabytes to read.

"ALL"
    specifies that the entire file can be read, no matter how large it is.

WRITETRANSFERSIZE= LIBNAME Option

Specifies the maximum data transfer size in bytes that can be used when writing to a table in SAS Cloud Analytic Services.

Valid in: CAS LIBNAME statement
Category: Data Access
Alias: WTS
Default: 512KB
Restriction: This option affects only write access.
See: DATALIMIT= LIBNAME option, DATALIMIT= data set option, READTRANSFERSIZE= LIBNAME option, READTRANSFERSIZE= data set option, WRITETRANSFERSIZE= data set option

Syntax

WRITETRANSFERSIZE=<integer | K | M | G>

Optional Arguments

integer
    specifies the total number of bytes to write.

K
    specifies the total number of kilobytes to write.
M
specifies the total number of megabytes to write.

G
specifies the total number of gigabytes to write.

"ALL"
specifies that the entire file can be written, no matter how large it is.

APPEND= Data Set Option
Specifies whether to append rows to the SAS Cloud Analytic Services table from the DATA step.

Valid in: DATA and PROC steps
Category: Data Access
Default: NO

Syntax
APPEND=<YES | NO>

CASLIB= Data Set Option
Specifies the name of the caslib to use for engine operations for the data set.

Valid in: DATA and PROC steps
Category: Data Access
Default: the active caslib
Tips: If you specify both the CASLIB= LIBNAME and data set option, the data set option has precedence.
To override the active caslib or the CASLIB= LIBNAME option, use the CASLIB= data set option.
See: CASLIB= statement, CASLIB= LIBNAME option, CASLIB= system option

Syntax
CASLIB=caslib

COMPRESS= Data Set Option
Requests that the table to be created in SAS Cloud Analytic Services is compressed.

Valid in: DATA and PROC steps
Category: Data Access
Default: NO
Restrictions: The CAS LIBNAME engine does not support the COMPRESS= system option.
This option is ignored when tables are opened for input.
See: COMPRESS= LIBNAME option

**Syntax**

```
COMPRESS=<YES | NO>
```

---

**COPIES= Data Set Option**

Specifies the number of replicate copies for a replicated table.

- **Valid in:** DATA and PROC steps  
- **Category:** Data Access  
- **Default:** NO

**Syntax**

```
COPIES=number
```

---

**DATALIMIT= Data Set Option**

Specifies the maximum total amount of bytes that can be read from a file.

- **Valid in:** DATA step  
- **Category:** Data Access  
- **Default:** 100MB  
- **Restriction:** This option affects only read access.

**See:** DATALIMIT= LIBNAME option, READTRANSFERSIZE= LIBNAME option, READTRANSFERSIZE= data set option, WRITETRANSFERSIZE= LIBNAME option, WRITETRANSFERSIZE= data set option

**Syntax**

```
DATALIMIT=<(integer) | K | M | G | "ALL">  
```

**Optional Arguments**

- `integer` specifies the total number of bytes to read.
- `K` specifies the total number of kilobytes to read.
- `M` specifies the total number of megabytes to read.
- `G` specifies the total number of gigabytes to read.
- "ALL" specifies that the entire file can be read, no matter how large it is.
**DUPLICATE= Data Set Option**

Specifies whether the output table in the SAS Cloud Analytic Services is duplicated on all nodes.

- **Valid in:** DATA and PROC steps
- **Category:** Data Access
- **Default:** NO
- **Restriction:** This option applies only to a distributed server and corresponds to the REPEAT option for the LOAD DATA= statement of the CASUTIL procedure.
- **Interaction:** The value for COPIES= is ignored (no duplicated data is replicated) if you use COPIES= with DUPLICATE=.
- **Note:** With a duplicated table, all nodes have all rows of the table, and these rows are active everywhere.

**Syntax**

```
DUPLICATE=<YES | NO>
```

---

**ONDEMAND= Data Set Option**

Specifies how to evaluate temporary computed columns.

- **Valid in:** DATA and PROC steps
- **Category:** Data Access
- **Default:** NO
- **Tip:** On-demand execution is recommended when you fetch data from SAS Cloud Analytic Services to the SAS session, such as when the SAS Cloud Analytic Services table is the input data of a DATA step or a procedure.

**Syntax**

```
ONDEMAND=<YES | NO>
```

**Syntax Description**

- **YES**
  - Specifies that temporary computed columns are evaluated one row at a time.
- **NO**
  - Specifies that temporary computed columns are evaluated collectively at the outset.

---

**ORDERBY= Data Set Option**

Specifies the variables by which to order the data within a partition.

- **Valid in:** DATA and PROC steps
Category: Data Access
Default: ascending order
Requirement: To use this option, you must first specify the PARTITION= option.
See: PARTITION= data set option
Example: This example shows how to partition and order a data set into SAS Cloud Analytic Services using ascending ordering of the numeric variable Z and descending ordering of the CH character variable to order the data within partitions that are formed based on the variable KEY.

```sas
%let nn=10000;
data myCas.testPart(partition=(key)
    orderby = (z descending ch));
retain key;
do i = 1 to %nn
   n = int(ranuni(1)*50)+1;
do j = 1 to n;
   if (j=1) then do;
      key = int(ranuni(1)*%nn)+1;
   end; else if (j > n/2) then do;
      key = int(ranuni(1)*(%nn/100))+1;
   end;
x = 1;
y = round(rannor(1),0.25);
z = x + y;
ch = put(abs(j),best4.);
output;
end;
end;
run;
```

Syntax

```
ORDERBY=<ascending | descending><variable-list>
```

Syntax Description

`variable-list`
specifies a list of variables for ordering data within a partition. To specify descending order, include the DESCENDING keyword before the variable name in `variable-list`.

Details

Ordering is hierarchical. For example, `ORDERBY=(A B)` indicates that the values of variable B are ordered within the ordered values of variable A. One or more specified variables must exist and cannot be partitioning variables. Order is determined based on the raw value of the variables and uses locale-sensitive collation for character variables.

PARTITION= Data Set Option

Specifies the list of partitioning variables for the output table.

Valid in: DATA and PROC steps
Category: Data Access
Default: none
Requirement: You must first specify this option before you can use the ORDERBY= option.
See: ORDERBY= data set option

Syntax
PARTITION=(variable-list)

Details
- Partitioning information is ignored when tables are opened for input.
- Errors result from partitioning by a variable that does not exist on output or that is in the ORDERBY= option.
- Partition keys are derived based on formatted values as to how variable names are ordered in the variable list. Key construction is not hierarchical, so PARTITION=(A B) indicates that any unique combination of formatted values for A and B variables forms a partition of the data.
- Observations that share the same partition key are arranged together on the same worker node in SAS Cloud Analytic Services.
- Partitioning is also available for tables in SMP CAS servers.

PROMOTE= Data Set Option
Requests that the table to be created in SAS Cloud Analytic Services is added with global scope.
Valid in: DATA and PROC steps
Category: Data Access
Default: NO
Requirement: The caslib target must also have global scope.
Note: Global scope lets other sessions access the table, subject to access controls.

Syntax
PROMOTE=<YES | NO>

READTRANSFERSIZE= Data Set Option
Specifies the maximum data transfer size in bytes that can used when reading a table from SAS Cloud Analytic Services.
Valid in: DATA and PROC steps
Category: Data Access
Alias: RTS
Default: 500MB
Restriction: This option affects only read access.

Interaction: READTRANSFERSIZE= is the maximum amount of data that is transferred with each read request that is made to SAS Cloud Analytic Services. 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 READTRANSFER= option.

See: DATALIMIT= LIBNAME option, DATALIMIT= data set option, READTRANSFERSIZE= LIBNAME option, WRITETRANSFERSIZE= LIBNAME option, WRITETRANSFERSIZE= data set option

Examples: 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 but larger than READTRANSFERSIZE=, this error results:

ERROR: The maximum allowed bytes of data have been fetched from SAS Cloud Analytic Services.

Table size=2GB, DATALIMIT="ALL", READTRANSFERSIZE=500MB. The entire table is handled in two read requests because the table size is equal to DATALIMIT= and greater than READTRANSFERSIZE=.

Syntax

READTRANSFERSIZE=<(<integer>) | K | M | G>

Optional Arguments

integer
  specifies the total number of bytes to read.

K
  specifies the total number of kilobytes to read.

M
  specifies the total number of megabytes to read.

G
  specifies the total number of gigabytes to read.

"ALL"
  specifies that the entire file can be read, no matter how large it is.

SCRIPT= Data Set Option

Specifies the file reference for the SAS script that defines the temporary computed columns.

Valid in: DATA and PROC steps

Category: Data Access

Alias: TEMPEXPRESS=
TAG= Data Set Option

Specifies the tag from which to construct table names in SAS Cloud Analytic Services.

Valid in: DATA and PROC steps
Category: Data Access
Default: (null)
Tip: Use `TAG="MYTAG"` if you want the table name to be constructed with a tag.
Example: This example appends the rows in a SAS data set to the SAS Cloud Analytic Services table, `user.sassek.cas.test.sales_fact`.

```sas
libname mycas cas host="rdcgrd001" port=19999;
data mycas.sales_fact(append=yes tag="user.sassek.cas.test");
  set local_sales;
run;
```

SAS syntax continues to use a two-level name (`mycas.sales_fact`). In communicating with SAS Cloud Analytic Services, the uppercased `tagName` replaces `mycas`.

Syntax

`TAG=tagName`

Details

A SAS data set is identified by its libref and member name—for example, work.foo. Table names in SAS Cloud Analytic Services can have more than two levels. For example, you must be able to map the libref.member syntax to be able to work with the five-level name for the SAS Cloud Analytic Services `user.sassek.cas.test.sales_fact` table. The CAS LIBNAME engine constructs the table name in SAS Cloud Analytic Services as `upper(tagName).member`, so the `tagName` replaces the libref.

TEMPNAMES= Data Set Option

Lists the names of the temporary computed columns that are added to the input table.

Valid in: DATA and PROC steps
Category: Data Access
Length: Temporary columns can be character or numeric (8-byte doubles).
Default: none
Restriction: Temporary computed columns are supported only for tables that are opened for input.
Example: This 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 mycars.cars data set are then listed in the TEMPNAMES= data set option. It appears that there is some duplication of information, because t1 through t3 are defined in the SAS script and are listed in the TEMPNAMES= option. However, this is only because it is a very simple example. Scripts can be very complicated, using hundreds of variables in assignments and expressions. 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 $ sign and the variable length in bytes. For example, t2 in the subsequent code is a character variable of length 10, while t1 and t3 are numeric variables.

```sas
data _null_;
  file "./mypgm.sas";
  put "t1 = round(mpg_highway/mpg_city,0.5);";
  put "t2 = round(cylinders/enginesize,0.2);";
  put "t3 = msrp / invoice;";
;
filename newcols "./mypgm.sas";

proc ccor data=mycas.cars(tempnames=(t1 t2 t3)
  tempexpress=newcols);
  var mpg_city mpg_highway t1 t3 weight;
  by t2;
  where type ne 'Hybrid';
run;

/********************************************************************
/* Add a temporary character column to the input table. */
filename newcols "./myOtherpgm.sas";

proc ccor data=mycas.cars(tempnames=(t1 t2 $ 10 t3)
  tempexpress=newcols);
```

Syntax

```
TEMPNAMES=
```

**WRITETRANSFERSIZE= Data Set Option**

Specifies the maximum data transfer size in bytes that can be used when writing to a table in SAS Cloud Analytic Services.

- **Valid in:** DATA and PROC steps
- **Category:** Data Access
- **Alias:** WTS
- **Default:** 512KB
- **Restriction:** This option affects only write access.
- **See:** DATALIMIT= LIBNAME option, DATALIMIT= data set option, READTRANSFERSIZE= LIBNAME option, READTRANSFERSIZE= data set option, WRITETRANSFERSIZE= LIBNAME option,
**Syntax**

\[ \text{WRITETRANSFER}\text{SIZE} = <(integer) \mid K \mid M \mid G> \]

**Optional Arguments**

- **integer**
  - specifies the total number of bytes to write.

- **K**
  - specifies the total number of kilobytes to write.

- **M**
  - specifies the total number of megabytes to write.

- **G**
  - specifies the total number of gigabytes to write.

- **"ALL"**
  - specifies that the entire file can be written, no matter how large it is.
Chapter 3
CASLIB Statement

Dictionary

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 Cloud Analytic Services: Authorization.

See: For more examples of accessing data, see SAS Cloud Analytic Services: Accessing and Manipulating Data.

For conceptual information about caslibs, see “Caslibs” in SAS Cloud Analytic Services: Fundamentals.

Syntax

Form 1: CASLIB caslib-reference-name <SESSREF=session-reference>
DATASOURCE=(SRCTYPE="type" <data-source-options>)<PATH=directory-path><option(s)>;

Form 2: CASLIB caslib-reference-name LIST | DROP<SESSREF=session-reference>;

Form 3: CASLIB _ALL_ ASSIGN | _ALL_ LIST | _ALL_ DROP

Required Arguments

caslib-reference-name
specifies the name of the caslib.

Range 1 to 256 characters

Requirement Names of session caslibs must be unique within the session. Names of global caslibs must be unique across all sessions within a server.
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 or DROP options.

For naming rules see Names in the SAS Language.

**DATASOURCE= (SRCTYPE="type", <data-source-options>, <ENCRYPTIONPASSWORD="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.

**ENCRYPTIONPASSWORD="string"**

specifies a password for encrypting or decrypting stored data.

**Restriction** The ENCRYPTIONPASSWORD= option can be specified only with the DATASOURCE= option.

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 Hive and Oracle, see Chapter 8, “Data Connectors,” on page 107.

**Table 3.1 Path-Based Data Source Types and Options**

<table>
<thead>
<tr>
<th>SRCTYPE= Type</th>
<th>Option Syntax</th>
<th>Example</th>
<th>Supported File Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>PATH</td>
<td>DATASOURCE=(SRCTYPE=PATH, &lt;ENCRYPTIONPASSWORD=&quot;password&quot;&gt;), PATH=&quot;file-path&quot;;</td>
<td>caslib mycsvs path=&quot;/data/Myxlsxfiles/&quot; datasource=(srctype=&quot;path&quot;, encryptpassword=&quot;password&quot;);</td>
<td>SASHDAT, SAS7BDAT, CSV, XLS, XLSX</td>
</tr>
<tr>
<td>DNFS</td>
<td>DATASOURCE=(SRCTYPE=DNFS, &lt;ENCRYPTIONPASSWORD=&quot;password&quot;&gt;), PATH=&quot;file-path&quot;;</td>
<td>caslib mycsvs datasource=(srctype=&quot;dnfs&quot;), path=&quot;/data/Mycsvfiles/&quot;;</td>
<td>SASHDAT, CSV</td>
</tr>
<tr>
<td>HDFS</td>
<td>DATASOURCE=(SRCTYPE=HDFS, &lt;ENCRYPTIONPASSWORD=&quot;password&quot;&gt;), PATH=&quot;file-path&quot;;</td>
<td>caslib mycsvs path=&quot;/data/Mycsvfiles/&quot; datasource=(srctype=&quot;hdfs&quot;, encryptpassword=&quot;password&quot;);</td>
<td>SASHDAT, CSV</td>
</tr>
</tbody>
</table>

See For information about the PATH= option, see “PATH="directory-path"” on page 38.

Examples “Example 1: Add a Global Caslib” on page 39

“Example 6: Encrypt Tables in a Caslib” on page 48
Optional Arguments

ASSIGN
used with the _ALL_ option to create SAS librefs for existing caslibs so that they are visible in the SAS Studio Libraries tree.

Requirement The ASSIGN option must be used with the _ALL_ option.

_ALL_
specifies that the ASSIGN, DROP, or LIST argument applies to all currently added caslibs.

Example “Program 2: List All Caslibs” on page 42

DESCRIPTION="description"
specifies a description of the data source.

Alias DESC=

DROP
removes a caslib. The caslib is removed from all sessions that had access to it. If the removed caslib is the active caslib for a session, then the active caslib returns to the initially active caslib.

Alias CLEAR

Tip Specify caslib-reference-name to remove a single caslib. Specify _ALL_ to drop all caslibs.

Example “Example 2: Drop a Caslib” on page 40

GLOBAL
adds the caslib so that it has global scope. A global-scope caslib can have access controls set 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. See “Adjust Caslib Management Privileges” in SAS Cloud Analytic Servcies: Authorization.

See For information about caslibs scope, see “Caslibs Scope” in SAS Cloud Analytic Services: Fundamentals.

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.

Examples “Example 2: Drop a Caslib” on page 40

“Example 3: List Caslib Settings” on page 41

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 39

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.

Tip   You do not need to use the SUBDIRS option if the full path to the subdirectory
   is specified.

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 file. 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 4, “CASUTIL
Procedure,” on page 53.

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.
• drop caslibs.
• 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 sessions and the server. By default, your personal caslib in global scope. 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. For caslib authorization information, see “Adjust Caslib Management Privileges” in *SAS Cloud Analytic Services: Authorization*.

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 “Caslibs Scope” in *SAS Cloud Analytic Services: Fundamentals*.

**The Active Caslib**

When you start a session, your personal caslib is added by default. Initially, it is the active caslib. 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. You can also set the active caslib with the CASLIB= session option. For information about the CASLIB= session option, see “CASLIB= Session Option” on page 141.

**Examples**

**Example 1: Add a Global Caslib**

Program

The following example adds the caslibs Vapublic and Hivelib. They each contain connection information to the data sources. After running this program, Hivelib is the active caslib.

```plaintext
caslib Vapublic path="/vapublic"
   datasource=(srctype="hdfs") global ; /*1*/

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
```

```plaintext
caslib Hivelib desc="Hive Caslib"
   datasource=(srctype="hive", /*2*/
   dataTransferMode="parallel",
   hadoopjarpath="/data/cdh54/sdm/lib",
   hadoopconfigdir="/data/cdh54/sdm/conf",
   username="hiveuser",
   server="hive01.example.com",
   schema="default") 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 Hivelib, which provides access to a Hive 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. Hivelib is now the active caslib.

**SAS Log**

The notes in the SAS log verify that the caslibs Vapublic and Hivelib were added. Note that Hivelib is the active caslib, because it was added last.

```
56         caslib Vapublic path="/vapublic" datasource=(srctype="hdfs") 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 STUDIO.
57         caslib Hivelib desc='Hive Caslib' datasource=(srctype='hive',
58             dataTransferMode='parallel',
59             hadoopjarpath="/data/cdh54/sdm/lib",
60             hadoopconfigdir="/data/cdh54/sdm/
conf",
61             username="hiveuser",
62             server="hive01.example.com",
63             schema="default") global;
NOTE: 'HIVELIB' is now the active caslib.
NOTE: Cloud Analytic Services added the caslib 'HIVELIB'.
NOTE: Action to ADD caslib HIVELIB completed for session STUDIO.
```

**Additional Information**

- For information about specifying Hive data source options, see “SAS Data Connector to Hive, SAS Data Connect Accelerator to Hive” on page 109.
- You can add global caslibs only if you are authorized to do so, and the caslib must also be set to global in the CAS Server Monitor. See “Adjust Caslib Management Privileges” in SAS Cloud Analytic Services: Authorization.

**Example 2: Drop a Caslib**

**Program: Drop a Caslib**

The following CASLIB statement drops the caslib Hivelib. You can use a second CASLIB LIST statement to verify that the Hivelib caslib has been removed.

```
caslib Hivelib drop;
caslib _all_ list;
```

**SAS Log**

The notes in the SAS log verify that the caslib Hivelib was dropped.
NOTE: Cloud Analytic Services removed the caslib 'HIVELIB'.

NOTE: Session = STUDIO Name = CASUSER(casdemo)
Type = PATH
Description = Personal File System Caslib
Path = /u/casdemo/
Definition =
Subdirs = Yes
Local = No
Active = Yes
Personal = Yes

NOTE: Session = STUDIO Name = CASUSERHDFS(casdemo)
Type = HDFS
Description = Personal HDFS Caslib
Path = /user/casdemo/
Definition =
Subdirs = Yes
Local = No
Active = No
Personal = Yes

NOTE: Session = STUDIO Name = VAPUBLIC
Type = HDFS
Description =
Path = /vapublic/
Definition =
Subdirs = No
Local = No
Active = No
Personal = No

NOTE: Session = STUDIO Name = Formats
Type = PATH
Description = Format Caslib
Path = /bigdisk/lax/formats/
Definition =
Subdirs = Yes
Local = No
Active = No
Personal = No

NOTE: Session = STUDIO Name = HPS
Type = HDFS
Description = HDAT files on /hps
Path = /hps/
Definition =
Subdirs = Yes
Local = No
Active = No
Personal = No

Additional Information

- If you drop the active caslib from a session, the first active caslib becomes active again.

- You can also use the CASUTIL procedure to drop caslibs and view information about caslibs. For documentation about the CASUTIL procedure, see Chapter 4, "CASUTIL Procedure," on page 53.

Example 3: List Caslib Settings
You can list the caslib properties for one or more caslibs with the LIST statement.
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.

```sas
caslib CASUSER list;
caslib HPS list;
```

SAS Log

```
NOTE: Session = STUDIO Name = CASUSER\(casdemo\)
     Type = PATH
     Description = Personal File System Caslib
     Path = /u/casdemo/
     Definition =
     Subdirs = Yes
     Local = No
     Active = No
     Personal = Yes

NOTE: Session = STUDIO Name = HPS
     Type = HDFS
     Description = HDAT files on /hps
     Path = /hps/
     Definition =
     Subdirs = Yes
     Local = No
     Active = Yes
     Personal = No
```

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.

```sas
caslib \_all\_ list;
```

SAS Log

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. Caslibs 5 and 6 were created by another session, but have global scope, so they are available to all sessions.
NOTE: Session = STUDIO 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 = STUDIO Name = CASUSERHDFS(casdemo) 2
Type = HDFS
Description = Personal HDFS Caslib
Path = /user/casdemo/
Definition =
Subdirs = Yes
Local = No
Active = No
Personal = Yes

NOTE: Session = STUDIO Name = HIVELIB 3
Type = hive
Description = 'Hive Caslib'
Path =
Definition = ui
Subdirs = No
Local = No
Active = No
Personal = No

NOTE: Session = STUDIO Name = VAPUBLIC 4
Type = HDFS
Description =
Path = /vapublic/
Definition =
Subdirs = No
Local = No
Active = No
Personal = No

NOTE: Session = STUDIO Name = Formats 5
Type = PATH
Description = Format Caslib
Path = /bigdisk/lax/formats/
Definition =
Subdirs = Yes
Local = No
Active = No
Personal = No

NOTE: Session = STUDIO Name = HPS 6
Type = HDFS
Description = HDAT files on /hps
Path = /hps/
Definition =
Subdirs = Yes
Local = No
Active = Yes
Personal = No

Additional Information

When you list caslibs, the following caslib setting information is displayed:

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.

• You can also use the CASUTIL procedure to view information about caslibs. For documentation about the CASUTIL procedure, see Chapter 4, “CASUTIL Procedure,” on page 53.

• For the Hive connect string syntax, see “SAS Data Connector to Hive, SAS Data Connect Accelerator to Hive” on page 109.

Example 4: 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 HDFS data source.

```sas
proc caslib Myvapublic path="/vapublic"
   datasource=(srctype="hdfs"); /*1*/
libname mycas cas; /*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 Vapublic directory. Myvapublic is now the active caslib.

2 The LIBNAME statement creates 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.

3 This DATA step creates an in-memory table named Mycas.CarsWght in the Myvapublic caslib. There is no on-disk representation and it does not persist in the Vapublic directory unless you save it.

4 The CASUTIL procedure saves the table to Vapublic. 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 HDFS in caslib
MYVAPUBLIC.

Results

The following results of the PROC CASUTIL LIST TABLES statement show the tables
that are in the Myvapublic caslib.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Number of Rows</th>
<th>Number of Columns</th>
<th>NLS encoding</th>
<th>Created</th>
<th>Last Modified</th>
<th>Prompted Table</th>
<th>Repeated Table</th>
<th>View</th>
<th>Compressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARSWGHT</td>
<td>7</td>
<td>5</td>
<td>utf-8</td>
<td>16Jun2016:17:35:43</td>
<td>16Jun2016:17:35:43</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Additional Information

• Caslibs that use SRCTYPE="HDFS" are for distributed servers only. They use a
  Hadoop instance that is co-located with SAS Cloud Analytic Services.

• 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 “DATA Step Feature
  Engineering” in SAS Cloud Analytic Services: Accessing and Manipulating Data.

• For documentation about the CASUTIL procedure syntax, see Chapter 4,
  “CASUTIL Procedure,” on page 53.

• For more examples of using the CASUTIL procedure to access and save data, see
  “Accessing Data” in SAS Cloud Analytic Services: Accessing and Manipulating
  Data.

Example 5: 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="donations.csv"; /*3*/
  load casdata="donations.csv" casout="donations"; /*4*/
  list tables incaslib="hps"; /*5*/
  save casdata="donations" incaslib="hps"; /*6*/
run;
The first CASLIB statement adds a session-scope caslib named Ldbeta. The DATASOURCE= option provides connection information to a path-based directory.

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.

The CONTENTS statement reads the on-disk file, Donations.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.

The LOAD CASDATA= statement reads the CSV file into memory and explicitly names the table Donations. The table is now available for analytics.

The LIST TABLES statement confirms that the in-memory table named Donations is available in the HPS caslib.

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 3.1  Table Metadata

Summary of Presentation and Taste Scores
The CASUTIL Procedure

<table>
<thead>
<tr>
<th>CAS File Information</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
<td>Permission</td>
<td>Owner</td>
<td>Group</td>
</tr>
<tr>
<td>donations.csv</td>
<td></td>
<td>-rwxr-xr-x</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>809.8MB</td>
<td>14May2014</td>
<td>12:32:46</td>
</tr>
</tbody>
</table>

Column Information for donations.csv in Caslib LDBETA

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Length</th>
<th>Formatted Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>donationid</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>projectid</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>donor_acctid</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>donor_city</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>donor_state</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>donor_zip</td>
<td>double</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>is_teacher_acct</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>donation_timestamp</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>donation_to_project</td>
<td>double</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>donation_optional_support</td>
<td>double</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>donation_total</td>
<td>double</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>dollar_amount</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>donation_included_optional_support</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>payment_method</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>payment_included_acct_credit</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>payment_included_campaign_gift_card</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>payment_included_web_purchased_gift_card</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>payment_was_promo_matched</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>via_giving_page</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>for_honoree</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>donation_message</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
The following results show the files available in the HPS caslib, displayed by the PROC CASUTIL LIST statement.

**Output 3.2  Partial Display of Files Available in the HPS Caslib**

<table>
<thead>
<tr>
<th>Label</th>
<th>Number of Rows</th>
<th>Number of Columns</th>
<th>NL S encoding</th>
<th>Created</th>
<th>Last Modified</th>
<th>Prompted Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Data Types</td>
<td>279970</td>
<td>64</td>
<td>utf-8</td>
<td>18Mar2016:14:13</td>
<td>18Mar2016:14:13</td>
<td>Yes</td>
</tr>
<tr>
<td>Amazon Reviews</td>
<td>55320</td>
<td>25</td>
<td>latin1</td>
<td>18Mar2016:14:51</td>
<td>18Mar2016:14:51</td>
<td>Yes</td>
</tr>
<tr>
<td>Asia cities map coordinates</td>
<td>1830</td>
<td>12</td>
<td>latin1</td>
<td>18Mar2016:14:15</td>
<td>18Mar2016:14:15</td>
<td>Yes</td>
</tr>
<tr>
<td>Big organics</td>
<td>193723</td>
<td>26</td>
<td>latin1</td>
<td>18Mar2016:14:17</td>
<td>18Mar2016:14:17</td>
<td>Yes</td>
</tr>
<tr>
<td>Bodyfat2</td>
<td>111115</td>
<td>13</td>
<td>latin1</td>
<td>18Mar2016:14:54</td>
<td>18Mar2016:14:54</td>
<td>Yes</td>
</tr>
<tr>
<td>Campaign for Investments</td>
<td>396699</td>
<td>8</td>
<td>utf-8</td>
<td>18Mar2016:14:16</td>
<td>18Mar2016:14:16</td>
<td>Yes</td>
</tr>
<tr>
<td>Car Sales data from sashelp library</td>
<td>1660</td>
<td>33</td>
<td>latin1</td>
<td>18Mar2016:14:34</td>
<td>18Mar2016:14:34</td>
<td>Yes</td>
</tr>
<tr>
<td>Census</td>
<td>128605</td>
<td>172</td>
<td>latin1</td>
<td>18Mar2016:14:17</td>
<td>18Mar2016:14:17</td>
<td>Yes</td>
</tr>
<tr>
<td>Claim History</td>
<td>6376</td>
<td>4</td>
<td>utf-8</td>
<td>18Mar2016:14:17</td>
<td>18Mar2016:14:17</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**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 SASDAT format by default.
- For information about using the DATA step in CAS, see “DATA Step Feature Engineering” in *SAS Cloud Analytic Services: Accessing and Manipulating Data*.
- For documentation about the CASUTIL procedure syntax, see Chapter 4, “CASUTIL Procedure,” on page 53.
- For more examples of using the CASUTIL procedure to access and save data, see “Accessing Data” in *SAS Cloud Analytic Services: Accessing and Manipulating Data*.

**Example 6: Encrypt Tables in a Caslib**

```sas
options pageno=1 pageno=1;

caslib Encr datasource=(srctype="path", encryptionPassword="your-password") path="your-file-path";

run;

libname mycas cas;

proc casutil;
    load data=sashelp.cars groupby="make" outcaslib="Encr" path="your-file-path";
    contents casdata="cars" path="your-file-path";
    list files incaslib="encr" path="your-file-path";
    run;
```

Example 6: Encrypt Tables in a Caslib

```sas
proc mdsummary data=mycas.cars path="your-file-path";
    var msrp invoice;
    output out=mycas.mdsumstatEcr 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 Create 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 CASDATA= 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 if 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 3.3 ENCR Caslib Information

The CASUTIL Procedure

<table>
<thead>
<tr>
<th>Library</th>
<th>ENCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Type</td>
<td>PATH</td>
</tr>
<tr>
<td>Path</td>
<td></td>
</tr>
<tr>
<td>EncryptionPassword</td>
<td>****</td>
</tr>
<tr>
<td>Session local</td>
<td>Yes</td>
</tr>
<tr>
<td>Active</td>
<td>Yes</td>
</tr>
<tr>
<td>Personal</td>
<td></td>
</tr>
</tbody>
</table>

Results: CONTENTS Statement

Output 3.4 Column Information for the Cars Data Set

The Column Information for CARS in Caslib ENCR

<table>
<thead>
<tr>
<th>Column</th>
<th>Label</th>
<th>Type</th>
<th>Length</th>
<th>Formatted Length</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make</td>
<td>char</td>
<td>13</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>char</td>
<td>40</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>char</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Origin</td>
<td>char</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DriveTrain</td>
<td>char</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSRP</td>
<td>double</td>
<td>8</td>
<td>8</td>
<td>DOLLAR</td>
<td></td>
</tr>
<tr>
<td>Invoice</td>
<td>double</td>
<td>8</td>
<td>8</td>
<td>DOLLAR</td>
<td></td>
</tr>
<tr>
<td>EngineSize</td>
<td>Engine Size (L)</td>
<td>double</td>
<td>8</td>
<td>12</td>
<td>DOLLAR</td>
</tr>
<tr>
<td>Cylinders</td>
<td>double</td>
<td>8</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Horsepower</td>
<td>double</td>
<td>8</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPG_City</td>
<td>MPG (City)</td>
<td>double</td>
<td>8</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>MPG_Highway</td>
<td>MPG (Highway)</td>
<td>double</td>
<td>8</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Weight</td>
<td>Weight (LBS)</td>
<td>double</td>
<td>8</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Wheelbase</td>
<td>Wheelbase (IN)</td>
<td>double</td>
<td>8</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Length</td>
<td>Length (IN)</td>
<td>double</td>
<td>8</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

Results: LIST Statement Information for the Data Source

Output 3.5 List of Files in the Data Source

The CASUTIL Procedure

<table>
<thead>
<tr>
<th>Name</th>
<th>Permission</th>
<th>Owner</th>
<th>Group</th>
<th>File Size</th>
<th>Last Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>ms cumshot</td>
<td>-rwrxr-x-x</td>
<td></td>
<td></td>
<td>19.6KB</td>
<td>30Mar2016:10 01:16</td>
</tr>
</tbody>
</table>
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 “DATA Step Feature Engineering” in SAS Cloud Analytic Services: Accessing and Manipulating Data.

- For documentation about the CASUTIL procedure syntax, see Chapter 4, “CASUTIL Procedure,” on page 53.

- For more examples of using the CASUTIL procedure to access and save data, see “Accessing Data” in SAS Cloud Analytic Services: Accessing and Manipulating Data.

- For documentation about the MDSUMMARY procedure, see Chapter 5, “MDSUMMARY Procedure,” on page 75.
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**

```sas
PROC CASUTIL <option(s)>;
    CONTENTS CASDATA="table-name" <INCASLIB="caslib"> <option(s)>;
    DELETESOURCE CASDATA="file-name" <INCASLIB="caslib"> <QUIET>;
    DROPTABLE CASDATA="table-name" <INCASLIB="caslib"> <QUIET>;
    LIST <FILES | TABLES> <options(s)>;
    LOAD CASDATA="file-name" | DATA=SAS-data-set | FILE="SAS-file" <option(s)>;
    PROMOTE CASDATA="table-name" <INCASLIB="caslib">
        <CASOUT="table-name"> <OUTCASLIB="caslib"> <DROP | KEEP>;
    SAVE CASDATA="table-name" <INCASLIB="caslib">
        <CASOUT="file-name" <OUTCASLIB="caslib">> <option(s)>;
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: Accessing and Manipulating Data” in SAS Cloud Analytic Services: Accessing and Manipulating Data.

Syntax

PROC CASUTIL <option(s)>;

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-name
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=

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 71
“Example 4: Saving and Loading Encrypted SASHDAT Files” on page 72

Syntax

CONTENTS CASDATA=”table-name” <INCASLIB=”caslib”> <option(s)>;

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
    filename 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.
    
For information about file-type and file-type-options, see
    “IMPORTOPTIONS=(FILETYPE="file-type" file-type-options)” on page 60.

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.

OPTIONS=(data-source-options)
    specifies overrides for the DATASOURCE= options for the caslib. For more
    information, see Chapter 6, “Platform Data Sources,” on page 93

ROWCOUNT
    specifies to include the number of rows in the results. The data source for the input
caslib must be HDFS and you must include the filename suffix in the CASDATA=
    option.

VARLIST=((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 “VARLIST=((casinvardesc-1)
casinvardesc-2) …)” on page 61.

Example: Viewing Column Names from a CSV File
    contents casdata=somefile.csv importoptions=(filetype=csv);

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.

Note: You can delete files from path-based caslibs. These are caslibs with a data source
    type of DNFS, HDFS, or PATH.

Syntax

DELETESOURCE CASDATA="file-name" <INCASLIB="caslib"> <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.
QUIET suppresses error messages and avoids setting the SYSERR automatic macro variable when the specified table or file is not found.

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 72

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.

Note: This option is currently ignored for session-scope caslibs.

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 67

Syntax

LIST <FILES | TABLES> <option(s)>;

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.

OPTIONS=(data-source-options)
specifies overrides for the DATASOURCE= options for the caslib. For more information, see Chapter 6, “Platform Data Sources,” on page 93 and Chapter 8, “Data Connectors,” on page 107.

Applies to: LIST FILES

ROWCOUNT
specifies to include the number of rows in the results. The data source for the input caslib must be HDFS.

Applies to: LIST FILES

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"
<CASOUT="table-name">	<OUTCASLIB="caslib">	<APPEND | COMPRESS>	<GROUPBY=(group-by-variable-1 <group-by-variable-2 …>)	<ORDERBY=(variable-1 <variable-2 …>)>	<LABEL="table-description">	<OPTIONS=(data-source-options)>	<PROMOTE | REPLACE>	<REPEAT>
Form 3: `LOAD FILE="SAS-file" CASOUT="table-name"
<OUTCASLIB="caslib">`<br>`<COMPRESS>`<br>`<IMPORTOPTIONS=(FILETYPE=file-type <file-type-options>)>`<br>`<LABEL="table-description">`<br>`<PROMOTE | REPLACE>`<br>`<STAYMAPPED> <COPIES=integer>;

**Required Arguments**

**CASDATA="file-name"**<br>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.<br><br>Requirement You must specify CASOUT=.

**CASOUT="table-name"**<br>specifies the name to use for the in-memory table.<br><br>Interaction This argument is required when you use the LOAD CASDATA= or LOAD FILE= forms.<br><br>Note This argument does not follow or enforce SAS naming rules such as the name literal syntax.<br><br>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**<br>specifies the libref and data set name to use.<br><br>**FILE="SAS-file"**<br>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.<br><br>Requirement You must specify CASOUT=.

**Optional Arguments**

**APPEND**<br>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.

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

**COPIES=integer**<br>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.<br><br>Alias REPLICATION=
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.

FILETYPE="AUTO" | "BASESAS" | "CSV" | "DTA" | "EXCEL" | "HDAT" | "LASR" | "XLS"

specifies the file format. AUTO attempts to determine the file type based on the filename suffix, such as .sashdat, .csv, and so on. Files with a .txt suffix are imported as a CSV file.

Default AUTO

file-type-options

specifies options for importing the data. For more information, see Chapter 7, “Platform File Types,” on page 101 and Chapter 8, “Data Connectors,” on page 107.

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.

OPTIONS=(data-source-options)

specifies overrides for the DATASOURCE= options for the input caslib. For more information, see Chapter 6, “Platform Data Sources,” on page 93 and Chapter 8, “Data Connectors,” on page 107.

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.

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.

STAYMAPPED
specifies that after the server loads and accesses a table, the active blocks remain
memory-mapped by the operating system. By default, active blocks are mapped "on
demand" as data is accessed and unmapped after use. This option overrides that
default and keeps the blocks in virtual memory at all times. This gives the table some
preference for use of real memory and results in better performance for multi-pass
algorithms and frequent multi-user access.

VARLIST=((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.

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.

WHERE="where-expression-1 <logical-operatorwhere-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.

Details

Summary of Options
Table 4.1  Summary of LOAD Statement Options

<table>
<thead>
<tr>
<th>LOAD Statement Option</th>
<th>LOAD Statement Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPEND</td>
<td>LOAD DATA=</td>
</tr>
<tr>
<td>COMPRESS</td>
<td>LOAD DATA=</td>
</tr>
</tbody>
</table>
### LOAD Statement Option

<table>
<thead>
<tr>
<th>Option</th>
<th>LOAD Statement Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPIES=</td>
<td>All LOAD statement forms. For more information, see “Data Redundancy” on page 93.</td>
</tr>
<tr>
<td>GROUPBY=</td>
<td>LOAD CASDATA= and LOAD DATA=</td>
</tr>
<tr>
<td>IMPORTOPTIONS=</td>
<td>LOAD CASDATA= and LOAD FILE=</td>
</tr>
<tr>
<td>LABEL=</td>
<td>All LOAD statement forms</td>
</tr>
<tr>
<td>OPTIONS=</td>
<td>LOAD CASDATA= and LOAD DATA=</td>
</tr>
<tr>
<td>ORDERBY=</td>
<td>LOAD CASDATA= and LOAD DATA=</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> ORDERBY= requires GROUPBY=</td>
</tr>
<tr>
<td>PROMOTE</td>
<td>All LOAD statement forms</td>
</tr>
<tr>
<td>REPEAT</td>
<td>LOAD DATA=</td>
</tr>
<tr>
<td>REPLACE</td>
<td>All LOAD statement forms</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Keep in mind that global-scope tables cannot be replaced. Use the DROPTABLE statement before the LOAD statement.</td>
</tr>
<tr>
<td>VARLIST=</td>
<td>LOAD CASDATA=</td>
</tr>
<tr>
<td>WHERE=</td>
<td>LOAD CASDATA=</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You can specify WHERE= as a data set option when you use the LOAD DATA= form.</td>
</tr>
</tbody>
</table>

### 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 71

### Syntax

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

---

**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.

- **Note:** You can save an in-memory table to a file with path-based caslibs. These are caslibs with a data source type of DNFS, HDFS, or PATH.

- **Example:** “Example 4: Saving and Loading Encrypted SASHDAT Files” on page 72

**Syntax**

```
SAVE CASDATA="file-name" <INCASLIB="caslib">
  <CASOUT="table-name" <OUTCASLIB="caslib">>
    <option(s)>;
```

**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. 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 CASLIBNAME engine when you load it again.

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 the output caslib data source is HDFS and you save to the SASHDAT file format.

Alias REPLICATION=

Default 1

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. For more information, see IMPORTOPTIONS= on page 60 for the LOAD statement.

OPTIONS=(data-source-options)
specifies overrides for the DATASOURCE= options for the caslib. For more information, see Chapter 6, “Platform Data Sources,” on page 93 and Chapter 8, “Data Connectors,” on page 107.

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 61 for the LOAD statement.

---

**Enclose Values in Quotation Marks**

When you specify a value for one of the following items, 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.

---

**Limitations and Restrictions**

When working with a SAS library, the engine for the library is restricted to the Base engine. This enables you to work with SAS data sets, but using any other engine, such as a SAS/ACCESS engine, is not supported.

When working with a caslib, the SAVE statement is restricted to caslibs with a data source that supports saving tables as SASHDAT files. These caslib data source types are as follows:

- PATH
- HDFS
- DNFS

---

**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.

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

**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 variable-width character columns.

**Blocks Mapped**
- This field shows the number of blocks that are currently mapped into memory.

**Memory Mapped**
- This field shows the number of bytes for the blocks that are mapped.

**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.

### Table Information for Caslib CASUSER:

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Label</th>
<th>Number of Rows</th>
<th>Number of Columns</th>
<th>NLS encoding</th>
<th>Created</th>
<th>Last Modified</th>
<th>Promoted Table</th>
<th>Repeated Table</th>
<th>View</th>
<th>Compressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRIS</td>
<td>Fisher's Iris Data (1936)</td>
<td>150</td>
<td>5</td>
<td>utf-8</td>
<td>15Apr2016:12:01:26</td>
<td>15Apr2016:12:01:26</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

### Detail Information for iris in Caslib CASUSER:

<table>
<thead>
<tr>
<th>Node</th>
<th>Number of Blocks</th>
<th>Active Blocks</th>
<th>Rows</th>
<th>Fixed Data size</th>
<th>Variable Data size</th>
<th>Blocks Mapped</th>
<th>Memory Mapped</th>
<th>Blocks Unmapped</th>
<th>Memory Unmapped</th>
<th>Blocks Allocated</th>
<th>Memory Allocated</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>14</td>
<td>7</td>
<td>150</td>
<td>7200</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>16864</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Column Information for IRIS in Caslib CASUSER:

<table>
<thead>
<tr>
<th>Column</th>
<th>Label</th>
<th>Type</th>
<th>Length</th>
<th>Formatted Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Species</td>
<td>Iris Species</td>
<td>char</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>SepalLength</td>
<td>Sepal Length (mm)</td>
<td>double</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>SepalWidth</td>
<td>Sepal Width (mm)</td>
<td>double</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>PetalLength</td>
<td>Petal Length (mm)</td>
<td>double</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>PetalWidth</td>
<td>Petal Width (mm)</td>
<td>double</td>
<td>8</td>
<td>12</td>
</tr>
</tbody>
</table>
Blocks Allocated
This field shows the number of blocks that 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.
- you set the MAXTABLEMEM= CAS session option to a lower value. If you append rows and cross the value, the server caches the blocks.

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

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 directory is specified at deployment time by an administrator and corresponds to the CAS_DISK_CACHE environment variable.
- For distributed servers that are co-located with HDFS or 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 4.2  ODS Tables Produced by the CASUTIL Procedure**

<table>
<thead>
<tr>
<th>ODS Table</th>
<th>Description</th>
<th>Statement Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>CaslibInfo</td>
<td>Caslib information</td>
<td>LIST</td>
</tr>
<tr>
<td>ColumnInfo</td>
<td>Column information</td>
<td>CONTENTS</td>
</tr>
<tr>
<td>FileInfo</td>
<td>CAS file information</td>
<td>LIST with the FILES option</td>
</tr>
<tr>
<td>TableDetails</td>
<td>Detailed table information</td>
<td>CONTENTS</td>
</tr>
<tr>
<td>TableInfo</td>
<td>CAS table information</td>
<td>LIST with the TABLES option, CONTENTS</td>
</tr>
</tbody>
</table>

**Examples: CASUTIL Procedure**

**Example 1: Load a CSV File into CAS**

Program

```bash
  caslib csvfiles task=add type=dnfs /* 2 */
  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.

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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Permission</th>
<th>Owner</th>
<th>Group</th>
<th>File Size</th>
<th>Last Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spreadsheet10k.xlsx</td>
<td>-rw-rw-rwx</td>
<td>nfsnobody</td>
<td>nfsnobody</td>
<td>663.5KB</td>
<td>08Nov2012:07:21:12</td>
</tr>
<tr>
<td>mailorder.csv</td>
<td>-rw-rw-rwx</td>
<td>nfsnobody</td>
<td>nfsnobody</td>
<td>117.2KB</td>
<td>29Sep2015:11:40:14</td>
</tr>
<tr>
<td>NST_EST2012_ALLDATA.csv</td>
<td>-rw-rw-rwx</td>
<td>nfsnobody</td>
<td>nfsnobody</td>
<td>20.0KB</td>
<td>15Jan2013:08:59:28</td>
</tr>
<tr>
<td>sine.csv</td>
<td>-rw-rw-rwx</td>
<td>nfsnobody</td>
<td>nfsnobody</td>
<td>309.3KB</td>
<td>15Jan2013:13:18:14</td>
</tr>
<tr>
<td>SC-EST2011-6RACE-ALL.csv</td>
<td>-rw-rw-rwx</td>
<td>nfsnobody</td>
<td>nfsnobody</td>
<td>138.3KB</td>
<td>15Jan2013:08:02:53</td>
</tr>
<tr>
<td>sine10k.csv</td>
<td>-rw-rw-rwx</td>
<td>nfsnobody</td>
<td>nfsnobody</td>
<td>682.9KB</td>
<td>09May2013:07:16:16</td>
</tr>
<tr>
<td>UNdata_Export_20130115_092219549.csv</td>
<td>-rw-rw-rwx</td>
<td>nfsnobody</td>
<td>nfsnobody</td>
<td>595.4KB</td>
<td>15Jan2013:09:22:20</td>
</tr>
<tr>
<td>2012_rill_pbp_data_reg_season1.csv</td>
<td>-rw-rw-rwx</td>
<td>nfsnobody</td>
<td>nfsnobody</td>
<td>6.5MB</td>
<td>06Feb2013:11:19:02</td>
</tr>
<tr>
<td>County_Population.csv</td>
<td>-rw-rw-rwx</td>
<td>nfsnobody</td>
<td>nfsnobody</td>
<td>1.3MB</td>
<td>14Jan2013:14:15:41</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>The CASUTIL Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caslib Information</td>
</tr>
<tr>
<td>Library</td>
</tr>
<tr>
<td>Source Type</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Path</td>
</tr>
<tr>
<td>Session local</td>
</tr>
<tr>
<td>Active</td>
</tr>
<tr>
<td>Personal</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The SAS System</th>
</tr>
</thead>
<tbody>
<tr>
<td>The CASUTIL Procedure</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table Information for Caslib CSVFILES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Name</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>COUNTY_POPULATION</td>
</tr>
</tbody>
</table>

Example 2: Append Rows to an In-Memory Table

**Program**

```sas
proc casutil;
  load data=sashelp.cars(where=(make="Buick"))           /* 1 */
    casout="some_cars"
    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")))
    casout="some_cars"
    append;                                                /* 2 */
  list tables;
quit;

libname mycas cas;

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

proc casutil;
  contents casdata="cars_cardinality";                       /* 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

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

### Results: CONTENTS Statement for the Cars_Cardinality Table

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Number of Rows</th>
<th>Number of Columns</th>
<th>NLS encoding</th>
<th>Created</th>
<th>Last Modified</th>
<th>Promoted Table</th>
<th>Repeated Table</th>
<th>View</th>
<th>Compressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARS_CARDINALITY</td>
<td>26</td>
<td>3</td>
<td>utf-8</td>
<td>08Apr2016:13:29:19</td>
<td>08Apr2016:13:29:19</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

### Results: Select Columns from the Cars/Cardinality Table

<table>
<thead>
<tr>
<th>Obs</th>
<th><em>VARNNAME</em></th>
<th><em>CARDINALITY</em></th>
<th><em>NOBS</em></th>
<th><em>NMISS</em></th>
<th><em>MIN</em></th>
<th><em>MAX</em></th>
<th><em>MEAN</em></th>
<th><em>STDDEV</em></th>
<th><em>SKEWNESS</em></th>
<th><em>KURTOESIS</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EngineSize</td>
<td>17</td>
<td>47</td>
<td>0</td>
<td>2</td>
<td>6.8</td>
<td>3.39657446609</td>
<td>1.0133795046</td>
<td>0.777671628</td>
<td>1.347838265</td>
</tr>
<tr>
<td>2</td>
<td>MPG_Highway</td>
<td>15</td>
<td>47</td>
<td>0</td>
<td>13</td>
<td>36</td>
<td>26.639267872</td>
<td>4.3410354</td>
<td>-0.725695797</td>
<td>1.3622654198</td>
</tr>
<tr>
<td>3</td>
<td>MPG_City</td>
<td>12</td>
<td>47</td>
<td>0</td>
<td>10</td>
<td>27</td>
<td>19.382078723</td>
<td>3.4550101711</td>
<td>0.192944536</td>
<td>0.628379857</td>
</tr>
</tbody>
</table>
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 62 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.

### 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.

```sas
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 */
  contents casdata="jta_free_wifi";                            /* 3 */
  save casdata="jta_free_wifi" options=(encryptionPassword='changeit');
  droptable casdata="jta_free_wifi";                           /* 4 */
  load casdata="jta_free_wifi.sashdat"                         /* 5 */
```

---

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Number of Rows</th>
<th>Number of Columns</th>
<th>NLS encoding</th>
<th>Created</th>
<th>Last Modified</th>
<th>Promoted Table</th>
<th>Repeated Table</th>
<th>View</th>
<th>Compressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRISOUT</td>
<td>150</td>
<td>7</td>
<td>utf-8</td>
<td>08Apr2010:13:47:40</td>
<td>08Apr2010:13:47:40</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**Table Information for Caslib HPS**

<table>
<thead>
<tr>
<th>Node</th>
<th>Number of Blocks</th>
<th>Active Blocks</th>
<th>Rows</th>
<th>Fixed Data size</th>
<th>Variable Data size</th>
<th>Blocks Mapped</th>
<th>Memory Mapped</th>
<th>Blocks Unmapped</th>
<th>Memory Unmapped</th>
<th>Blocks Allocated</th>
<th>Memory Allocated</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>14</td>
<td>7</td>
<td>150</td>
<td>9600</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>18200</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Column Information for IRISOUT in Caslib HPS**

<table>
<thead>
<tr>
<th>Column</th>
<th>Label</th>
<th>Type</th>
<th>Length</th>
<th>Formatted Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Species</td>
<td>Iris Species</td>
<td>char</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>SepalLength</td>
<td>Sepal Length (mm)</td>
<td>double</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>SepalWidth</td>
<td>Sepal Width (mm)</td>
<td>double</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>PetalLength</td>
<td>Petal Length (mm)</td>
<td>double</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>PetalWidth</td>
<td>Petal Width (mm)</td>
<td>double</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>sepalratio</td>
<td></td>
<td>double</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>petalratio</td>
<td></td>
<td>double</td>
<td>8</td>
<td>12</td>
</tr>
</tbody>
</table>
Example 4: Saving and Loading Encrypted SASHDAT Files

The importoptions=(filetype="hdat" encryptionPassword='changeit');
quit;

proc mdsummary data=mycas.jta_free_wifi;                        /* 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 55 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 74.

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 output 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 4.1 on page 74.
Results: CONTENTS Statement for the Jta_free_wifi Table

The CASUTIL Procedure

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Number of Rows</th>
<th>Number of Columns</th>
<th>ALIAS Encoding</th>
<th>Created</th>
<th>Last Modified</th>
<th>Promoted Table</th>
<th>View</th>
<th>Source Name</th>
<th>Source Caslib</th>
<th>Compressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>JTA_FREE_WIFI</td>
<td>42250</td>
<td>21</td>
<td>utf-8</td>
<td>08Apr2016:13:56:01</td>
<td>08Apr2016:13:56:01</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Jta_free_wifi</td>
<td>No</td>
</tr>
</tbody>
</table>

---

Results: WiFi Hotspot Categories and Counts

Output 4.1 WiFi Hotspot Categories and Counts

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

PROC MDSUMMARY DATA=libref.table-name <NTHREADS=integer>;
   VAR <variable-list>;
   OUTPUT <OUT=table-name>;

GROUPBY variable-list \(</OUT=table-name>\);
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.

Requirement The table name must be a two-level name where Libref is a CAS engine libref.

Alias TABLE=

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.
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=table-name>;

Without Arguments
If no variables are listed, then the statistics are calculated across all input observations

Optional Arguments

OUT=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 with the OUT= option specified. You cannot specify both the OUTPUT= statement and a GROUPBY statement with the OUT= option specified.

Syntax

OUTPUT <OUT=libref.table name>;

Optional 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.

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 5.1  Table of Statistic Produced by the MDSUMMARY Procedure**

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Statistic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSS_</td>
<td>CSS</td>
<td>Corrected sum of squares</td>
</tr>
<tr>
<td>CV_</td>
<td>CV</td>
<td>Coefficient of variation</td>
</tr>
<tr>
<td>MAX_</td>
<td>MAX</td>
<td>Maximum value</td>
</tr>
<tr>
<td>MEAN_</td>
<td>MEAN</td>
<td>Arithmetic mean</td>
</tr>
<tr>
<td>MIN_</td>
<td>MIN</td>
<td>Minimum value</td>
</tr>
<tr>
<td>NMISS_</td>
<td>NMISS</td>
<td>Number of values that are missing</td>
</tr>
<tr>
<td>NOBS_</td>
<td>NOBS</td>
<td>Total number of observations</td>
</tr>
</tbody>
</table>
Example 1: Compute Descriptive Statistics

Program

The following example shows you how to access your data with SAS Cloud Analytics Services (CAS), compute all statistics for each variable, and treat the entire input table as one group. The results are written to an output table.

```
 caslib MyCasLib datasource={srctype="path"} path='your-file-path';/* 1 */

 libname mycas cas;/* 2 */

 proc casutil;/* 3 */
   load data=sashelp.cars outcaslib="MyCasLib ";
   contents casdata="cars";
 quit;

 proc mdsummary data=mycas.cars;/* 4 */
   var MSRP MPG_City;
   output out=mycas.mdsumstat;
 run;

 options obs=50;
 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" incaslib="MyCasLib" outcaslib="MyCasLib";
list files incaslib="MyCasLib";
quit;

1 The CASLIB statement adds a session-scope caslib named MyCasLib. The caslib provides access to your data source. The DATASOURCE= option and the PATH= option provide connection information to your data source.

2 The LIBNAME statement creates a CAS engine libref. 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.


5 The MDSUMMARY procedure does not print output. Use the PRINT procedure to print the table Mycas.MdsumStat.

6 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. The SAVE statement stored the data as UTF-8 when it created the SASHDAT file. The LIST FILES statement confirms that the in-memory table named Mdsumstat is saved in the data source.

Results

Output 5.1  Column Information for the Cars Data Set
Example 2: Computing Descriptive Statistics with Group-By Variables

Program

The following examples shows you how to access your data with SAS Cloud Analytics Services (CAS), compute all statistics for each variable (treating the entire input table as one group), and computes all statistics for every unique combination of the formatted values of variables Make and Type. The results are written to an output table.

```sas
libname mycas cas; /*
libname mycas cas; /*
proc casutil incaslib="casuser" outcaslib="casuser" ; /*
  load data=sashelp.cars;
  contents casdata="cars";
quit;

proc mdssummary data=mycas.cars; /*
  var MPG_City;
  groupby / out=mycas.carsmpgcityall; /*
  groupby make type / out=mycas.carsmaketype; /*
run;

option obs=50;
proc print data=mycas.carsmpgcityall; /*
  var _Column_ _NObs_ _Mean_ _Std_ _Min_ _Max_;
  title 'Overall City Mileage';
run;
```
option obs=25;
proc print data=mycas.carsmaketype;/**/
   var make Type _Column_ _Nobs_ _Mean_ _Std_ _Min_ _Max_;
   title 'City Mileage by Make and Type';
run;

1 The LIBNAME statement creates a CAS engine libref. 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.

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 MDSUMMARY procedure produces summary statistics for MPG_CITY.

4 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.

5 The second GROUPBY statement with the OUT= option creates an in-memory table named Mycas.CarsMakeType that calculates summaries for rows grouped by Make and Type.

6 The MDSUMMARY procedure does not print output. The PRINT procedure prints the table Mycas.MdsumStat.

Results: MDSUMMARY Procedure

Output 5.3 Overall Summary of City Mileage

<table>
<thead>
<tr>
<th>Obs</th>
<th><em>Column</em></th>
<th><em>Nobs</em></th>
<th><em>Mean</em></th>
<th><em>Std</em></th>
<th><em>Min</em></th>
<th><em>Max</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MPG_City</td>
<td>428</td>
<td>20.060747664</td>
<td>5.2382176386</td>
<td>10</td>
<td>60</td>
</tr>
</tbody>
</table>
### City Mileage by Make and Type

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Type</th>
<th><em>Column</em></th>
<th><em>NObs</em></th>
<th><em>Mean</em></th>
<th><em>Std</em></th>
<th><em>Min</em></th>
<th><em>Max</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>SUV</td>
<td>MPG_City</td>
<td>1</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>Sedan</td>
<td>MPG_City</td>
<td>5</td>
<td>20.4</td>
<td>2.6076809621</td>
<td>18</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>Sports</td>
<td>MPG_City</td>
<td>1</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>Audi</td>
<td>Sedan</td>
<td>MPG_City</td>
<td>13</td>
<td>18.615384615</td>
<td>2.3642638579</td>
<td>14</td>
<td>23</td>
</tr>
<tr>
<td>5</td>
<td>Audi</td>
<td>Sports</td>
<td>MPG_City</td>
<td>4</td>
<td>19</td>
<td>2.7080128015</td>
<td>15</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td>Audi</td>
<td>Wagon</td>
<td>MPG_City</td>
<td>2</td>
<td>16.5</td>
<td>2.1213203436</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>7</td>
<td>BMW</td>
<td>SUV</td>
<td>MPG_City</td>
<td>2</td>
<td>16</td>
<td>0</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>BMW</td>
<td>Sedan</td>
<td>MPG_City</td>
<td>13</td>
<td>19.230769231</td>
<td>0.8320502943</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>BMW</td>
<td>Sports</td>
<td>MPG_City</td>
<td>4</td>
<td>18.25</td>
<td>2.6299556397</td>
<td>16</td>
<td>21</td>
</tr>
<tr>
<td>10</td>
<td>BMW</td>
<td>Wagon</td>
<td>MPG_City</td>
<td>1</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>11</td>
<td>Buick</td>
<td>SUV</td>
<td>MPG_City</td>
<td>2</td>
<td>17</td>
<td>2.8264271247</td>
<td>15</td>
<td>19</td>
</tr>
<tr>
<td>12</td>
<td>Buick</td>
<td>Sedan</td>
<td>MPG_City</td>
<td>7</td>
<td>19.426571429</td>
<td>0.9759000729</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>13</td>
<td>Cadillac</td>
<td>SUV</td>
<td>MPG_City</td>
<td>2</td>
<td>15</td>
<td>1.4142135624</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>14</td>
<td>Cadillac</td>
<td>Sedan</td>
<td>MPG_City</td>
<td>4</td>
<td>18</td>
<td>0</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>15</td>
<td>Cadillac</td>
<td>Sports</td>
<td>MPG_City</td>
<td>1</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>16</td>
<td>Cadillac</td>
<td>Truck</td>
<td>MPG_City</td>
<td>1</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>17</td>
<td>Chevrolet</td>
<td>SUV</td>
<td>MPG_City</td>
<td>4</td>
<td>15.75</td>
<td>2.3629078131</td>
<td>14</td>
<td>19</td>
</tr>
<tr>
<td>18</td>
<td>Chevrolet</td>
<td>Sedan</td>
<td>MPG_City</td>
<td>15</td>
<td>22.266666667</td>
<td>4.0064560758</td>
<td>14</td>
<td>28</td>
</tr>
<tr>
<td>19</td>
<td>Chevrolet</td>
<td>Sports</td>
<td>MPG_City</td>
<td>2</td>
<td>18</td>
<td>0</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>20</td>
<td>Chevrolet</td>
<td>Truck</td>
<td>MPG_City</td>
<td>5</td>
<td>15.2</td>
<td>1.9235384062</td>
<td>13</td>
<td>18</td>
</tr>
<tr>
<td>21</td>
<td>Chevrolet</td>
<td>Wagon</td>
<td>MPG_City</td>
<td>1</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>22</td>
<td>Chrysler</td>
<td>Sedan</td>
<td>MPG_City</td>
<td>13</td>
<td>20.307692308</td>
<td>1.6525039276</td>
<td>18</td>
<td>22</td>
</tr>
<tr>
<td>23</td>
<td>Chrysler</td>
<td>Sports</td>
<td>MPG_City</td>
<td>1</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>24</td>
<td>Chrysler</td>
<td>Wagon</td>
<td>MPG_City</td>
<td>1</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>25</td>
<td>Dodge</td>
<td>SUV</td>
<td>MPG_City</td>
<td>1</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>
Results: PROC CASUTIL CONTENTS Statement

Output 5.5 Metadata for the Cars Table

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Label</th>
<th>Number of Rows</th>
<th>Number of Columns</th>
<th>Encoding</th>
<th>Created</th>
<th>Last Modified</th>
<th>Promoted Table</th>
<th>Repetited Table</th>
<th>View</th>
<th>Compressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARS</td>
<td>2004 Car Data</td>
<td>428</td>
<td>15</td>
<td>utf-8</td>
<td>04Apr2016 17:05:10</td>
<td>04Apr2016 17:05:10</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node</th>
<th>Number of Blocks</th>
<th>Active Blocks</th>
<th>Rows</th>
<th>Fixed Data Size</th>
<th>Variable Data Size</th>
<th>Blocks Mapped</th>
<th>Memory Mapped</th>
<th>Blocks Unmapped</th>
<th>Memory Unmapped</th>
<th>Blocks Allocated</th>
<th>Memory Allocated</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>1</td>
<td>1</td>
<td>428</td>
<td>68480</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>68480</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Column</th>
<th>Label</th>
<th>Type</th>
<th>Length</th>
<th>Formatted Length</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make</td>
<td>char</td>
<td>13</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>char</td>
<td>40</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>char</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Origin</td>
<td>char</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DriveTrain</td>
<td>char</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSRP</td>
<td>double</td>
<td>8</td>
<td>8</td>
<td></td>
<td>DOLLAR</td>
</tr>
<tr>
<td>Invoice</td>
<td>double</td>
<td>8</td>
<td>8</td>
<td></td>
<td>DOLLAR</td>
</tr>
<tr>
<td>EngineSize</td>
<td>Engine Size (L)</td>
<td>double</td>
<td>8</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Cylinders</td>
<td>double</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Horsepower</td>
<td>double</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPG_City</td>
<td>MPG (City)</td>
<td>double</td>
<td>8</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>MPG_Highway</td>
<td>MPG (Highway)</td>
<td>double</td>
<td>8</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Weight</td>
<td>double</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weight (LBS)</td>
<td>double</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wheelbase</td>
<td>Wheelbase (IN)</td>
<td>double</td>
<td>8</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Length</td>
<td>Length (IN)</td>
<td>double</td>
<td>8</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

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 4, “CASUTIL Procedure,” on page 53.

• For more examples of using the CASUTIL procedure to access and save data, see “Accessing Data” in SAS Cloud Analytic Services: Accessing and Manipulating Data.

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

```sas
proc casutil; /* 1 */
   load data=sashelp.cars;
   contents casdata="cars";
quit;

libname mycas cas; /* 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.

The LIBNAME statement for the CAS engine creates a CAS libref. 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.

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.

The PRINT procedure prints the table Mycas.CarsMiles.
Results

**Output 5.6**  PROC MDSUMMARY Output: City Mileage for Make, Model, and Type

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th><em>Column</em></th>
<th><em>N Obs</em></th>
<th><em>Min</em></th>
<th><em>Max</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>VPG_City</td>
<td>1</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
<td>Sedan</td>
<td>VPG_City</td>
<td>1</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>Audi</td>
<td>A6 2.7 Turbo Quattro 4dr</td>
<td>Sedan</td>
<td>VPG_City</td>
<td>1</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>VPG_City</td>
<td>1</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>Audi</td>
<td>TT 1.8 convertible 2dr (coupe)</td>
<td>Sports</td>
<td>VPG_City</td>
<td>1</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>Audi</td>
<td>TT 3.2 coupe 2dr (convertible)</td>
<td>Sports</td>
<td>VPG_City</td>
<td>1</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>BMW</td>
<td>325i 4dr</td>
<td>Sedan</td>
<td>VPG_City</td>
<td>1</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>BMW</td>
<td>330Ci 2dr</td>
<td>Sedan</td>
<td>VPG_City</td>
<td>1</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>BMW</td>
<td>330xi 4dr</td>
<td>Sedan</td>
<td>VPG_City</td>
<td>1</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>10</td>
<td>BMW</td>
<td>C20i 4dr</td>
<td>Sedan</td>
<td>VPG_City</td>
<td>1</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>11</td>
<td>BMW</td>
<td>M3 convertible 2dr</td>
<td>Sports</td>
<td>VPG_City</td>
<td>1</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>12</td>
<td>BMW</td>
<td>M3 coupe 2dr</td>
<td>Sports</td>
<td>VPG_City</td>
<td>1</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>13</td>
<td>BMW</td>
<td>Z4 convertible 2.5i 2dr</td>
<td>Sports</td>
<td>VPG_City</td>
<td>1</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>14</td>
<td>Buick</td>
<td>Lesabre Custom 4dr</td>
<td>Sedan</td>
<td>VPG_City</td>
<td>1</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>15</td>
<td>Buick</td>
<td>Lesabre Limited 4dr</td>
<td>Sedan</td>
<td>VPG_City</td>
<td>1</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>16</td>
<td>Buick</td>
<td>Regal GS 4dr</td>
<td>Sedan</td>
<td>VPG_City</td>
<td>1</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>17</td>
<td>Buick</td>
<td>Rendezvous CX</td>
<td>SUV</td>
<td>VPG_City</td>
<td>1</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>18</td>
<td>Cadillac</td>
<td>SRX V6</td>
<td>SUV</td>
<td>VPG_City</td>
<td>1</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>19</td>
<td>Chevrolet</td>
<td>Cavalier 4dr</td>
<td>Sedan</td>
<td>VPG_City</td>
<td>1</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>20</td>
<td>Chevrolet</td>
<td>Malibu LT 4dr</td>
<td>Sedan</td>
<td>VPG_City</td>
<td>1</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>21</td>
<td>Chevrolet</td>
<td>Silversado SS</td>
<td>Truck</td>
<td>VPG_City</td>
<td>1</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>22</td>
<td>Chevrolet</td>
<td>Tahoe LT</td>
<td>SUV</td>
<td>VPG_City</td>
<td>1</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>23</td>
<td>Chrysler</td>
<td>Concorde LX 4dr</td>
<td>Sedan</td>
<td>VPG_City</td>
<td>1</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>24</td>
<td>Chrysler</td>
<td>Concorde LXI 4dr</td>
<td>Sedan</td>
<td>VPG_City</td>
<td>1</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>25</td>
<td>Chrysler</td>
<td>PT Cruiser 4dr</td>
<td>Sedan</td>
<td>VPG_City</td>
<td>1</td>
<td>22</td>
<td>22</td>
</tr>
</tbody>
</table>

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 4, “CASUTIL Procedure,” on page 53.

- For more examples of using the CASUTIL procedure to access and save data, see “Accessing Data” in *SAS Cloud Analytic Services: Accessing and Manipulating Data*. 
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.

libname mycas cas; /*
libname mycas cas; */

proc format casfmtlib='fmtlib' ; /*
proc format casfmtlib='fmtlib' ; */
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;

libname mycas cas; /*
libname mycas cas; */

data mycas.cake; /*
data mycas.cake; */
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; /*
proc mdsummary data=mycas.cake; */
var TasteScore;
    groupby flavor / out=mycas.flav; /*
    groupby flavor / out=mycas.flav; */
    groupby flavor age / out=mycas.flag; /*
run;
proc print data=mycas.flav; /*7*/
  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 **LIBNAME** statement for the CAS engine creates a CAS engine libref. 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.

2 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.

3 The DATA step creates the input data set. This DATA step runs in the SAS client session 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.

4 The MDSUMMARY procedure computes summary statistics for cake tasting scores.

5 The first GROUPBY statement with the OUT= option creates an in-memory table named Mycas.Flav that is grouped by Flavor.

6 The second GROUPBY statement with the OUT= option creates an in-memory table named Mycas.Flag that is grouped by Flavor and Age.

7 The PRINT procedure prints the output data sets.

**Results**

**Output 5.7** PROC PRINT Output: Cake Flavors and Participant’s Age Grouped by Flavor

<table>
<thead>
<tr>
<th>Flavor</th>
<th><em>Column</em></th>
<th><em>NObs</em></th>
<th><em>Min</em></th>
<th><em>Max</em></th>
<th><em>Mean</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chocolate</td>
<td>TasteScore</td>
<td>1</td>
<td>74</td>
<td>74</td>
<td>74</td>
</tr>
<tr>
<td>Other Flavor</td>
<td>TasteScore</td>
<td>9</td>
<td>72</td>
<td>92</td>
<td>81.4444444444</td>
</tr>
<tr>
<td>Vanilla</td>
<td>TasteScore</td>
<td>4</td>
<td>72</td>
<td>91</td>
<td>81.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>73</td>
<td>94</td>
<td>82.1686666667</td>
</tr>
</tbody>
</table>
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 4, “CASUTIL Procedure,” on page 53.

For more examples of using the CASUTIL procedure to access and save data, see “Accessing Data” in SAS Cloud Analytic Services: Accessing and Manipulating Data.

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.

```sas
proc casutil;        /* 1 */
   load data=sashelp.cars;
   contents casdata="cars";
quit;

libname mycas cas;   /* 2 */
proc mdsummary data=mycas.cars;          /* 3 */
   var mpg_highway;
   groupby origin type / out=mycas.mpghw_sum;
run;
ods graphics / width=4in;
```
title "Summarized Highway MPG";
proc sgpipeline data=mcas.mpghw_sum;   /* 4 */
    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 **LIBNAME** statement for the CAS engine creates a CAS engine libref. 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.

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 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.

4 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**

The SGPANEL procedure generates the following graph:

**Output 5.9  Graph Output**

![Graph Output](image)

**Additional Information**

- Caslibgs provide a way to organize in-memory tables and an associated data source. They also provide a way to apply access controls to data. In this example, the default
caslib CASUSER is being used, so no CASLIB statement is needed. The in-memory table Mycas.Mpghw_Sum 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 4, “CASUTIL Procedure,” on page 53.

- For more examples of using the CASUTIL procedure to access and save data, see “Accessing Data” in SAS Cloud Analytic Services: Accessing and Manipulating Data.

- For information about the SGPANEL procedure, see “SGPANEL” in SAS ODS Graphics: Procedures Guide.
Chapter 6
Platform Data Sources

Data Redundancy

The following table shows how several factors interact with respect to data redundancy. Data redundancy applies to distributed servers only.

Table 6.1  Data Redundancy by Data Access Method, Data Source, and File Type

<table>
<thead>
<tr>
<th>Data Access Method</th>
<th>Caslib Data Source</th>
<th>Redundancy</th>
<th>File Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA step and PROC CASUTIL; LOAD DATA=</td>
<td>The caslib data source isn't a factor for these data access methods.</td>
<td>Not applicable. Based on COPIES= when the file is loaded.</td>
<td>SAS Data Sets</td>
</tr>
</tbody>
</table>
Dictionary

HDFS Data Source

Specifies a Hadoop Distributed File System directory for loading and saving files that the SAS Cloud Analytic Services controller can access. SAS Cloud Analytic Services must be co-located with the Hadoop cluster to use this data source type.

**Applies to:** Distributed servers only, CASLIB statement

**Example:** Add a caslib to access files from the `/vapublic` directory in HDFS.

```sas
Caslib public datasource=(srctype="hdfs") path="/vapublic";
proc casutil incaslib="public" outcaslib="public";
list files;
quit;
```

**Syntax**

**Data Source Arguments**

`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="HDFS"
specifies that the data source is an HDFS directory that is co-located with SAS Cloud Analytic Services.

Requirement The SRCTYPE= argument is required.

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.

Applies to: Distributed servers only, CASLIB statement
Example: Add a caslib to access files from the /data01 directory on the controller.

caslib dnfsds datasource=(srctype="dnfs") path="/net/fileserver/";

proc casutil incaslib="dnfsds" outcaslib="dnfsds";
   list files;
quit;

Syntax

Data Source Arguments

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= argument is required.

Details

DNFS is an acronym 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 provides a good alternative for deployments where the server is not co-located with Hadoop and yet must provide similar capabilities.

DNFS can also be used to access NFS-mounted directories from standard UNIX or 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.

**SAS LASR Analytic Server**

Specifies the connection options for loading data from the SAS LASR Analytic Server into SAS Cloud Analytic Services.

**Applies to:** CASUTIL procedure, CASLIB statement

**Examples:** Add a caslib to access data from a SAS LASR Analytic Server.

```sas
caslib publiclasr datasource=(
   srctype="lasr"
   server="gridhost.example.com"
   port=10050
   signer="https://webserver.example.com/SASLASRAuthorization"
   username="sasdemo"
   password="secret"
);
```

Load a table from a SAS LASR Analytic Server into SAS Cloud Analytic Services.

```sas
proc casutil incaslib="publiclasr";
   load casdata="epa_cars"
      importoptions=(filetype="lasr" varchars="true");
run;
```

**Syntax**

**Data Source Arguments**

**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.

**PORT=**integer

specifies the network port that the SAS LASR Analytic Server listens on.

**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. This is specified in the form SIGNER="https://server.example.com/SASLASRAuthorization".

**SRCTYPE=**"LASR"

specifies that the data source is a SAS LASR Analytic Server.

**Requirement** The SRCTYPE= argument is required.
**USERNAME**="user-ID"

specifies an identity that is authorized to access data in the SAS LASR Analytic Server.

**File Type Arguments**

**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 parameter in the COMPPGM option.

**FILETYPE**="LASR"

specifies the file type.

**Requirement** The FILETYPE= argument 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, then parallel data transfer is not possible and that is equivalent to NONE.

**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, then 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.

**Default** FALLBACK

**PRESERVEORDER**=TRUE | FALSE

when set to True, the 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 this parameter is used.

**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 where expression for subsetting the input data.
Details

Table 6.2  Summary of Arguments for SAS LASR Analytic Server

<table>
<thead>
<tr>
<th>Argument</th>
<th>Valid Data Source Options in the CASLIB Statement</th>
<th>Valid Import Options in the CASUTIL Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPPGM=</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>COMPVARS=</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>FILETYPE=</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>METALIB=</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>PARALLELMODE=</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>PASSWORD=</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>PORT=</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>PRESERVEORDER=</td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>SRCTYPE=</td>
<td>Required</td>
<td></td>
</tr>
<tr>
<td>SERVER=</td>
<td>Required</td>
<td></td>
</tr>
<tr>
<td>SIGNER=</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>USERNAME=</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>VARCHARS=</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>VARS=</td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>WHERE=</td>
<td>●</td>
<td></td>
</tr>
</tbody>
</table>

Example

This example shows how to load a table from a SAS LASR Analytic Server and use the VARS= options. The sample Epa_cars data is available from the SAS Visual Analytics 7.3 documentation page at http://support.sas.com/documentation/onlinedoc/va/index.html.

```r
 caslib valasr datasource=(srctype="lasr"
  /* ...data source options... */
);

cas casauto sessopts=(caslib="valasr"); /* */
```
proc casutil;
  list files; /* 2 */

load casdata="epa_cars" casout="epa_cars"
importoptions=
  filetype="lasr"
  varschars="true"
  vars=(*model_year* "vehicle_manufacturer_name")  /* 3 */
);

  contents casdata="epa_cars";
quit;

1 The CAS statement is used to set the active caslib for the Casauto session to Valasr explicitly.
2 The LIST FILES statement shows the tables that are in SAS LASR Analytic Server.
3 The VARS= option is used to subset the columns that are read from SAS LASR Analytic Server and loaded into SAS Cloud Analytic Services.

Path Data Source

Specifies a server-side directory for loading and saving files that the SAS Cloud Analytic Services controller can access.

Applies to: CASLIB statement

Example: Add a caslib to access files from the /data01 directory on the controller.

    caslib pathds datasource=(srctype="path") path="/data01";
    proc casutil incaslib="pathds" outcaslib="pathds";
      list files;
    quit;

Syntax

Data Source Arguments

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= argument is required.
Chapter 7
Platform File Types

Dictionary

Delimited Files (CSV)
Specifies the file type options for loading data from delimited files.

**Applies to:** CASUTIL procedure

**Example:** Load a Latin1 encoded CSV file into SAS Cloud Analytic Services.

```sas
proc casutil;
  load casdata="iris.csv"
    importoptions=(
      filetype="csv" encoding="latin1"
    );
run;
```

**Syntax**

**File Type Arguments**

**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= argument 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.

**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 parameter 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**
Delimited files can be read from caslibs with the following data source types:

- DNFS
- HDFS
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.

File 7.1  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 7.1  Load a CSV File and Specify Column Names

```sas
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.

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 7.1 Column Information from the CONTENTS Statement

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Length</th>
<th>Formatted Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>sexe</td>
<td>varchar</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>nom</td>
<td>varchar</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>âge</td>
<td>double</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>la taille</td>
<td>double</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>poids</td>
<td>double</td>
<td>8</td>
<td>12</td>
</tr>
</tbody>
</table>

Output 7.2 Three Rows from the Class_Fr Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>sexe</th>
<th>nom</th>
<th>âge</th>
<th>la taille</th>
<th>poids</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Masculin</td>
<td>André</td>
<td>14</td>
<td>69.0</td>
<td>112.5</td>
</tr>
<tr>
<td>2</td>
<td>Masculin</td>
<td>Benoît</td>
<td>14</td>
<td>63.5</td>
<td>102.5</td>
</tr>
<tr>
<td>3</td>
<td>Masculin</td>
<td>Kévin</td>
<td>12</td>
<td>57.3</td>
<td>83.0</td>
</tr>
</tbody>
</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**

**File Type Arguments**

**FILETYPE="HDAT"**

specifies the file type.

Requirement The FILETYPE= argument is required.

**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.
Details

SASHDAT files can be read from caslibs with the following data source types:

- DNFS
- HDFS
- PATH

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 48.
Chapter 8
Data Connectors

Working with SAS Data Connectors

To access data, the administrator must create and configure data connectors for SAS Cloud Analytic Services. Data connectors contain connection information and data-source specifics to connect with data sources such as Oracle or SAS data sets.

Data connectors contain information that identifies the location of tables that reside in your data source. If a data source does not support native catalogs, SAS Cloud Analytic Services lets you define a logical catalog name to use as an SQL identifier. This allows unique identification of each data source when performing heterogeneous operations.

There are two types of data connectors: serial and parallel. Parallel data connectors are used with the SAS Embedded Process and must be licensed separately.

You must 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 the CAS server.

Data connectors can also contain optional information to control SAS Cloud Analytic Services data source behavior. Data connectors form the foundation for connectivity to a third-party database. 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. SAS Cloud Analytic Services respects authorizations that are defined and enforced by a third-party database. Authorizations that are defined on a third-party database overrule permissions and privileges that are set in 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 "DATASOURCE="
Table 8.1 Data Source Types and Options

<table>
<thead>
<tr>
<th>SRCTYPE= Type</th>
<th>Option Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIVE</td>
<td>“SAS Data Connector to Hive, SAS Data Connect Accelerator to Hive” on page 109</td>
<td>caslib hivelib desc=&quot;Hive Caslib&quot; datasource=(srctype=&quot;hive&quot;, dataTransferMode=&quot;parallel&quot;, hadoopjarpath=&quot;/data/cdh54/sdm/lib&quot;, hadoopconfigdir=&quot;/data/cdh54/sdm/conf&quot;, username=&quot;hiveuser&quot;, server=&quot;hive01.example.com&quot;, schema=&quot;default&quot;);</td>
</tr>
<tr>
<td>ODBC</td>
<td>“SAS Data Connector to ODBC” on page 115</td>
<td>caslib odbccaslib desc=&quot;My ODBC caslib&quot; datasource=(srctype=&quot;odbc&quot; username=&quot;user1&quot; password=&quot; password1&quot; database=&quot;dbodbc&quot; catalog=&quot;*&quot; );</td>
</tr>
<tr>
<td>ORACLE</td>
<td>“SAS Data Connector to Oracle” on page 121</td>
<td>caslib oraclecaslib desc=&quot;Oracle Caslib&quot; datasource=(srctype=&quot;oracle&quot;, dataTransferMode=&quot;serial&quot;, username=&quot;user1&quot;, password=&quot;******&quot;, path=&quot;/machine.lnx.com:5570/exadat&quot;);</td>
</tr>
</tbody>
</table>
SAS Data Connector to Hive, SAS Data Connect Accelerator to Hive

Specifies the connection options for loading data from Hive into SAS Cloud Analytic Services.

**Applies to:** CASLIB statement, PROC CASUTIL: LOAD statement

**Restriction:** The Hive ARRAY, MAP, STRUCT, UNION, or BOOLEAN data types are not supported.

**Requirement:** JAVA_HOME must point to the location of the installed Java 8 JRE, and LIBJVM.SO must be specified in LD_LIBRARY_PATH. Modify the following example to match the layout of your SAS Cloud Analytic Services cluster.

```bash
export JAVA_HOME=/usr/java/latest/jre
export LD_LIBRARY_PATH=$JAVA_HOME/lib/amd64/server:$LD_LIBRARY_PATH
```

**See:** For configuration instructions, see the SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS.

**Examples:** Add a caslib to access data from Hive.

```sas
s:addCaslib{lib="hvhad", datasource={srcType="hive",
dataTransferMode="<value>",
hadoopJarPath="<Hadoop-jar-file-path>",
hadoopConfigDir="<Hadoop-config-files-path>",
hdfsTempDir="/temp",
username="<userid>",
server="<Hive-server-name>",
schema="<schema-name>",
dbmaxText = 100
}
```
Load a table from Hive data.

```plaintext
proc casutil;
  list files
  incaslib="hivecaslib";
load casdata="dbtable",
  incaslib="hivecaslib";
quit;
```

**Syntax**

**Required Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRCTYPE=&quot;HIVE&quot;</td>
<td>specifies that the data source is Hive.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>HADOOPCONFIGDIR=&quot;configuration-files-directory&quot;</td>
<td>specifies the Hadoop configuration files directory, which is the one that is obtained by running the Hadoop tracer tool on the target Hadoop cluster.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>HADOOPJARPATH=&quot;jar-files-path&quot;</td>
<td>specifies one or more paths to the Hadoop JAR files, which are the JAR files that are obtained by running the Hadoop tracer tool on the target Hadoop cluster. These files are delimited by colons for Linux.</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>SERVER=&quot;server-name&quot;</td>
<td>specifies the Hadoop server name that runs the Hive service.</td>
<td>&quot;&quot;</td>
</tr>
</tbody>
</table>

**Optional Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFFERSIZE=integer</td>
<td>specifies in bytes the size of an input buffer to use for each group of records that is retrieved. Increasing the size might result in better performance with a tradeoff of increased memory usage.</td>
<td>1048576</td>
</tr>
<tr>
<td>CATALOG=&lt;catalog-name&gt;</td>
<td>specifies the TKTS catalog name.</td>
<td>caslib name</td>
</tr>
</tbody>
</table>

Valid in CASLIB statement

**PROC** CASUTIL, **LOAD** CASDATA=, **LIST FILES** OPTIONS=, **CONTENTS** CASDATA= statements

**CHARMULTIPLIER=double**

specifies a common option for the Multiplier for the character column length for nonwide character types.
This does not apply to Hive because all Hive tables are UTF-8.

**DATATRANSFERMODE=** *enum*

specifies the mode of data transfer. Here are the valid values:

- **auto**: specifies to try parallel (embedded) processing first. If this fails, an error is issued and serial processing is then attempted.
- **parallel**: specifies to always use only parallel (embedded) processing. To use this option, you must have a licensed copy of SAS Data Connect Accelerator to Hive.
- **serial**: specifies to use serial processing.

**Alias**

`dataTransfer, dtm`

**Default**

`serial`

**DBMAXTEXT=** *integer*

specifies the length of a string that all data string types should be interpreted as. For example, if `dbMaxText=100`, all Hive string data types are loaded as `VARCHAR(100)`.

**Default**

`32767`

**Range**

`1` (minimum)

**HDFSTEMPDIR=** "*string"

specifies the Hadoop HDFS directory to use to store temporary data.

**Default**

`/tmp`

**PASSWORD=** "*string"

specifies the password for the identity in the `USERNAME=` option.

**Alias**

`pass, pwd`

**Default**

"" (zero-length string)

**PORT=** *integer*

specifies the Hive JDBC port number.

**Default**

`10000`

**Range**

`1–65535`

**PROPERTIES=** "*string"

specifies a free-form value for Hive JDBC properties. The value is appended to the JDBC connection URI. You can use it to override default Hive behaviors.

**Default**

`none`

**READBUFF=** *integer*

specifies the row array size to use when fetching in serial data transfer mode.

**Default**

`none`
SCHEMA="string"
    specifies the name of the Hive database.
    Default  "default"

STATUSINTERVAL=integer
    specifies to output an immediate message to the client when a node adds n buffers to
    the table, where n is the value of this parameter.
    Default  0 (no rows)

URI="string"
    specifies a free-form JDBC URI to use as the Hive JDBC connection URI to
    override the default URI. If you use this parameter, parameters that alter the JDBC
    URI (such as PROPERTIES=) are ignored.
    Default  none

USERNAME="user-ID"
    specifies an identity that is authorized to access data in Hive.
    Alias    uid

Details

Summary of Data Connector Usage and Precedence

Load table values override data source values. Column and file information values
(DATATRANSFERMODE=, PASSWORD=, SCHEMA=, and USERNAME=) override
data source values.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Valid Data Source Options in the CASLIB Statement</th>
<th>Valid Import Options in the CASUTIL Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFFERSIZE=</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>CHARMULTIPLIER=</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>DATATRANSFERMODE=</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>DBMAXTEXT=</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>HADOOPCONFIGDIR=</td>
<td></td>
<td>required</td>
</tr>
<tr>
<td>HADOOPJARPATH=</td>
<td></td>
<td>required</td>
</tr>
<tr>
<td>HDFSTEMPDIR=</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>PASSWORD=</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>PORT=</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>PROPERTIES=</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>READBUFF=</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>Argument</td>
<td>Valid Data Source Options in the CASLIB Statement</td>
<td>Valid Import Options in the CASUTIL Procedure</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>SCHEMA=</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>SERVER=</td>
<td>required</td>
<td></td>
</tr>
<tr>
<td>SRCTYPE=</td>
<td>required</td>
<td></td>
</tr>
<tr>
<td>STATUSINTERVAL=</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>URI=</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>USERNAME=</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

**Examples**

**Example 1: Add a CASLIB Action and Load a Table from Hive Data (in SAS)**

This step adds a CASLIB action.

```sas
   caslib mycaslib desc="Hive Datafeeder Caslib"
       sessref=mycassession
       datasource=(srctype="hive",
                   hadoopJarPath="Hadoo-jar-file-path",
                   hadoopConfigDir="Hadoop-config-files-path",
                   username="user-id",
                   schema="schema-name",
                   server="Hive-server-hostname", dataTransferMode="parallel");
```

This step lets you access data from Hive.

```sas
   proc casutil sessref=mycassession;
       load casdata="mydataset" incaslib="mycaslib";
   run;
```

Here is the output from these steps.

```
NOTE: Cloud Analytic Services made the external data from mydataset available as table MYDATASET in caslib MYCASLIBIN.
```

**Example 2: Obtain Information about Table Columns (in SAS)**

Here is the output from the contents of PROC CASUTIL.
Example 3: Obtain Information about Files

This step lets you access information about files.

    proc casutil sessref=mycassession;
    list tables;

    CAS proc casutil contents output                      2
    08:43 Wednesday, April 6, 2016

    CAS proc casutil contents output                      3
    08:43 Wednesday, April 6, 2016

    CAS proc casutil contents output                      4
    08:43 Wednesday, April 6, 2016

    NOTE: Cloud Analytic Services processed the combined requests in 0.039445 seconds.
run;
quit;

Here is the output from the contents of PROC CASUTIL.

```
Caslib Information
    Library
    Source Type
    Description
    DataTransferMode
    Session local
    Active
    Personal
    Server
    HadoopConfigDir
    HadoopJarPath
    Schema

Caslib Information
    MYCASLIB
    hive
    "Hive Datafeeder Caslib"
    Parallel
    Yes
    Yes
    No
    hive-hostname
    hadoop-config-path
    hadoop-jar-path
    schema

Table Information for Caslib MYCASLIB
    Table    Number        Number    NLS
    Name        of Rows    of Columns    encoding    Created
    MYDATASET         2             2       utf-8    06Apr2016:08:44:04

Table Information for Caslib MYCASLIB
    Table      Last           Promoted  Repeated            Source  Source
    Name       Modified          Table     Table      View  Name    Caslib
    MYDATASET  6Apr2016:08:44:04    No        No        No    uty1  MYCASLIB

Table Information for Caslib MYCASLIB
    Table
    Name  Compressed
    MYDATASET      No
```
**Examples:** Establish a connection to ODBC data. (Note: The GLOBAL option is restricted to administrators.)

```plaintext
caslib odbccaslib desc="SQLviaODBCtoCaslib"
    datasource=(srctype="odbc"
        username="user1"
        password="password1"
        database="dbodbc"
        catalog="*")
global;
```

Load ODBC data.

```plaintext
proc casutil;
    list tables incaslib="odbccaslib";
    load casdata="DBtable" incaslib="odbccaslib"
        casout="myData_from_odbcaslib";
    list tables incaslib="casuser";
    contents casdata="class_from_odbcaslib" incaslib="casuser";
quit;
```

**Syntax**

**Data Connector Arguments for ODBC**

- **CATALOG= "<catalog-name>"**
  - Specifies the TKTS catalog name.
  - To access the Microsoft SQL Server, set CATALOG="*".
  - **Valid in** CASLIB statement
  - **Default** caslib name

- **charMultiplier= <integer>**
  - Specifies an increase to the width of fixed-byte-width character columns. The number of bytes that are needed for multi-byte characters depends on the characters in a string. For double-byte character sets, set charMultiplier=2.
  - This value overrides a value of charMultiplier that was set in the CASLIB statement or in the addCaslib action.
  - **Valid in** CASLIB statement
  - **Default** 1
  - **Range** 1–5

- **CLIENT_ENCODING= "<encoding-value>"**
  - Specifies a client encoding type. Source data is transcoded from this encoding to a UTF-8 encoding.
Valid in CASLIB statement

PROC CASUTIL: LOAD CASDATA= statement

Default none

CONOPTS="<connection-options>"
specifies optional connection options that you can pass to the underlying driver.

Valid in CASLIB statement

PROC CASUTIL: LOAD CASDATA=, LIST FILES OPTIONS=, and CONTENTS CASDATA= statements

Default none

DM_UNICODE="<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.

Valid in CASLIB statement

PROC CASUTIL: LOAD CASDATA= statement

Default UTF-8

ODBC_DSN="<DSN-name>"
specifies the data source name for your DBMS.

Valid in CASLIB statement [Required]

Aliases DATABASE=

DB=

password="<password>"
specifies the DBMS password for the given user.

Valid in CASLIB statement [required]

CAS actions: addCaslib [required]

PROC CASUTIL: CONTENTS CASDATA=, LIST FILES OPTIONS=, LOAD CASDATA= statements

Aliases Pwd=

Pass=

Default none

READBUFF="<number-of-rows>"
specifies the number of rows to fetch per block of data retrieved.

Valid in CASLIB statement
PROC CASUTIL: LOAD CASDATA= statement

<table>
<thead>
<tr>
<th>Default</th>
<th>auto-calculated based on row size</th>
</tr>
</thead>
</table>

**schema**="<schema-name>" "
specifies the schema to use for the connection to the DBMS.

<table>
<thead>
<tr>
<th>Valid in</th>
<th>CASLIB statement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CAS actions: addCaslib, fileInfo, columnInfo, loadTable</td>
</tr>
</tbody>
</table>

PROC CASUTIL: CONTENTS CASDATA=, LIST FILES, LOAD CASDATA= statements

<table>
<thead>
<tr>
<th>Default</th>
<th>&quot;&quot; (empty string)</th>
</tr>
</thead>
</table>

**SRCTYPE**="ODBC"
specifies that the data source for the caslib is accessed through ODBC.

<table>
<thead>
<tr>
<th>Valid in</th>
<th>CASLIB statement [Required]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Default</th>
<th>none</th>
</tr>
</thead>
</table>

**userName**="<user-name>" "
specifies the user name.

<table>
<thead>
<tr>
<th>Valid in</th>
<th>CASLIB statement [required]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CAS actions: addCaslib [required], columnInfo</td>
</tr>
</tbody>
</table>

PROC CASUTIL: CONTENTS CASDATA=, LOAD CASDATA= statements

<table>
<thead>
<tr>
<th>Aliases</th>
<th>UID=</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>USER=</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Default</th>
<th>none</th>
</tr>
</thead>
</table>

| Requirement | You must specify either userName= and Password= or authDomain= in the CASLIB statement or for the addCaslib action. To change credentials in other statements or actions, you must specify userName= and Password= values or an authDomain= value. |

### Details

#### Usage Summary
The following table shows a summary of the arguments for the ODBC data connector. When you specify an argument in multiple statements, the following rules determine the precedence of argument values:

- When a value is supplied for both, the value for the LOAD CASDATA= statement in PROC CASUTIL overrides the value that is set in the CASLIB statement.

- A value that you specify in the CONTENTS statement in PROC CASUTIL overrides the value that you specify in the CASLIB statement or in the LOAD CASDATA= statement.
• A value that you specify in the LIST FILES statement in PROC CASUTIL overrides the value that you specify in the CASLIB statement, the LOAD CASDATA= statement, or the CONTENTS statement.

**Table 9.1  Summary of ODBC Data Connector Argument Usage**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Valid Data Source Options in the CASLIB Statement</th>
<th>Valid in the LOAD CASDATA= Statement in PROC CASUTIL</th>
<th>Valid in the CONTENTS statement in PROC CASUTIL</th>
<th>Valid in the LIST FILES statement in PROC CASUTIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATALOG=</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>CLIENT_ENCODING=</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>CONOPTS=</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DM_UNICODE=</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ODBC_DSN=</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PASSWORD=</td>
<td>Required</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>PATH=</td>
<td></td>
<td>Required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>READBUFF=</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCHEMA=</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>SRCTYPE=</td>
<td>Required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USERNAME=</td>
<td>Required</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

**Data Type Conversions**

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:* 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 SAS reads are rounded to meet this condition. When you use a large numeric value in a WHERE clause, this rounding can cause unexpected results, such as not selecting desired rows. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data.

**Table 9.2  Data Type Conversions When Loading Data from ODBC into CAS**

<table>
<thead>
<tr>
<th>ODBC Data Type</th>
<th>CAS Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL_CHAR</td>
<td>CHAR</td>
</tr>
<tr>
<td>ODBC Data Type</td>
<td>CAS Data Type</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>SQL_VARCHAR</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>SQL_LONGVARCHAR</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>SQL_WCHAR</td>
<td>CHAR</td>
</tr>
<tr>
<td>SQL_WVARCHAR</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>SQL_WLONGVARCHAR</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>SQL_BINARY</td>
<td>CHAR</td>
</tr>
<tr>
<td>SQL_VARBINARY</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>SQL_LONGVARBINARY</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>SQL_NUMERIC</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>SQL_DECIMAL</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>SQL_INTEGER</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>SQL_SMALLINT</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>SQL_FLOAT</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>SQL_REAL</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>SQL_DOUBLE</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>SQL_BIGINT</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>SQL_TINYINT</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>SQL_BIT</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>SQL_TYPE_DATE</td>
<td>DOUBLE (formatted as DATE)</td>
</tr>
<tr>
<td>SQL_TYPE_TIME</td>
<td>DOUBLE (formatted as TIME)</td>
</tr>
<tr>
<td>SQL_TYPE_TIMESTAMP</td>
<td>DOUBLE (formatted as DATETIME)</td>
</tr>
</tbody>
</table>
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 through 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 35.

```
caslib odbccaslib desc="SQLviaODBCtoCaslib"
    datasource=(srctype="odbc"
        username="user1"
        password="password1"
        database="dbodbc"
        catalog="*")
    global;
```

Example 2: Load Data from an External Database into SAS Cloud Analytic Services

```
proc casutil;
    list tables incaslib="odbcaslib"; /* 1 */
    load casdata="myDBdata" incaslib="odbcaslib" outcaslib="casuser"
        casout="myData_from_odbcaslib"; /* 2 */
    list tables incaslib="casuser"; /* 3 */
    contents casdata="class_from_odbcaslib" incaslib="casuser"; /* 4 */
run;
```

1 List the tables in odbccaslib prior to loading your data.
2 Load the table myDBdata from an external database into caslib Casuser. Call the new table myData_from_oraclecaslib.
3 List the tables in odbccaslib again to see the newly created table, myData_from_oraclecaslib, that you loaded.
4 List information about the newly loaded table, including column names, data types, and so on.

SAS Data Connector to Oracle

Specifies the options to use when loading data from Oracle into SAS Cloud Analytic Services.

- **Applies to:** CASLIB statement, PROC CASUTIL: LOAD statement
- **Examples:** Establish a connection between your Oracle database and SAS Cloud Analytic Services.

```
caslib oraclecaslib desc='Oracle Caslib'
    datasource=(srctype='oracle'
        username='user1'
        password='******'
        path="/machine.lnx.com:1521/exadat");
```
Overriding the user and password values.

```sas
proc casutil;
  load casdata="$upcase(mycas.orexamp)" options={
    username='user5'
    password='******'};
```

## Syntax

### Data Connector Options for Oracle

**ORA_ENCODING="<encoding-name>"**

specifies the encoding of the data in the Oracle database. This value is independent of the NSL_LANG environment variable setting.

Valid values include LATIN1, WLATIN1, and UNICODE. Set this value to UNICODE when you are loading non-Latin1 data.

<table>
<thead>
<tr>
<th>Valid in</th>
<th>CASLIB statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>none</td>
</tr>
</tbody>
</table>

**password="<password>"**

specifies the DBMS password for the given user.

<table>
<thead>
<tr>
<th>Valid in</th>
<th>CASLIB statement [required]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAS actions:</td>
<td>addCaslib [required]</td>
</tr>
<tr>
<td>PROC CASUTIL: CONTENTS CASDATA=, LIST FILES OPTIONS=, LOAD CASDATA= statements</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Aliases</th>
<th>Pwd=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>none</td>
</tr>
</tbody>
</table>

**PATH="<Oracle-table>"**

specifies the path for the Oracle source table.

<table>
<thead>
<tr>
<th>Valid in</th>
<th>PROC CASUTIL: LOAD statement [Required]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>none</td>
</tr>
</tbody>
</table>

**schema="<schema-name>"**

specifies the schema to use for the connection to the DBMS.

<table>
<thead>
<tr>
<th>Valid in</th>
<th>CASLIB statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAS actions:</td>
<td>addCaslib, fileInfo, columnInfo, loadTable</td>
</tr>
<tr>
<td>PROC CASUTIL: CONTENTS CASDATA=, LIST FILES, LOAD CASDATA= statements</td>
<td></td>
</tr>
</tbody>
</table>

122  Chapter 9 • Data Connector Reference
SRCTYPE="ORACLE"

specifies that the data source is an Oracle database.

Valid in CASLIB statement [Required]

Default none

userName=" <user-name> "

specifies the user name.

Valid in CASLIB statement [required]

CAS actions: addCaslib [required], columnInfo

PROC CASUTIL: CONTENTS CASDATA=, LOAD CASDATA= statements

Aliases UID=

USER=

Default none

Requirement You must specify either userName= and Password= or authDomain= in the CASLIB statement or for the addCaslib action. To change credentials in other statements or actions, you must specify userName= and Password= values or an authDomain= value.

Details

Restrictions on Manipulating Oracle Data

When you load data from Oracle into SAS Cloud Analytic Services, the following Oracle data types are not supported:

- Array
- Boolean
- Map
- Struct
- Union

Summary of Data Connector Usage and Precedence

<table>
<thead>
<tr>
<th>Valid Data Source Options in the CASLIB Statement</th>
<th>Valid in the LOAD CASDATA= Statement in PROC CASUTIL*</th>
<th>Valid in the CONTENTS statement in PROC CASUTIL</th>
<th>Valid in the CONTENTS statement in PROC CASUTIL</th>
<th>Valid in the LIST FILES statement in PROC CASUTIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORA_ENCODING=</td>
<td>⬤</td>
<td>⬤</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Valid Data Source Options in the CASLIB Statement

<table>
<thead>
<tr>
<th>Option</th>
<th>Valid in the LOAD CASDATA= Statement in PROC CASUTIL</th>
<th>Valid in the CONTENTS statement in PROC CASUTIL</th>
<th>Valid in the CONTENTS statement in PROC CASUTIL</th>
<th>Valid in the LIST FILES statement in PROC CASUTIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASSWORD=</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>PATH=</td>
<td>Required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCHEMA=</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>SRCTYPE=</td>
<td>Required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USERNAME=</td>
<td>Required</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

* When values are supplied for both, LOAD statement values override data source values.

**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, oraclecaslib. All of the options supplied in this example are required in the CASLIB statement, except the PASSWORD= option.

In this example, the Oracle data is stored at the location designated by the PATH= argument. The data is read serially into the caslib oraclecaslib.

```sas
    caslib oraclecaslib desc='Oracle Caslib'
    datasource=(srctype='oracle',
                 dataTransferMode='serial',
                 username='user1',
                 password='******',
                 path='//machine.lnx.com:5570/exadat'
    );
```

**Example 2: Load Oracle Data into SAS Cloud Analytic Services Using PROC CASUTIL**

```sas
    proc casutil;
      list tables incaslib="oraclecaslib";       /* 1*/
      load casdata="MYORADATA" incaslib="oraclecaslib" outcaslib="casuser" casout="ORAdata_from_oraclecaslib";  /* 2*/
      list tables incaslib="casuser";             /* 3*/
      contents casdata="%upcase(class_from_oraclecaslib)" incaslib="casuser"; /* 4*/
    run;
```

1 List the tables in oraclecaslib prior to loading your data.

2 Load the table myORADATA from Oracle into caslib Casuser. Call the new table ORAdata_from_oraclecaslib.

   **Note:** You must list Oracle table names in all uppercase.
3  List the tables in oraclecaslib again to see the newly created table, ORAdata_from_oraclecaslib, that you loaded.

4  List information about the newly loaded table, including column names, data types, and so on.

---

**SAS Data Connector to PC Files**

Specifies the options to use when loading data from Microsoft Excel workbooks into SAS Cloud Analytic Services.

**Applies to:** CASUTIL procedure

**Example:** Load the Titanic data set into SAS Cloud Analytic Services. The file is from http://biostat.mc.vanderbilt.edu/wiki/Main/DataSets.

```sas
options validvarname=any;
proc casutil;
   load file="/path/to/titanic3.xls" casout="titanic3"
      importoptions=(filetype="xls" getnames=true);
quit;
```

---

**Syntax**

**File Type Arguments**

**FILETYPE="XLS" | "EXCEL"**

specifies the file type. Specify EXCEL when the file has a .xlsx filename suffix. National language characters are not displayed correctly for .xls files, use .xlsx instead.

**Requirement** The FILETYPE= argument 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

**RANGE="string"**

specifies the sheet and subset of the cells to import. For XLS files, the range "Sheet1$A1:B5" is the range address for a rectangular block of 8 cells, where the top left cell is A1 and the bottom right cell is B5. For XLSX files (FILETYPE=EXCEL), do not include the worksheet name in the range.

**SHEET="string"**

specifies the name of the worksheet to import. If you plan to import a range of cells from an XLS file, then specify the sheet as part of the RANGE= option and do not specify this option.

**Details**

Working with Microsoft Excel files does not require any data source arguments, but you must use a caslib with a data source type of PATH.
If you can navigate to the file in the **Server Files and Folders** section, then you can use the CASUTIL procedure with the LOAD FILE= syntax to load the data into SAS Cloud Analytic Services.

For large files, if the file can be accessed by SAS Cloud Analytic Services, then you can use the CASUTIL procedure with the LOAD CASDATA= syntax.

**Example**

The following 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-data-set](http://catalog.data.gov/dataset/2010-federal-stem-education-inventory-data-set). 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. The GETNAMES="false" option indicates to read the data in row 3 as data values rather than column names.

```sas
proc casutil;
   load file="2010 Federal STEM Education Inventory Data Set.xls"
       casout="stem2010"
       importoptions=(filetype="xls" getnames="false" range="Sheet1$A3:IV254");
/*
 * Ensure 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;
```

**SAS Data Connector to SAS Data Sets**

Specifies the settings to use when loading data from SAS data sets into SAS Cloud Analytic Services.

**Applies to:** CASUTIL procedure

**Example:** Load a password-protected SAS data set into SAS Cloud Analytic Services.

```sas
proc casutil;
   load casdata="salary.sas7bdat" casout="salary"
       importoptions=(filetype="basesas" password="secret");
run;
```

**Syntax**

**File Type Arguments**

CHARMULTIPLIER=integer

specifies to increase the width of fixed-byte-width character columns. The number of bytes needed for multi-byte characters depends on the particular characters in the
string. Specifying 1.5 is common, but sometimes it is an overestimate, and sometimes it truncates.

Default 1
Range 1–5

**DATATRANSFERMODE=**"AUTO" | "PARALLEL" | "SERIAL"

specifies how the SAS data set is read by SAS Cloud Analytic Services when the server is a distributed server.

**AUTO**
specifies that the worker nodes try to access and read data from the SAS data set. The data set must be on a shared file system that all the machines can access. If the worker nodes cannot access the data set, then the operation falls back to a serial data read by the SAS Cloud Analytic Services controller node.

**PARALLEL**
specifies that the worker nodes try to access and read data from the SAS data set. If the worker nodes cannot access the data set, then the load request fails.

**SERIAL**
specifies that the SAS Cloud Analytic Services controller node accesses and reads the SAS data set.

Default AUTO

**ENCRYPTIONPASSWORD=**"string"

specifies a password for decrypting stored data.

Restriction This parameter is valid only if the data set was encrypted with AES encryption. If it was encrypted with SAS proprietary encryption, you must instead use the password option.

**FILETYPE=**"BASE SAS"

specifies the file type.

Note The FILETYPE= argument is not required if you specify an extension if the SAS7BDAT extension is provided as part of the table name.

**PASSWORD=**"string"

specifies the password for a password-protected data set. Use this parameter if the data set is password-protected or uses SAS proprietary encryption.

**READ=**"string"

specifies the Read password for the SAS data set.

Details

Working with SAS data sets (.sas7bdat files) does not require any data source arguments. They can only be read from a caslib with a data source type of PATH.

For large data sets, if the .sas7bdat file can be accessed by SAS Cloud Analytic Services, then you can use the CASUTIL procedure with the LOAD CASDATA= syntax. In this case, the file type arguments in the preceding section can be used.

If you can access the SAS data set with the LIBNAME statement, then you can use the CASUTIL procedure with the LOAD DATA= syntax to load the data into SAS Cloud Analytic Services. For an example, look for sashelp.iris in load a client-side file.
Chapter 10
Data Types

SAS Cloud Analytic Services Data Types

A data type is an attribute of every column in a table that specifies the type of data that
the column stores. For example, the data type is the characteristic of a piece of data that
indicates whether it is a character string, an integer, a floating-point number, a date, or a
time. The data type also determines how much memory to allocate for the column value.

SAS Cloud Analytic Services currently supports these data types, which support missing
values.

Table 10.1  Data Types for SAS Cloud Analytic Services Table Columns

<table>
<thead>
<tr>
<th>Data Type Definition Keyword</th>
<th>SAS Cloud Analytic Services Table Column Data Type</th>
<th>Description</th>
<th>Data Type Returned</th>
<th>Missing Values</th>
</tr>
</thead>
</table>
| CHAR($n$)                   | CHAR($n$)                                        | Stores a fixed-length character string, where $n$
is the maximum number of characters to store. The maximum number of characters is
required to store each value regardless of the actual size of the value. If CHAR(10) is
specified and the character string is only five characters long, the value is right-padded with
spaces.

*Note:* This data type cannot contain ANSI SQL null values. | CHAR($n$) | all blanks (the same as in SAS) |
<table>
<thead>
<tr>
<th>Data Type Definition Keyword</th>
<th>SAS Cloud Analytic Services Table Column Data Type</th>
<th>Description</th>
<th>Data Type Returned</th>
<th>Missing Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR(n)</td>
<td>VARCHAR(n)</td>
<td>Stores a varying-length character string, where ( n ) is the actual number of characters to store. If VARCHAR(10) is specified and the character string is only 5 characters long, the value is 5. It is not padded with spaces.</td>
<td>VARCHAR(n)</td>
<td>all blanks or a zero length</td>
</tr>
</tbody>
</table>
| Benefits of Using VARCHAR   |                                                   | - when the lengths of the character data vary significantly  
- when the longest strings are infrequent and would require a fixed length of 64 bytes | \n | \n |
| VARCHAR Variables with an Undefined Maximum Length |       | - VARCHAR(*) indicates that no maximum length on the column is being defined.  
- Using VARCHAR(*) can be helpful if the maximum length of data for a column is not known when the column is being defined. | \n | \n |
| Consideration               |                                                   | Here is a consideration to keep in mind when you use VARCHAR(*). 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 and is defined with the maximum length of 32767 bytes. If you instead provide an explicit length, such as VARCHAR(10), a 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. | \n | \n |
| Note                        |                                                   | This data type cannot contain ANSI SQL null values. | \n | \n |
| DOUBLE                      | 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. | DOUBLE | the same as in SAS, TKMACMISSING, or values that TK_MISSV returns |
Chapter 11
Functions

Dictionary

GETCASURL Function
Returns the value for a URL for connecting to the CAS Server Monitor.

Requirement:
• The server name identified by the SAS CASHOST= option is used when constructing the URL.
• The value provided is a valid session name for the SAS CASHOST= option or, if the value is not provided, the SAS SESSREF= option is a valid session on the server. This connection obtains additional URL information.

Syntax
GETCASURL(<session>)

Optional Argument
session
if 0 parameters are specified, then the SAS SESSREF= option value is used.

Example: GETCASURL Example
$put httpaddr= sysfunc(getcasurl());
httpaddr=http://host and port value

GETSESSOPT Function
Returns the value for a SAS Cloud Analytic Services session option.
Syntax

GETSESSOPT (session-name session-option-name)

Required Arguments

session-name
CAS session name.

session-option-name
CAS session option name.

For more information, see Chapter 13, “Session Options,” on page 137.

**Tip** You can list the session option names with this code:

```plaintext
CAS mysess LISTSESSOPTS;
```

Example: Listing the Active CASLIB

This example returns the CASLIB option value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>%put caslib = %sysfunc(GETSESSOPT(mysess, caslib));</td>
<td>x=CASUSERHDFS(userid)</td>
</tr>
<tr>
<td>data work.one ;</td>
<td></td>
</tr>
<tr>
<td>x = GETSESSOPT(&quot;mysess&quot;, &quot;caslib&quot;) ;</td>
<td></td>
</tr>
<tr>
<td>put x= ;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

SESSFEND Function

Returns a 0 when a CAS session is not connected to a server and a 1 when the session is connected to a server.

Syntax

SESSFEND (session-name)

Required Argument

session-name
returns a value to indicate the status of the session.

0
The session is not found.

1
The session is found.
Details

This function has access to sessions that you started in your SAS session only. You can use the CAS statement with the LISTSESSIONS option to identify all your CAS sessions on a server.

Example: Determining CAS Sessions

This example shows whether the CAS session is found.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>%put doIExist= %sysfunc(sessfound(mysess));</td>
<td>doIExist= 0</td>
</tr>
<tr>
<td>%put doIExist= %sysfunc(sessfound(existingSession));</td>
<td>doIExist= 1</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 12
Macro Variables

Dictionary

_CASHOST_ Macro Variable
Specifies the name of the SAS Cloud Analytic Services server.

- **Default:** Not defined
- **Range:** 256 characters
- **Interactions:** When set, this macro variable overrides SAS system option CASHOST.
  The CAS statement HOST= option overrides this macro variable.
- **Tip:** If you want to delete macro variable _CASHOST_ and allow SAS system option CASHOST to prevail, use the following statement:
  ```
  %symdel _CASHOST_;
  ```

_Syntx

%let _CASHOST_=cloud.example.com

_CASPORT_ Macro Variable
Specifies the SAS Cloud Analytic Services server port.

- **Default:** 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.
- **Note:** When set to 0, CAS selects a port number.
Tip: If you want to delete macro variable _CASPORT_ and allow SAS system option CASPORT to prevail, use the following statement:

```sas
%symdel _CASPORT_
```

Syntax

```sas
%let _CASPORT_=5570
```

_SESSREF_ Macro Variable

Stores the name of the active SAS Cloud Analytic Services session.

- **Default:** Not defined
- **Interactions:** When you use the CAS statement to create a new session, this macro variable is automatically set to the name of the new session, which is CASAUTO by default. When you set SAS system option CASNAME (alias SESSREF), this macro variable is automatically set to the same value.
Chapter 13
Session Options

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 8. To see the setting for a specific property, use the GETSESSOPT function. See “GETSESSOPT Function” on page 131.
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.
- the TIMEOUT= option in the CAS statement for the session TIMEOUT option.
- 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 server configuration file
- command-line options that are used in the server start-up command
- SAS Cloud Analytic Services system defaults

You can browse the configured and default option values on the Configuration page of the SAS Cloud Analytic Services Server Monitor. For each option, this page shows the current value and the source of the value. The values for the session options are used as defaults for the session options, unless they are overridden. For information about the Server Monitor, see “About CAS Server Monitor” in SAS Viya Administration: Getting Started.

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.

<table>
<thead>
<tr>
<th>Task</th>
<th>Option to use:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Override one or more session options for a new session only.</td>
<td>CAS statement SESSOPTS= option</td>
</tr>
<tr>
<td>Override the TIMEOUT option value for a new session only.</td>
<td>CAS statement TIMEOUT= option*</td>
</tr>
<tr>
<td>Override the TIMEOUT option value for all subsequently created sessions.</td>
<td>SAS system option CASTIMEOUT=***</td>
</tr>
<tr>
<td>Override the NWORKERS option value for all subsequently created sessions.</td>
<td>SAS system option CASNWORKERS=****</td>
</tr>
</tbody>
</table>

* TIMEOUT in the CAS statement SESSOPTS= option overrides this option.
** NWORKERS in the CAS statement SESSOPTS= option overrides this option.
*** Setting this option does not affect existing sessions.
**Setting Options for Existing Sessions**

For an existing session, use the option shown in the following table to complete the desired task.

<table>
<thead>
<tr>
<th>Task</th>
<th>Option to use:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change one or more session options for a specific session.</td>
<td>CAS statement <strong>SESSOPTS</strong> option</td>
</tr>
<tr>
<td>Change one or more session options for the active session.*</td>
<td>SAS system option <strong>CASSESSOPTS=</strong> (alias <strong>SESSOPTS=</strong>)</td>
</tr>
<tr>
<td>Specify a caslib for the active session.*</td>
<td>SAS system option <strong>CASLIB=</strong></td>
</tr>
</tbody>
</table>

* SAS system option **CASNAME** (alias **SESSREF**) stores the name of the currently active session.

---

**Session Options by Category**

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td><strong>APPTAG=</strong> Session Option (p. 140)</td>
<td>specifies the string to prefix to log messages.</td>
</tr>
<tr>
<td>Caslib</td>
<td><strong>CASLIB=</strong> Session Option (p. 141)</td>
<td>specifies the caslib name to set as the active caslib.</td>
</tr>
<tr>
<td></td>
<td><strong>MAXTABLEMEM=</strong> Session Option (p. 145)</td>
<td>specifies the maximum amount of physical memory, in bytes, to allocate for a table.</td>
</tr>
<tr>
<td>DATA Step</td>
<td><strong>AUTOCORRECT=</strong> Session Option (p. 141)</td>
<td>automatically corrects misspelled procedure names and keywords, and global statement names.</td>
</tr>
<tr>
<td></td>
<td><strong>DATASTEPREPLACEABLE</strong> = Session Option (p. 142)</td>
<td>specifies whether a DATA step can replace an existing table.</td>
</tr>
<tr>
<td></td>
<td><strong>DATASTMTCHK=</strong> Session Option (p. 142)</td>
<td>specifies which SAS statement keywords are prohibited from being specified as a one-level DATA step name to protect against overwriting an input data set.</td>
</tr>
<tr>
<td></td>
<td><strong>DKRICOND=</strong> Session Option (p. 143)</td>
<td>specifies the error level to report when a variable is missing from an input data set during the processing of a <strong>DROP=</strong>, <strong>KEEP=</strong>, or <strong>RENAME=</strong> data set option.</td>
</tr>
<tr>
<td></td>
<td><strong>DKROCOND=</strong> Session Option (p. 143)</td>
<td>specifies the error level to report when a variable is missing from an output data set during the processing of a <strong>DROP=</strong>, <strong>KEEP=</strong>, or <strong>RENAME=</strong> data set option.</td>
</tr>
<tr>
<td></td>
<td><strong>ERRORS=</strong> Session Option (p. 143)</td>
<td>specifies the maximum number of observations for which SAS issues complete error messages.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>INVALIDDATA= Session Option (p. 144)</td>
<td>specifies the value that SAS assigns to a variable when invalid numeric data is encountered.</td>
</tr>
<tr>
<td></td>
<td>VARINITCHK= Session Option (p. 148)</td>
<td>specifies whether a DATA step stops or continues executing when a variable is not initialized and the type of message to write to the SAS log in that case.</td>
</tr>
<tr>
<td>Formats</td>
<td>FMTCASLIB Session Option (p. 144)</td>
<td>specifies the caslib where persisted format libraries are retained.</td>
</tr>
<tr>
<td>Localization</td>
<td>LOCALE= Session Option (p. 144)</td>
<td>specifies the locale to use for sorting and formatting.</td>
</tr>
<tr>
<td>Log</td>
<td>LOGFLUSHTIME= Session Option (p. 144)</td>
<td>specifies the log flush time, in milliseconds.</td>
</tr>
<tr>
<td></td>
<td>MESSAGELEVEL= Session Option (p. 145)</td>
<td>specifies the log message level.</td>
</tr>
<tr>
<td></td>
<td>METRICS= Session Option (p. 146)</td>
<td>specifies whether to include detailed performance metrics reports in the SAS log.</td>
</tr>
<tr>
<td>Session</td>
<td>NWORKERS= Session Option (p. 147)</td>
<td>specifies the number of worker nodes for a new session.</td>
</tr>
<tr>
<td></td>
<td>TIMEOUT= Session Option (p. 148)</td>
<td>specifies the SAS Cloud Analytic Services session time-out in seconds for a new or existing session.</td>
</tr>
<tr>
<td></td>
<td>TIMEZONE= Session Option (p. 148)</td>
<td>specifies the time zone offset, in hours, from UTC.</td>
</tr>
<tr>
<td>Sort</td>
<td>COLLATE= Session Option (p. 141)</td>
<td>specifies the collating sequence for sorting.</td>
</tr>
</tbody>
</table>

**Dictionary**

**APPTAG= Session Option**

specifies the string to prefix to log messages.

- **Valid in:** CAS statement SESSOPTS option
  OPTIONS statement CASSESSOPTS option
  GETSESSOPT function

- **Category:** Action

- **Default:** No prefix
**Syntax**

`APPTAG="tag-string"`

---

**AUTOCORRECT= Session Option**

automatically corrects misspelled procedure names and keywords, and global statement names.

**Valid in:**
- CAS statement `SESSOPTS` option
- `OPTIONS` statement `CASSESSOPTS` option
- `GETSESSOPT` function

**Category:** DATA Step

**Default:** TRUE

---

**Syntax**

`AUTOCORRECT=TRUE | FALSE`

---

**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

**Category:** Caslib

**See:** “CASLIB Statement” on page 35

---

**Syntax**

`CASLIB="caslib-name"`

---

**COLLATE= Session Option**

specifies the collating sequence for sorting.

**Valid in:**
- CAS statement `SESSOPTS` option
- `OPTIONS` statement `CASSESSOPTS` option
- `GETSESSOPT` function

**Category:** Sort

**Default:** UCA

---

**Syntax**

`COLLATE="MVA" | "UCA"`
Parameter Values

MVA
   specifies SAS client collating.

UCA
   specifies a locale-appropriate collating sequence.

DATASTEPREPLACE= Session Option
specifies whether a DATA step can replace an existing table.

Valid in: CAS statement SESSOPTS option
          OPTIONS statement CASSESSOPTS option
          GETSESSOPT function

Category: DATA Step
Default: TRUE

Syntax

DATASTEPREPLACE=TRUE | FALSE

DATASTMTCHK= Session Option
specifies which SAS statement keywords are prohibited from being specified as a one-level DATA step name to protect against overwriting an input data set.

Valid in: CAS statement SESSOPTS option
          OPTIONS statement CASSESSOPTS option
          GETSESSOPT function

Category: DATA Step
Default: COREKEYWORDS

Syntax

DATASTMTCHK= "ALLKEYWORDS" | "COREKEYWORDS" | "NONE"

Parameter Values

ALLKEYWORDS
   prohibits any keyword that can begin a statement in the DATA step (for example, ABORT, ARRAY, INFILE) as a one-level data set name in the DATA statement.

COREKEYWORDS
   prohibits using MERGE, RETAIN, SET, and UPDATE as one-level SAS data set names in the DATA statement.

NONE
   provides no protection against overwriting SAS data sets.
DKRICOND= Session Option
specifies the error level to report when a variable is missing from an input data set during the processing of a DROP=, KEEP=, or RENAME= data set option.

Valid in: CAS statement SESSOPTS option
OPTIONS statement CASSESSOPTS option
GETSESSOPT function

Category: DATA Step

Default: ERROR

Syntax
DKRICOND="ERROR" | "NOWARN" | "NOWARNING" | "WARN" | "WARNING"

DKROCOND= Session Option
specifies the error level to report when a variable is missing from an output data set during the processing of a DROP=, KEEP=, or RENAME= data set option.

Valid in: CAS statement SESSOPTS option
OPTIONS statement CASSESSOPTS option
GETSESSOPT function

Category: DATA Step

Default: WARNING

Syntax
DKROCOND="ERROR" | "NOWARN" | "NOWARNING" | "WARN" | "WARNING"

ERRORS= Session Option
specifies the maximum number of observations for which SAS issues complete error messages.

Valid in: CAS statement SESSOPTS option
OPTIONS statement CASSESSOPTS option
GETSESSOPT function

Category: DATA Step

Default: 20

Range: 0–2147483647

Syntax
ERRORS=number
**FMTCASLIB Session Option**

specifies the caslib where persisted format libraries are retained.

**Valid in:** CAS statement SESSOPTS option
OPTIONS statement CASSESSOPTS option
GETSESSOPT function

**Category:** Formats

**Note:** This option is set by the system administrator.

**Default:** FORMATS

**INVALIDDATA= Session Option**

specifies the value that SAS assigns to a variable when invalid numeric data is encountered.

**Valid in:** CAS statement SESSOPTS option
OPTIONS statement CASSESSOPTS option
GETSESSOPT function

**Category:** DATA Step

**Default:** . (dot)

**Syntax**

```
INVALIDDATA="invalid-number-string"
```

**LOCALE= Session Option**

specifies the locale to use for sorting and formatting.

**Valid in:** CAS statement SESSOPTS option
OPTIONS statement CASSESSOPTS option
GETSESSOPT function

**Category:** Localization

**Default:** en_US

**Syntax**

```
LOCALE="locale"
```

**LOGFLUSHTIME= Session Option**

specifies the log flush time, in milliseconds.
Valid in: CAS statement SESSOPTS option
OPTIONS statement CASSESSOPTS option
GETSESSOPT function

Category: Log
Default: 100
Range: −1−86400

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.

MAXTABLEMEM= Session Option
specifies the maximum amount of physical memory, in bytes, to allocate for a table.

Valid in: CAS statement SESSOPTS option
OPTIONS statement CASSESSOPTS option
GETSESSOPT function

Category: Caslib
Default: 16M

Note: After this threshold is reached, the server uses temporary files and operating system facilities for memory management.

Tip: 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.

Syntax
MAXTABLEMEM=number

MESSAGELEVEL= Session Option
specifies the log message level.

Valid in: CAS statement SESSOPTS option
OPTIONS statement CASSESSOPTS option
GETSESSOPT function
**Category:** Log  
**Default:** ALL

---

**Syntax**

```
MESSAGELEVEL="ALL" | "DEFAULT" | "ERROR" | "NONE" | "NOTE" | "WARNING"
```

---

**METRICS= Session Option**

Specifies whether to include detailed performance metrics reports in the SAS log.

- **Valid in:** CAS statement SESSOPTS option  
  OPTIONS statement CASSESSOPTS option  
  GETSESSOPT function

- **Category:** Log  
- **Default:** FALSE

- **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 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 metric notes that might be written to the log.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Note Written to the SAS Log</th>
</tr>
</thead>
<tbody>
<tr>
<td>The number of bytes moved.</td>
<td>NOTE: bytes moved number&lt;units&gt;</td>
</tr>
<tr>
<td>The CPU time in seconds and as a</td>
<td>NOTE: cpu time number seconds (number%)</td>
</tr>
<tr>
<td>percentage of cluster utilization. Cluster</td>
<td></td>
</tr>
<tr>
<td>utilization is the sum of the utilization</td>
<td></td>
</tr>
<tr>
<td>for each core in the cluster and can</td>
<td></td>
</tr>
<tr>
<td>exceed 100%. For example, cluster</td>
<td></td>
</tr>
<tr>
<td>utilization for a 96-core cluster where</td>
<td></td>
</tr>
<tr>
<td>each core is 100% used is 9600%.</td>
<td></td>
</tr>
<tr>
<td>The data movement time in seconds.</td>
<td>NOTE: data movement time number seconds</td>
</tr>
</tbody>
</table>
Here is an example of the metrics that are written to the SAS log when the MDSUMMARY procedure is executed.

```plaintext
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
```

**NWORKERS= Session Option**

specifies the number of worker nodes for a new session.

- **Valid in:** CAS statement SESSOPTS option
  OPTIONS statement CASSESSOPTS option
  GETSESSOPT function

- **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.

- **See:** “CASNWORKERS= System Option” on page 155

**Syntax**

```
NWORKERS=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
- OPTIONS statement CASSESSOPTS option
- GETSESSOPT function

Category: Session

Default: In order of descending precedence:
1. CAS statement TIMEOUT= option value, if specified
2. SAS system option CASTIMEOUT=, if you explicitly set it in SAS to a value greater than 0
3. 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” on page 157

Syntax

TIMEOUT=number

TIMEZONE= Session Option

specifies the time zone offset, in hours, from UTC.

Valid in:
- CAS statement SESSOPTS option
- OPTIONS statement CASSESSOPTS option
- GETSESSOPT function

Category: Session

Default: –1

Range: –1–23

Syntax

TIMEZONE=number

VARINITCHK= Session Option

specifies whether a DATA step stops or continues executing when a variable is not initialized and the type of message to write to the SAS log in that case.
Valid in: CAS statement SESSOPTS option
OPTIONS statement CASSESSOPTS option
GETSESSOPT function

Category: DATA Step

Default: NOTE

Syntax

VARINITCHK="ERROR" | "NONOTE" | "NOTE" | "WARN" | "WARNING"

Parameter Values

ERROR
 specifies that the DATA step stops executing and writes an error message to the SAS log when a variable is not initialized.

NONOTE
 specifies that the DATA step continues to execute without writing a message to the SAS log when a variable is not initialized.

NOTE
 specifies that the DATA step continues to execute and writes a note to the SAS log when a variable is not initialized.

WARN | WARNING
 specifies that the DATA step continues to execute and writes a warning message to the SAS log when a variable is not initialized.
AUTHINFO= System Option

specifies a file where user ID and passwords are kept for authentication.

Valid in: SAS Studio: OPTIONS statement, Batch Submit
          Batch and line modes: SAS command, configuration file, SASV9_OPTIONS environment variable, OPTIONS statement

PROC OPTIONS GROUP= CAS
  Alias: CASAUTHINFO=

Interaction: SAS Studio user credentials are used to authenticate the connection to CAS. The Authinfo file credentials are not used. The most frequent use of this system option is to submit code to CAS from the command line, in batch mode.

Note: AUTHINFO is also an environment variable.

See: For more information about the Authinfo file, see “Create an Authinfo File” in SAS Viya Administration: Authentication.

Syntax

AUTHINFO='authinfo_file_path';
Session Options

authinfo_file_path

specifies the path where an Authinfo file is located. The Authinfo file provides an alternative to including passwords in programs. The most frequent use of this option is to submit code to CAS from the command line, in batch mode. For more information, see “Authinfo Authentication Method” in SAS Viya Administration: Authentication and “Create an Authinfo File” in SAS Viya Administration: Authentication.

AUTHINFO can also be used as an environment variable. The environment variable can hold the name of one or more files. This variable is formatted like a PATH environment variable where a colon separates the filenames.

End users can store an encoded password in an Authinfo file. For more information, see “Encode the Authinfo Password Using PROC PWENCODE” in SAS Viya Administration.

Examples

Set AUTHINFO= SAS system option. This option overrides the Authinfo file pointed to by the AUTHINFO environment variable.

Options AUTHINFO='$HOME/authInfo-file';

AUTHINFO can also be set as an environment variable. This option overrides the .authinfo file.

Options insert=(set=AUTHINFO='$HOME/authInfo-file');

Details

There is an order of precedence to using AUTHINFO options. The following order applies.

1. The AUTHINFO environment variable overrides the .authinfo file.
2. The AUTHINFO= SAS system option overrides the AUTHINFO environment variable.
3. The CAS statement AUTHINFO option overrides the AUTHINFO SAS System option and the environment variable.

See Also

• “Authinfo Authentication Method” in SAS Viya Administration: Authentication
• “Create an Authinfo File” in SAS Viya Administration: Authentication
• “CAS Statement” on page 1

CASSESSOPTS= System Option

Specifies one or more session options for the active CAS session.

Valid in:

- SAS Studio: OPTIONS statement, Batch Submit
- Batch and line modes: SAS command, configuration file, SASV9_OPTIONS environment variable, OPTIONS statement

PROC OPTIONS

GROUP= CAS

Alias: SESSOPTS=
Default: None

Note: This option can be restricted by a site administrator. For more information, see “Restricted Options” in SAS Viya System Options: Reference.

Tip: A best practice is to explicitly set options for a session using the CAS statement

```cas mysess sessopts=(caslib=mycaslib collate=UCA);
```

See “SESSOPTS=(session-option(s))” on page 9.

Example: For the default session, set the caslib to MYCASLIB and the session connection
time-out to 60 minutes:

```options sessopts=(caslib="mycaslib" timeout=3600)
```

---

**Syntax**

```CASSESSOPTS=(session-option(s))
SESSOPTS=(session-option(s))
```

**Syntax Description**

`session-option(s)` specifies one or more session options as `option=value` pairs separated by a space and
enclosed in parentheses.

**Tip** To reflect a session option value, use this statement:

```%put caslib=%sysfunc(GETSESSOPT(session, option));
```

See Chapter 13, “Session Options,” on page 137 for a list of the options that you
can specify for `session-option(s)`.

---

**See Also**

Functions:

- “GETSESSOPT Function” on page 131

Macro Statement and Functions:

- “%PUT Statement” in SAS Viya Macro Language: Reference
- “%SYSFUNC and %QSYSFUNC Functions” in SAS Viya Macro Language: Reference

Statements:

- “CAS Statement” on page 1

---

**CASHOST= System Option**

Specifies the CAS host name that is associated with a CAS session.

**Valid in:** SAS Studio: OPTIONS statement, Batch Submit
Batch and line modes: SAS command, configuration file, SASV9_OPTIONS environment variable, OPTIONS statement

**PROC OPTIONS GROUP=**

**CAS**

Default: None

Range: 256 characters

**Note:** This option can be restricted by a site administrator. For more information, see "Restricted Options" in SAS Viya System Options: Reference.

**Example:** options cashost="cloud.example.com";

---

**Syntax**

**CASHOST= ”host-name”**

---

**CASLIB= System Option**

Specifies the caslib name for the session that is identified by the SESSREF= option.

**Valid in:** SAS Studio: OPTIONS statement, Batch Submit

Batch and line modes: SAS command, configuration file, SASV9_OPTIONS environment variable, OPTIONS statement

**PROC OPTIONS GROUP=**

**CAS**

Default: None

Range: 128 characters

**Interaction:** The CAS statement session option CASLIB= overrides this option. For more information, see “CASLIB= Session Option” on page 141.

**Note:** This option can be restricted by a site administrator. For more information, see “Restricted Options” in SAS Viya System Options: Reference.

**Tip:** A best practice is to explicitly set options for a session using the CAS statement SESSOPTS= option. Here is an example:

cas mysess sessopts=(caslib=mycaslib collate=UCA);

See “SESSOPTS=(session-option(s))” on page 9.

**Example:** Set the default caslib:

    options caslib="casuser";

---

**Syntax**

**CASLIB= ”name”**

---

**See Also**

**Statements:**

• “CAS Statement” on page 1
• “CASLIB Statement” on page 35

---

**CASNAME= System Option**

Specifies the name to associate with a generated CAS session.

**Valid in:**
- SAS Studio: OPTIONS statement, Batch Submit
- Batch and line modes: SAS command, configuration file, SASV9_OPTIONS environment variable, OPTIONS statement

**PROC OPTIONS**

GROUP= CAS

**Alias:** SESSREF

**Default:** CASAUTO

**Range:** 256 characters

**Interactions:**
- When you create a session using the CAS statement, the value of the CASNAME= option and the _SESSREF_ macro variable are set to the session name. For more information, see “CAS Statement” on page 1.
- When you name a session using the CASNAME= option, the value of the _SESSREF_ macro variable is set to the same name.

**Note:** This option can be restricted by a site administrator. For more information, see “Restricted Options” in SAS Viya System Options: Reference.

**Example:**
```
options casname=mysessref;
```

**Syntax**

```
CASNAME= session-name
SESSREF= session-name
```

**See Also**

**Statements:**
- “CAS Statement” on page 1

---

**CASWORKERS= System Option**

Specifies the number of worker nodes to use for a CAS session.

**Valid in:**
- SAS Studio: OPTIONS statement, Batch Submit
- Batch and line modes: SAS command, configuration file, SASV9_OPTIONS environment variable, OPTIONS statement

**PROC OPTIONS**

GROUP= CAS

**Default:** ALL

**Restriction:** The number of worker nodes can be set for new sessions only during session creation.
Note: This option can be restricted by a site administrator. For more information, see “Restricted Options” in SAS Viya System Options: Reference.

Tip: The CAS statement session option NWORKERS= overrides this option. For more information, see “CAS Statement” on page 1 and “NWORKERS= Session Option” on page 147.

Example: options casnworkers=10;

### Syntax

**CASNWORKERS= ALL | number**

### Syntax Description

**ALL**
- specifies to use all of the worker nodes.

**number**
- specifies the number of worker nodes to use.
- In SMP mode, *number* is always 0, whether you set CASNWORKERS= to 0 or 1.
- In MPP mode, specify *number*=0 to use the maximum number of worker nodes that are available. You can set CASNWORKERS= to a number that is less than or equal to the maximum number of worker nodes.

Range 0-5000

---

### CASPORT= System Option

Specifies the port to use when connecting to CAS.

**Valid in:**
- SAS Studio: OPTIONS statement, Batch Submit
- Batch and line modes: SAS command, configuration file, SASV9_OPTIONS environment variable, OPTIONS statement

**PROC OPTIONS GROUP= CAS**
- **Default:** 0
- **Range:** 0-65535
- **Note:** This option can be restricted by a site administrator. For more information, see “Restricted Options” in SAS Viya System Options: Reference.

**Example:** options casport=12345;

### Syntax

**CASPORT=port-number**

### Syntax Description

**port-number**
- specifies the CAS server port number.
CASTIMEOUT= System Option

Specifies the CAS session time-out in seconds for new sessions. The session time-out starts when the number of connections to the session becomes zero and all session activity is complete.

Valid in: SAS Studio: OPTIONS statement, Batch Submit
Batch and line modes: SAS command, configuration file, SASV9_OPTIONS environment variable, OPTIONS statement

PROC OPTIONS
GROUP= CAS

Default: 60
Range: 0–31536000

Interaction: The CAS statement TIMEOUT= option and the session option TIMEOUT= override this option. For more information, see “TIMEOUT=seconds” on page 9 and “TIMEOUT= Session Option” on page 148.

Notes: 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. This option is ignored when the value is set to 0. In that case, the default for the TIMEOUT= CAS statement option or the TIMEOUT= session option applies. This option can be restricted by a site administrator. For more information, see “Restricted Options” in SAS Viya System Options: Reference.

Tip: To change the time-out for an existing session, use the TIMEOUT= session option.

Example: options castimeout=28800;

Syntax

CASTIMEOUT= n | nK | nM | hexX | MAX | MIN

Syntax Description

n | nK | nM
specifies the session time out in seconds that are processed in multiples of 1, 1,024 (K) or 1,048,576 (M). For example, a value of 432008 specifies 43,200 seconds, and a value of 43k specifies 44,032 seconds.

hexX
specifies the session time out seconds as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by an X. For example, the value 0a8c0x sets the number of seconds to 43200 seconds.

MAX
sets the time out value to 31536000.

MIN
sets the time out value to 0.
CASUSER= System Option

Specifies the user ID to use when connecting to CAS.

Valid in: SAS Studio: OPTIONS statement, Batch Submit
          Batch and line modes: SAS command, configuration file, SASV9_OPTIONS
          environment variable, OPTIONS statement

PROC OPTIONS
GROUP= CAS

Alias: CASUSERID=

Default: None

Requirement: The user ID that you specify must match a user ID in your personal .authinfo file. For
             more information about the .authinfo file, see “Create an Authinfo File” in SAS Viya
             Administration: Authentication.

Interactions: SAS Studio user credentials are used to authenticate your connection to CAS. After
              you are logged in to SAS Studio, you can submit code to CAS without using this
              option. The most frequent use of this option is to submit code to CAS from the
              command line, in batch mode.

              The CAS statement USER= option overrides the user ID specified by this option. If
              the user ID is not specified in the CAS statement, SAS looks for a user ID that is set
              by the CASUSER= option. For more information, see “USER= user-ID” on page 10.

Note: This option cannot be restricted by a site administrator. For more information, see
       “Restricted Options” in SAS Viya System Options: Reference.

Example: options casuser=myid;

Syntax

CASUSER= user-ID
CASUSERID= user-ID
Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore for additional books and resources.