



# An Introduction to SAS® Viya® 3.3 Programming

## What's Included in SAS Viya?

This software supports analytical data preparation, variable transformations, exploratory analysis, analytical modeling, integrated model comparison, and scoring. Here are the main components:

SAS Viya	The third generation of high-performance in-memory analytics.
SAS Studio	The integrated programming environment.
SAS Cloud Analytic Services (CAS)	The analytic engine. CAS uses high-performance, multithreaded analytic code to rapidly process requests against data of any size.
SAS Visual Analytics	<p>Programming tools that provide baseline functionality, including reporting and basic analytics. The following functionality is provided:</p> <ul style="list-style-type: none"><li>■ analytical data preparation</li><li>■ variable transformations</li><li>■ exploratory analysis</li><li>■ descriptive statistics</li></ul>
SAS Visual Statistics	<p>An additional set of advanced analytic functionality that builds on SAS Visual Analytics, providing the following capabilities:</p> <ul style="list-style-type: none"><li>■ ability to build predictive models</li><li>■ integrated model comparison</li></ul>
SAS Visual Data Mining and Machine Learning	<p>An additional set of advanced analytic functionality that builds on SAS Visual Statistics and that enables you to do the following:</p> <ul style="list-style-type: none"><li>■ tune machine learning algorithm hyperparameters</li><li>■ perform advanced statistical operations</li><li>■ analyze complex data</li></ul>
SAS Econometrics	A set of functionality that provides techniques to model complex business and economic scenarios and to analyze the dynamic impact that specific events might have over time.

SAS Visual Forecasting	Provides automatic variable, event, and model selection. It then automatically generates your forecasts.
SAS Visual Text Analytics	A text analytics framework that combines text mining, contextual extraction, categorization, sentiment analysis, and search capabilities.
SAS Optimization	A set of procedures for exploring models of distribution networks, production systems, resource allocation problems, and scheduling problems, using the tools of operations research.

## Servers and Sessions

### SAS Cloud Analytic Services

The high-performance processing power of SAS Viya is SAS Cloud Analytic Services (CAS). CAS is a server that provides the run-time environment for data management and analytics with SAS. (Run-time environment refers to the combination of hardware and software where data management and analytics take place.)

**Note:** Your site must license and install one or more of the SAS Viya products to use this functionality.

The CAS server has the following characteristics:

- Data being transmitted to and from the CAS server and stored data is secure. See these publications:

- [Encryption in SAS Viya: Data in Motion](#)

**Note:** If you are connecting the CAS server from SAS 9.4, see [Configure SAS 9.4 Clients to Work with SAS Viya](#) in this publication.

- [Encryption in SAS Viya: Data at Rest](#)

- Authentication is used to control access to the CAS server and its resources. Your identity must be successfully authenticated before your session is created.

SAS Studio and SAS Enterprise Guide authenticate the connection to CAS by using your user credentials. When password information is not available, an attempt is made to find an authinfo file (.authinfo is the default filename on Linux). The authinfo file provides a user name and password to CAS for PAM authentication. For more information, see [Authentication Mechanisms](#).

The SAS windowing environment requires an authinfo file for authentication. For information about creating an authinfo file, see [“Create an Authinfo File” in Client Authentication Using an Authinfo File](#) or contact your site administrator.

- The CAS server session encoding is UTF-8.
- The server can run on a single machine or as a distributed server on multiple machines. For both architectures, the server is multi-threaded for high-performance analytics.
  - The distributed server consists of one controller and one or more workers. This architecture is often referred to as a massively parallel processing (MPP) architecture.
 

Using single-server symmetric multi-processing (SMP), the threaded processing of the CAS server is shared by multiple CPUs or is shared between multiple cores of a single CPU.
  - The distributed server has a communication layer that supports fault tolerance. A distributed server can continue to process requests even after losing connectivity to some nodes. The communication layer also enables you to remove or add worker nodes from a server while it is running.

- ❑ If you have multiple CAS servers, one can be designated as a backup server. If the primary CAS server fails, CAS sessions are automatically connected to the backup server. New CAS sessions are then connected to the backup server.
- The CAS server processes in-memory tables. The source data for the in-memory tables can come from SAS data sets, server-side files, event stream processing, and database tables. Database tables can be loaded serially or in parallel.
- The server can manage all of your data and easily share data with multiple users.
- You can program using new high-performance SAS Viya procedures, most SAS 9.4 procedures and language elements, or CAS actions. CAS actions are the smallest unit of work for the CAS server. To program using CAS actions, you use the CAS procedure.
- The DATA step, DS2, and FedSQL languages can run in the CAS server.
- The CAS server organizes user-defined formats in format libraries. You can use the FORMAT procedure to make formats stored in catalogs available to the server in format libraries. You can also use PROC FMT2ITM to move SAS format catalogs to CAS format libraries. The CAS server does not support SAS catalogs.
- The programming interfaces include all programming interfaces that you use with SAS as well as the open-source languages Python, Lua, Java, and R.

### See Also

- For an overview of SAS Viya and architectural diagrams, see [SAS® Viya 3.3: Overview](#)
- [SAS Cloud Analytic Services: Fundamentals](#)
- [SAS Viya 3.3 Administration](#)

## The SAS Workspace Server

The SAS Workspace Server uses the V9 engine to process SAS data sets locally and to read external data.

The DATA step runs in the SAS Workspace Server as well as in the CAS server. If it is determined that the DATA step cannot run in the CAS server, the DATA step runs in the SAS Workspace Server.

The data management and utility procedures and macros run in the SAS Workspace Server. SAS procedures that do not run in the CAS server can process CAS tables in the SAS Workspace Server.

FedSQL and DS2 run on the SAS Workspace Server when they access SAS libraries. They run on the CAS server when they access caslibs.

The SAS Workspace Server supports SAS catalogs.

High-performance SAS Viya procedures and other supporting procedures use CAS actions to perform data analysis and processing in the CAS server, not in the workspace server.

## SAS Studio and Server Connection

Connection with the CAS server is automatically created when you use a version of SAS Studio in the 4.x series. The SAS Studio 4.x series is the programming environment for SAS Viya.

SAS Studio 3.71 is the web programming environments for SAS 9.4M5. After you sign on to SAS Studio, you must explicitly connect to the CAS server by using the CAS statement or by setting the CASHOST= and CASPORT= system options. These options can be set in a configuration file, an autoexec file, or in a SAS program.

## SAS Windowing Environment, SAS Enterprise Guide, and Server Connection

When you open the SAS windowing environment or SAS Enterprise Guide, you must connect with the CAS server by using the CAS statement or by using the CASHOST= and CASPORT= system options. These options can be set in a configuration file, an autoexec file, or in a SAS program. For an example of connecting to SAS Viya, see [Use the SAS Windowing Environment to Connect to the CAS Server and Run the FACTMAC Procedure on page 17](#).

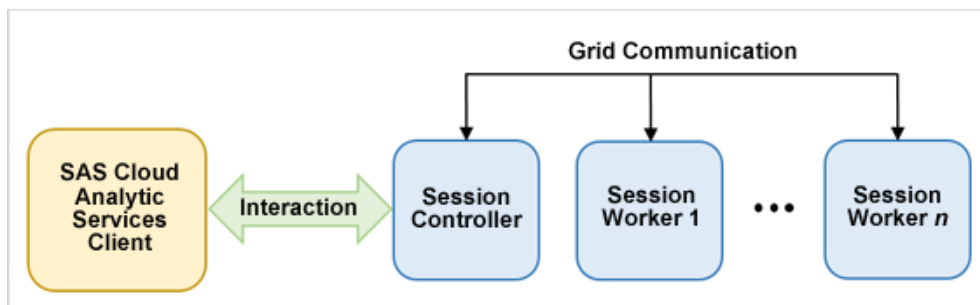
### CAS Sessions

Use the CAS statement to connect to a CAS server and start a session.

When you create a CAS session, the server completes the following process:

- 1 The server authenticates your identity.
- 2 The server listens for a session request.
- 3 The server starts a session between the client process and the session process, as shown in [Session Processes in a Distributed System](#).

**Figure 1** Session Processes in a Distributed System



You can view sessions by using the CAS Server Monitor that is available in the **More application options** menu in SAS Studio or by submitting the `CAS _ALL_ LIST;` statement.

Sessions provide the following functionality:

- user identification.
- fault isolation for each session. If a problem occurs in your session, it does not impact other sessions or the server. The client might be affected if the session that it connects to has a problem.
- efficiency. Resources that are visible only to your session (session-scoped tables and caslibs) do not require checks for concurrent access, and they run faster.
- resource tracking, which is enabled through the use of resource metrics.

Here is an example of using the SAS Studio code snippet **New CAS Session**. When you use the code snippet, you specify values that are particular to your environment. If you have not connected to the CAS server by using the CASHOST= and CASPORT= system options, you can add the connection information by using the CAS statement HOST= and PORT= options.

```

1 /*****
2 /*  Start a session named mySession using the existing CAS server connection */
3 /*  while allowing override of caslib, timeout (in seconds), and locale      */
4 /*  defaults.                                                                */
5 /*****
6
7 cas mySession sessopts=(caslib=casuser timeout=1800 locale="en_US");

```

You can add the CAS statement to the autoexec.sas file so that your CAS session starts when you start SAS or SAS Studio.

In the SAS windowing environment, you can use the AUTHINFO= option in the CAS statement to provide the connection information. For more information about the AUTHINFO= option, see [“AUTHINFO=authentication-info-file”](#) in *SAS Cloud Analytic Services: User’s Guide*.

## See Also

- [Editing the Autoexec File](#)
- [“Sessions” in SAS Cloud Analytic Services: Fundamentals](#)
- [“CAS Statement” in SAS Cloud Analytic Services: User’s Guide](#)
- [Working with Code Snippets](#)

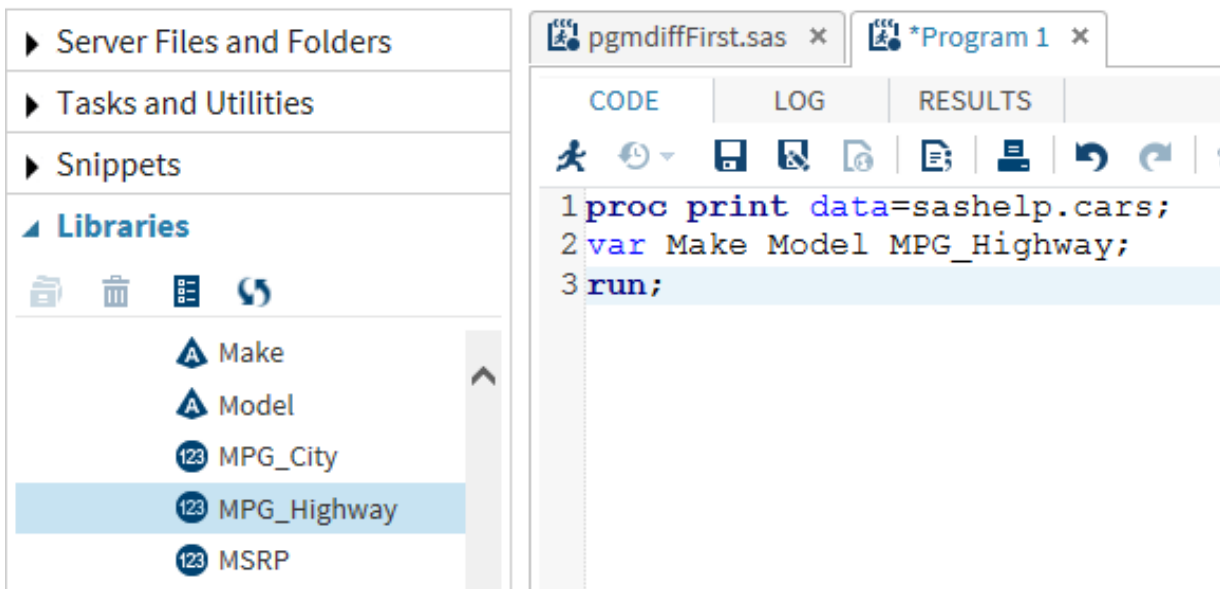
---

## Programming Interfaces

### SAS Studio

SAS Studio is a browser-based SAS coding interface. It is the only graphical editor that is installed with a SAS Viya deployment. SAS Studio is designed to help you write your SAS programs as quickly and accurately as possible. You can program using the point-and-click interface, or you can enter your code directly on the **CODE** tab.

- Click **Libraries** in the Navigation pane to access all of your libraries and the tables in those libraries.
- Save time when you are writing a program by dragging items from the **Libraries** section to your program. SAS Studio adds the code for those items to your program. In the following figure, the variables Make, Model, and MPG\_highway were dragged to the program on the **CODE** tab:

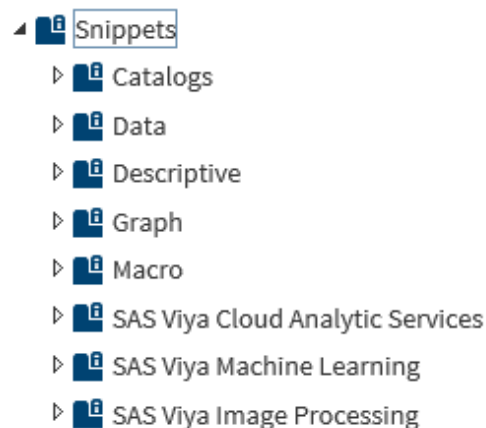


- Each time you submit code, the **LOG** tab displays messages that indicate whether the submission was successful. Table output from your program appears on the **RESULTS** tab. By default, SAS Studio creates HTML5 output. On the **RESULTS** tab, you can click one of the first three buttons to download the results in HTML5, PDF, or RTF formats:



To assist you in coding, SAS Studio provides snippets and tasks that automatically generate code when you drag them to the **CODE** tab. A snippet is code that you might use frequently. The following example explains how to use snippets in your programming.

- 1 In the Navigation pane, expand **Snippets** for a list of code snippets for use with SAS Viya Cloud Analytic Services, SAS Viya Machine Learning, and SAS Viya Image Processing.



- 2 Expand **SAS Viya Cloud Analytic Services**, drag the **New CAS Session** snippet to the **CODE** tab, and submit the code. A CAS session named mySession that uses the initially active caslib casuser is established.

```

1/*****
2/*  Start a session named mySession using the existing CAS server connection */
3/*  while allowing override of caslib, timeout (in seconds), and locale      */
4/*  defaults.                                                                */
5/*****
6
7cas mySession sessopts=(caslib=casuser timeout=1800 locale="en_US");

```

- 3 Drag the **Load data to caslib** snippet to the **CODE** tab. Delete the code above and below the PROC CASUTIL code that loads a SAS data set from a Base engine library.

```

26/*****
27/*  Load SAS data set from a Base engine library (library.tableName) into  */
28/*  the specified caslib ("myCaslib") and save as "targetTableName". The    */
29/*  PROMOTE option makes loaded data available to all active sessions.      */
30/*****
31
32proc casutil;
33    load data=library.tableName outcaslib="myCaslib"
34    casout="targetTableName" promote;
35run;

```





















- 4 Customize the code:

- Replace `library.tableName` with `sashelp.cars`.
- Replace `myCaslib` with `casuser`.
- Replace `targetTableName` with `cars`.

- 5 Submit the code. SAS runs the CASUTIL procedure code to perform the client-side load of the Cars data set from the Sashelp library to the Cars table in the Casuser caslib.

**TIP** This technique is appropriate for small data sets and convenient for a getting-started experience. More efficient techniques are available for larger data sets.

- 6 Now that you have data in a caslib, you can use the **Machine Learning** snippets or the **Tasks** to explore, model, and score data. Expand **Tasks** to see the types of tasks that assist you by generating code. Here are some of the tasks:

- ▾  Tasks
  -  Data
  -  Graph
  -  Combinatorics and Probability
  -  Statistics
  - ▾  SAS Viya Prepare and Explore
    -  Summary
    -  Transform Data
    -  Variable Selection
    -  Sampling
    -  Partitioning
    -  Binning
    -  Imputation
    -  SAS Viya Unsupervised Learning
    -  SAS Viya Supervised Learning
    -  SAS Viya Evaluate and Implement
    -  SAS Viya Text Analytics
    -  SAS Viya Network Analysis and Optimization
    -  SAS Viya Econometrics
    -  SAS Viya Forecasting

See the following videos for a demonstration of using the SAS Studio point-and-click interface:

- [Video: Code Snippets](#)
- [Video: Exploring Data](#)

## See Also

- [Getting Started with Programming in SAS Studio 3.7](#)
- [SAS Studio 3.7: User's Guide](#)

## Batch and Interactive Line-Mode Processing

You can submit code in batch mode and interactive line mode from the command line as you did in previous releases of SAS, with these differences:

- Only Zip, SFTP, and URL FILENAME access methods are available in SAS Viya.
- You set the CASHOST= and CASPORT= system options in the SAS command line, a configuration file, or in the autoexec file. For example:

```
-cashost "cloud.myorg.com" -casport 5570
```

- The default path to the SAS Viya install directory is `/opt/sas/spre`.

**Note:** You can submit batch programs from SAS Studio.



## See Also

- [“Batch Mode in UNIX Environments” in SAS Companion for UNIX Environments](#)
- [“Interactive Line Mode in UNIX Environments” in SAS Companion for UNIX Environments](#)
- [“Connecting to the CAS Server” in SAS Companion for UNIX Environments](#)

## SAS Windowing Environment and SAS Enterprise Guide

You can submit code in the SAS windowing environment and in SAS Enterprise Guide. Beginning with SAS 9.4M5, the CAS statement creates a session or CAS UUID to an existing session. The libref then becomes your handle to communicate from SAS with the specific session.

To run SAS 9.4 procedures and the DATA step that use CAS processing or transfer data from CAS in the SAS windowing environment or in SAS Enterprise Guide:

- 1 Specify your connection information. Before you run the code, you must provide a CAS port number and a CAS host name for the CASPORT= and CASHOST= system options. The values for these options that are provided below are just examples.

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

**Note:** If these options are specified in a configuration file or in an autoexec file, you do not need to specify them in your SAS program.

- 2 Start a session. Your system administrator might have set the default host and port for your server in a configuration file. You can start a session manually with the CAS statement. In the following example, the CAS statement creates the CAS session named casauto:

```
cas casauto;
```

For a complete example, see [Use the SAS Windowing Environment to Connect to the CAS Server and Run the FACTMAC Procedure on page 17](#).

## Open-Source Code for Python, Lua, R, and Java

SAS open-source code provides an interface to the CAS server using [CAS actions](#) from Python, Lua, R, and Java. A CAS action is the smallest unit of work in the server. Loading an in-memory table from a file is an example of an action. Related CAS actions are grouped in an action set. An example of an action is the table.loadtable that loads an in-memory table from a file.

The client machine that runs Python, Lua, R, or Java applications must use the following:

- 64-bit Linux or 64-bit Windows
- 64-bit version of Python, Lua, R, or Java

The following versions of the scripting languages are compatible with the SAS interfaces:

- ☐ Python 2.7.x or Python 3.4.x
- ☐ Lua 5.2 or Lua 5.3
- ☐ Oracle JDK SE Java 8
- ☐ OpenJDK version 1.8.0.65
- ☐ R 3.1.0 or later.
- ☐ Install the httr and jsonlite packages. These packages have additional dependencies that are automatically installed from CRAN when you run install.packages().

For Python, R, and Lua, you use the SAS Scripting Wrapper for Analytics Transfer (SWAT) interface to connect to the CAS server. For Java, you use the SAS Java Interface for Viya to connect to the CAS server. You can download the client interfaces from the following locations

Python	<a href="http://support.sas.com/downloads/package.htm?pid=1977">http://support.sas.com/downloads/package.htm?pid=1977</a>
Lua	<a href="http://support.sas.com/downloads/package.htm?pid=1975">http://support.sas.com/downloads/package.htm?pid=1975</a>
Java	<a href="http://support.sas.com/downloads/package.htm?pid=1976">http://support.sas.com/downloads/package.htm?pid=1976</a>
R	<a href="https://support.sas.com/downloads/package.htm?pid=2061">https://support.sas.com/downloads/package.htm?pid=2061</a>

This Python example connects to the CAS server, starts a session, and executes the listSessOpts action to list the name, type, and current value for all of the session options in session MyCasSess.

```
import swat

# Create session CASAUTO. Specify the CAS host and port for your site.
s = swat.CAS("cloud.example.com", 5570)
s.sessionName(name="mycassess")

# Get the current session options.
res = s.sessionProp.listSessOpts()

# Print the option name and value for each option.
print "Current session options:"
display(res.SessOpts.ix[:, ['Name', 'Type', 'Value']])
```

Many of the action examples in the Help Center show examples in the new CAS language (CASL), Lua, R, and Python. For example, the [Aggregation Action Set examples](#) provide CASL, Lua, R, and Python links below the navigation buttons. You can find additional examples in the [SAS Viya community](#).

The REST APIs are available at [developer.sas.com](https://developer.sas.com).

## Data Migration to UTF-8 Encoding

The CAS server supports only UTF-8 encoding. UTF-8 is variable-width, multi-byte encoding. One UTF-8 character can be 1–4 bytes in length. If your SAS or SAS Viya session encoding is not UTF-8, then the data that is read to a CAS table must be transcoded to UTF-8. Set your SAS or SAS Viya session to UTF-8 in order to avoid the transcoding of data. If you want to change your session encoding, see [SAS Viya 3.3 Administration: General Servers and Services](#).

When double-byte character set (DBCS) characters and some single-byte character set (SBCS) characters are transcoded to UTF-8, they require additional bytes to represent a character.

**CAUTION! Data truncation can occur during transcoding.** If a data set column or a database column is not wide enough to store the results that are encoded as UTF-8 when data transcoding occurs, then data might be truncated. To avoid truncation, specify the appropriate option for the CAS LIBNAME engine, the CASUTIL procedure LOAD statement, or CAS data connectors. If a variable width is associated with a format, the format width is not increased. You must re-create the format using the FORMAT procedure. If there is the possibility of truncated data, an error message appears in the log.

When data is written to the CAS server, the client estimates the number of bytes that are needed to transcode the data to UTF-8. For the CAS engine and the CASUTIL procedure LOAD statement, SBCS environments

estimate 1 byte in UTF-8 for every 1 byte in the local encoding. DBCS environments estimate 1.5 bytes in UTF-8 for every 1 byte in the local encoding. For CAS data connectors, all environments estimate 3 bytes in UTF-8 for every 1 byte in the local encoding. You can use the following options to replace the estimate with an explicit value of the byte multiplier when you know the number of bytes that are needed to represent the data in UTF-8:

- [NCHARMULTIPLIER= CAS engine LIBNAME option](#)
- [NCHARMULTIPLIER= data set option](#)
- [NCHARMULTIPLIER= LOAD statement option](#) in the CASUTIL procedure
- [CASNCHARMULTIPLIER= system option](#)
- [CHARMULTIPLIER= option](#) for these [CAS data connectors](#):
  - ☐ Amazon Red Shift
  - ☐ DB2
  - ☐ Hana
  - ☐ Impala
  - ☐ Microsoft SQL Server
  - ☐ ODBC
  - ☐ Oracle
  - ☐ PostgreSQL
  - ☐ SAS data sets
  - ☐ Teradata

**Note:** The NCHARMULTIPLIER= option overrides the value of the CASNCHARMULTIPLIER= system option. CAS data connectors do not recognize the value of the CASNCHARMULTIPLIER= system option.

A best practice is to perform a test reading of DBCS data by the CAS engine before the data is used in a production environment.

Because ASCII encoding is compatible with UTF-8 encoding, no transcoding is necessary.

Indexes and format catalogs must also be compatible with UTF-8 session encoding.

For additional information about transcoding SAS data sets to UTF-8, see these publications:

- [Migrating Data to UTF-8 for SAS Viya](#)
- [SAS Viya FAQ for Processing UTF-8 Data](#)

## See Also

[“Avoiding Character Data Truncation By Using the CVP Engine” in SAS National Language Support \(NLS\): Reference Guide](#)

---

## Caslibs

All data is available to the CAS server through caslibs and all operations in the CAS server that use data are performed using caslibs. A caslib provides access to files in a data source, such as a database or a file system directory, and to in-memory tables. Access controls are associated with caslibs to manage access to data. You can think of a caslib as a container where the container has two areas where data is referenced: a physical space that includes the source data or files, and an in-memory space that makes the data available for CAS action processing.

Caslibs can be of the following types:

- personal and automatically available for a given user (analogous to the Sasuser library, but in-memory)
- pre-defined by an administrator
- manually added

Authorized users can add or manage caslibs with the CASLIB statement. Caslib authorization is set by your administrator. In some instances, such as when you copy native CAS tables that are not in-memory, a caslib is required although data is not copied to the caslib.

When you start a CAS session, your personal caslib is added by default. It is the active caslib and is named *Casuser(my-user-ID)*. When you specify the caslib in your code, you use only the name *Casuser*, you do not need to specify *Casuser(my-user-ID)*. Your personal caslib is available only to you. To share tables in your caslib, make your table available in another caslib that is accessible by other users. Your site administrator can add caslibs with global scope for anyone to use.

The following output lists the metadata that is associated with a personal caslib. The code to obtain this information is `caslib casuser list;`.

```
57      caslib casuser list;
NOTE: Session = MYSESSION Name = CASUSER(my-user-ID)
      Type = PATH
      Description = Personal File System Caslib
      Path = /home/myuserID/
      Definition =
      Subdirs = Yes
      Local = No
      Active = Yes
      Personal = Yes
```

This caslib is a path-based caslib that enables the user specified by *my-user-ID* to save in-memory tables as files in the specified path and to perform a server-side load of data from the path.

You use the CASLIB statement to add a new caslib if you are authorized to do so. Only one caslib can be active in a session at a time. When you use the CASLIB statement, that caslib becomes the active caslib. If an additional caslib is added, that one becomes the active caslib. If that second one is dropped, the session returns to use the initially active caslib.

The active caslib is used as the default data source if you do not override it. You can override the active caslib using the CASUTIL procedure INCASLIB and OUTCASLIB= options or the CASLIB= LIBNAME option.

To view all caslibs, enter `caslib _all_ list;`.

SAS Studio provides a **Cloud Analytic Services** snippet that adds a new caslib:

```
9 /*****
10 /* Create a CAS library (myCaslib) for the specified path ("/filePath/") */
11 /* and session (mySession). If "sessref=" is omitted, the caslib is */
12 /* created and activated for the current session. Setting global makes */
13 /* myCaslib visible in all active sessions. The default is local. Setting */
14 /* subdirs extends the scope of myCaslib to subdirectories of "/filePath". */
15 /*****/
16
17 caslib myCaslib datasource=(srctype="path") path="/filePath/" sessref=mySession global subdirs;
```

Specify the values that are particular for your environment.

In the following example, the CASLIB statement adds the caslib *Myvapublic* for a Hadoop Distributed File System (HDFS) data source.

```
caslib Myvapublic path="/vapublic"
      datasource=(srctype="hdfs");
```

Alternatively, you can set the active caslib for your session with the CASLIB session option in the CAS statement:

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

[Video: Accessing, Manipulating, and Saving Data Using SAS Cloud Analytic Services.](#)

## See Also

- [“Caslibs” in SAS Cloud Analytic Services: Fundamentals](#)
- [“CASLIB Statement” in SAS Cloud Analytic Services: User’s Guide](#)
- [“CASLIB= LIBNAME Option” in SAS Cloud Analytic Services: User’s Guide](#)
- [“CASUTIL Procedure” in SAS Cloud Analytic Services: User’s Guide](#)

---

## The CAS LIBNAME Engine

The CAS LIBNAME engine performs client/server communication. The engine is part of the SAS client and connects to the CAS server. The CAS libref is your handle to communicate from SAS with the specific session.

The CAS LIBNAME engine provides two important uses for SAS programming:

- The CAS libref and table name are used to identify in-memory tables that can then be used as input data for SAS Viya procedures.
- The engine enables data transfer between the SAS session and the CAS session.

**Note:** Transferring large data sets between SAS and the server causes poor performance. To prevent SAS from appearing to be unresponsive, the LIBNAME engine includes a DATALIMIT= option that limits data transfer to 100 M.

When you assign a CAS engine libref, the CAS LIBNAME engine associates the libref with the active caslib unless you specify the LIBNAME statement CASLIB= option. If you use the CASLIB= option, the CAS engine libref is bound to the caslib. Binding the CAS engine libref with the active caslib prevents the CAS engine libref from switching to a subsequently added caslib.

Here are two examples of the CAS LIBNAME statement:

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

```
libname mycas cas;
```

```
/* The libref mycas is bound to the testTables caslib,          */
/* even if the active caslib changes to a caslib other than testTables. */
```

```
libname mycas cas caslib=testTables;
```

For a complete example that uses the CAS LIBNAME engine, see [Use the SAS Windowing Environment to Connect to the CAS Server and Run the FACTMAC Procedure on page 17](#).

**Note:** Downloading a CAS table with millions of rows can cause network degradation. Use the DATALIMIT= LIBNAME option, the DATALIMIT= DATA set option, or the CASDATALIMIT= system option. For more information about these options, see [SAS Cloud Analytic Services: User’s Guide](#).

## See Also

“CAS LIBNAME Statement” in *SAS Cloud Analytic Services: User’s Guide*

## Load Data to a Caslib

You can load a SAS data set, database tables, and more to a caslib. After the data is in a caslib, you can use a DATA step, procedures, CAS actions, PROC DS2, or PROC FEDSQL, operations on the CAS table. Tables are not automatically saved when they are loaded to a caslib. You can use PROC CASUTIL to save tables. Native CAS tables have the file extension .sashdat. For information about file types that are supported in CAS, see “Path-Based Data Source Types and Options” in *SAS Cloud Analytic Services: User’s Guide*.

## Use the DATA Step to Load Data

This example uses the DATA step to perform client-side load of a data set from the Sashelp library to the caslib Mycaslib. PROC CASUTIL saves the table.

```
cas casauto;                                /* 1 */

caslib mycaslib datasource=(src="path")
  path="/myCasTables/" sessref=casauto;    /* 2 */

libname mycas cas caslib=mycaslib;        /* 3 */

data mycas.heart;
  set sashelp.heart;
  keep status BP_status weight_status
      smoking_status chol_status deathcause; /* 4 */
run;

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

proc casutil incaslib="mycaslib" outcaslib="mycaslib"; /* 6 */
  save casdata="heart" replace;
run;
```

- 1 Start the CAS session and name it CASAUTO.
- 2 Add the caslib Mycaslib. The data source for this caslib is a server directory path, and the caslib is associated with the CASAUTO session. If you omit the SESSREF option, the caslib is associated with the most recently started session.
- 3 Assign a CAS LIBNAME libref to load the data to, and to bind the libref Mycas to the caslib Mycaslib.
- 4 Keep a limited number of variables from the original data set.
- 5 View the results. The table Mycas.Heart is an in-memory table. This table exists in memory for the duration of your CAS session. PROC PRINT performs data transfer of the five rows from the mycas.heart table on the CAS server to the SAS session.

**Note:** Printing large SAS Cloud Analytic Services tables can cause a large amount network traffic. You can use the OBS= system option to limit the number of rows the PRINT procedure processes.

- 6 Save the table to the /myCasTables directory on the server by using the SAVE CASDATA statement in PROC CASUTIL. To save a table in CAS, you specify the caslib and the table.

Video: [Understanding Caslibs and Loading Data in SAS Viya](#)

## See Also

- [“CASUTIL Procedure” in SAS Cloud Analytic Services: User’s Guide](#)
- [“Tables Action Set Details” in SAS Viya: System Programming Guide](#)
- [“Data Lifecycle” in SAS Cloud Analytic Services: Fundamentals](#)

## Use PROC CASUTIL to Load Data

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

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

proc casutil;
    list files;(* 2 */

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

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

quit;

run;
```

- 1 Add a caslib that uses Oracle as the data source. Oralib becomes the active caslib for the session and the subsequent programming statements use it for input and output.
- 2 The LIST FILES statement displays the tables that are available in the Oracle database.
- 3 The DROPTABLE statement includes the QUIET option. Running this statement is useful on repeated runs because it ensures that no table named Sales can be in-memory to interfere with the subsequent LOAD CASDATA= statement.
- 4 Because the first CONTENTS statement follows the DROPTABLE statement, the table information and column information from Oracle are read.
- 5 The CASDATA= argument in the LOAD statement indicates that the Sales table is read from the caslib's data source (Oracle) into SAS Cloud Analytic Services. Options are specified to add labels to the table and columns.

[Video: Loading Data](#)

**See Also**

“CASUTIL Procedure” in *SAS Cloud Analytic Services: User’s Guide*

**Use CAS Actions to Load Data**

The CAS procedure interacts with the CAS server by using CAS actions. An advantage to using CAS actions for table management is that the table actions have more features that you can use in PROC CAS than are available in PROC CASUTIL. Here is an example that loads the iris.sashdat table from the server to the active caslib and names the table IRIS:

```
proc cas;
  table.loadtable /path="iris.sashdat" casOut={name="IRIS"};
run;
```

Some of the features available in the Tables action set that are not part of PROC CASUTIL are managing caslibs, creating user-defined table attributes, fetching table rows from a table, and updating table rows. See [Table Action Set: Syntax](#) for a complete list of table actions.

**See Also**

- “CAS Procedure” in *SAS Cloud Analytic Services: CASL Reference*
- [CAS Actions](#)

---

## Programming Examples Using SAS Cloud Analytic Services

**Example Data**

Some examples use data from the Sashelp library. If a data set is not in the Sashelp library, you can install the sample data in the Sashelp library if you have sudo privileges. The instructions for installing the sample data sets are in the deployment guide at [support.sas.com](https://support.sas.com). Contact your administrator for assistance.

To download CSV files, see [SAS Viya Example Data Sets](#).

**Use SAS Studio to Connect to the CAS Server and Run a Gradient Boosting Model Procedure**

The CAS server connection is created for you when you [sign on to SAS Studio](#). The SAS system options that configure the CAS server, CASHOST= and CASPORT=, are set during deployment.

After you are signed in to SAS Studio, initiate a session with the server by using this CAS statement:

```
cas myCasSess;
```

This example starts a session with the CAS server, loads data to the server, and creates a gradient boosting model using the GRADBOOST procedure.

```
cas casauto;                                /* 1 */
libname mycas cas;                          /* 2 */

data mycas.junkmail;                        /* 3 */
```



```

set sashelp.junkmail;
run;

proc gradboost data=mycas.junkmail outmodel=mycas.gradboost_model; /* 4 */
  input Address Addresses All Bracket Business CS CapAvg CapLong
        CapTotal Conference Credit Data Direct Dollar Edu Email
        Exclamation Font Free George HP HPL Internet Lab Labs
        Mail Make Meeting Money Order Original Our Over PM Paren
        Parts People Pound Project RE Receive Remove Semicolon
        Table Technology Telnet Will You Your _000 _85 _415 _650
        _857 _1999 _3D / level = interval;
  target class /level=nominal;
  output out=mycas.score_at_runtime; /* 5 */
  ods output FitStatistics=fit_at_runtime;
run;

```

- 1 The CAS statement connects to the CAS server and creates a session with the default name CASAUTO on that server. This step also sets Casuser as the active caslib for the CASAUTO session.
- 2 The LIBNAME statement creates a new SAS libref, Mycas, and assigns it to the CAS engine.
- 3 The DATA step loads the Sashelp.Junkmail data set to the server-side table Junkmail in the Casuser caslib.
- 4 Execute the GRADBOOST procedure. The input data comes from the Casuser caslib. The same is true for the output model table Gradboost\_Model.
- 5 The OUTPUT statement scores the training data and saves the results to a new table named fit\_at\_runtime.

To further explore this example and to view the output, see the [GRADBOOST procedure](#).

## See Also

- [CAS Sessions on page 4](#)
- [Caslibs on page 11](#)
- [The CAS LIBNAME Engine on page 13](#)
- [Load Data to a Caslib on page 14](#)

## Use the SAS Windowing Environment to Connect to the CAS Server and Run the FACTMAC Procedure

After connecting to the CAS server and establishing a session using the CAS statement, the CAS LIBNAME engine libref is your handle to communicate with the specific session from a SAS programming client.

### Note:

Authentication is used to control access to the CAS server and its resources. Your identity must be successfully authenticated before your session is created.

SAS Studio and SAS Enterprise Guide authenticate the connection to CAS by using your user credentials. When password information is not available, an attempt is made to find an authinfo file (.authinfo is the default filename on Linux). The authinfo file provides a user name and password to CAS for PAM authentication. For more information, see [Authentication Mechanisms](#).

The SAS windowing environment requires an authinfo file for authentication. For information about creating an authinfo file, see [“Create an Authinfo File” in Client Authentication Using an Authinfo File](#) or contact your site administrator.

The example below demonstrates the following concepts:

- Connection information is required to connect to CAS from the SAS windowing environment.
- You must explicitly start a CAS session from the SAS windowing environment.
- The CAS LIBNAME engine is required when you use a SAS procedure to transfer data from the CAS server.
- SAS Viya analytic procedures use CAS processing, so the engine provides the way to identify the tables to use for analysis.

You can download the compressed archive file from the website at <http://files.grouplens.org/datasets/movielens/ml-100k.zip> and use any third-party unzip tool to extract all the files in the archive to the destination directory of your choice. The file that contains the ratings is `u.data`. The following CASUTIL procedure loads the data table from the directory into your CAS session:

```
option casport=5570 cashost="cloud.example.com";           /* 1 */
cas casauto sessopts=(caslib=casuser)                     /* 2 */
libname mycas cas;                                         /* 3 */

proc casutil;                                              /* 4 */
  load file="your-file-path/data/u.data"
  casout="movlens"
  importoptions=(filetype="CSV" delimiter="TAB" getnames="FALSE"
    vars=("userid" "itemid" "rating" "timestamp"));
run;

proc factmac data=mycas.movlens nfactors=10 learnstep=0.15 /* 5 */
  maxiter=20 outmodel=mycas.factors;

  input userid itemid /level=nominal;
  target rating /level=interval;
  output out=mycas.out1 copyvars=(userid itemid rating);
run;

proc print data=mycas.factors(obs=10);                     /* 6 */
run;
cas casauto terminate;                                     /* 7 */
```

- 1 Specify your connection information. The CASHOST= and CASPORT= system options specify the connection information. As an alternative, your administrator might have set the default host and port for your server in a configuration file.
- 2 Start a session and set the active caslib to Casuser. You can start a session manually with the CAS statement. In this example, the CAS statement starts the CAS session named CASAUTO.
- 3 Specify the CAS engine LIBNAME statement and use the libref with the input table name. The LIBNAME statement assigns a CAS engine libref named Mycas that you use to connect to the session CASAUTO. In subsequent procedure steps, input table names must begin with the CAS engine libref named Mycas.

Some procedures require output table names to also begin with the CAS engine libref. See the documentation for your procedure for more information. PROC CASUTIL require a CAS engine libref on both the input and output table names. For documentation about the CAS engine, see [SAS Cloud Analytic Services: User's Guide](#).

- 4 Add the u.Data data set as an in-memory table named Movlens.
- 5 Execute the FACTMAC procedure. The input data comes from the Casuser caslib. Use the Mycas libref to identify the input table for a SAS Viya procedure.
- 6 View ten rows from the new data. The engine performs data transfer from the CAS server to SAS.

- 7 At the end of your program, when you no longer need to access data in CAS, you can terminate your session to preserve resources. If you do not terminate your session at the end of your program, the session has a time-out interval and eventually terminates on its own.

## A Factorization Machine Model Using the Factmac CAS Action

You can also do factorization by using the factmac CAS action. You can use the movie recommendation example in [Use the SAS Windowing Environment to Connect to the CAS Server and Run the FACTMAC Procedure](#) to compare programming using CAS actions and procedures.

This example uses the factmac action, which uses observations in a data table to train a factorization machine model. In this example, the data is derived from companies that provide movies for online viewing. A company wants to offer its customers recommendations of movies that they might like. These recommendations are based on ratings that are provided by users.

```
/*In the SAS Windowing Environment, you must specify the connection information*/
option casport=your-port-number cashost="cloud.example.com";
cas casauto sessopts=(caslib=casuser);                                /* 1 */

proc casutil;                                                         /* 2 */
  load file="u.data"
  casout="movlens"
  importoptions=(filetype="CSV" delimiter="TAB" getnames="FALSE"
                 vars=("userid" "itemid" "rating" "timestamp"));
run;

proc cas;                                                             /* 3 */
  factmac result=R / table={name="movlens"},                          /* 4 */
  outModel={name="factors_out", replace=true},                       /* 5 */
  inputs={"userid", "itemid"},                                       /* 6 */
  nominals={"userid", "itemid"},                                     /* 7 */
  target="rating",                                                  /* 8 */
  maxIter=20, nFactors=10, learnStep=0.15,                          /* 9 */
  output={casout={name="score_out", replace="TRUE"},               /* 10 */
  copyvars={"userid","itemid","rating"}};
run;

print r;
```

- 1 Create a session with the CAS server and use the Casuser caslib. Casuser is the personal caslib that is set up for your user ID by the administrator.
- 2 Use the CASUTIL procedure to load the data to the in-memory table Movlens using the import options that are specified in the IMPORTOPTIONS= option. The file `u.data` contains the movie ratings.

PROC CASUTIL is a data management procedure for managing CAS server tables.

- 3 Execute the CAS actions with the CAS procedure.  
PROC CAS executes CAS actions. In addition, you can code by using the new CAS language (CASL) within PROC CAS.
- 4 Execute the factmac action from the factmac action set, and place the results in the variable R. The table parameter names the input data table to be analyzed. An action sends a request to the server and returns a result. The results are usually a data table or a metadata table.
- 5 The `outModel` parameter creates output for the fitted parameter estimates to the `factors_out` data table.
- 6 The `inputs` parameter specifies the input variables.

- 7 The `nominals` parameter specifies that the `userid` and `itemid` variables are nominal.
- 8 The `target` parameter specifies the target variable.
- 9 The `maxIter` parameter specifies the number of iterations for the optimization. The `nFactors` parameter specifies the number of factors to be estimated for the factorization machine model. The `learnStep` parameter specifies the learning step size for the optimization.
- 10 The `output` parameter names the in-memory table in the `casout` parameter and specifies that an existing table is to be replaced.

For more information about this example, see the [MovieLens data example for the Factorization Machine action set](#).

### See Also

- [CAS Actions on page 34](#)
- [SAS Cloud Analytic Services: CASL Reference](#)

## Getting Started with Machine Learning

[Getting Started with SAS Visual Data Mining and Machine Learning](#) is a case study that demonstrates the new high-performance analytic procedures. You can explore and prepare data, create and assess models, and score new data. Follow along with this case study by downloading the data and copying the code to the SAS Studio programming interface.

---

## Data Types

The CAS server supports the VARCHAR, INT32, INT64, IMAGE data type in addition to the CHARACTER and NUMERIC data types, which are traditionally supported by SAS.

Variables that are created using the VARCHAR data type are varying width and use character semantics, rather than being fixed-width and using the byte semantics of the traditional CHARACTER data type. Using the VARCHAR data type in the DATA step in the CAS server has some restrictions. For more information, see [“Restrictions for the VARCHAR Data Type in the CAS Engine” in SAS Cloud Analytic Services: User’s Guide](#).

Variables that are created or loaded using the INT32 or INT64 data types support more digits of precision than the traditional NUMERIC data type. All calculations that occur on the CAS engine maintain the INT32 or INT64 data type. Calculations in DATA steps or procedures that run on the SAS 9 engine are converted to NUMERIC values.

The CHARACTER and NUMERIC data types continue to be the supported data types for processing in the SAS Workspace Server.

The DS2 language supports several additional [data types](#). On the CAS server, DS2 converts non-native data types to CHARACTER, NUMERIC, or VARCHAR. For information about data types that are supported for specific data sources, see [“Data Type Reference” in SAS DS2 Language Reference](#).

The CAS language (CASL) determines the data type of a variable when the variable is assigned.

In the following table, the letter Y indicates the data types that are supported for programming on the CAS server. In the last column, Y indicates data types that are supported on the SAS Workspace Server.

Data Type	CAS Actions	CASL	Data Connectors ***	Procedures and DATA Step	DS2	FedSQL	Workspace Server Processing*
BIGINT			Y		Y		
BLOB		Y					
BOOLEAN		Y	Y				
CHARACTER (CHAR)	Y	Y	Y	Y	Y	Y	Y
DATE		Y	Y		Y		
DATETIME		Y	Y				
DOUBLE**	Y	Y	Y	Y	Y	Y	
FLOAT			Y		Y		
IMAGE	Y						
INTEGER			Y		Y		
INT32		Y	Y			Y	
INT64		Y	Y			Y	
ITEMS		Y					
LISTS		Y					
NCHAR			Y		Y		
NUMERIC (NUM)			Y				Y **
NVARCHAR			Y		Y		
SMALLINT			Y		Y		
STRING UTF-8		Y					
TABLE		Y					
TIME		Y	Y		Y		
TIMESTAMP			Y		Y		
TINYINT			Y		Y		
VARCHAR	Y	Y	Y	Y	Y	Y	

\* Some of the SAS utility procedures read CAS tables or metadata about the tables, but do not run on the CAS server. For example, the PRINT procedure reads and prints CAS tables, including VARCHAR columns. For a list of utility procedures that support CAS data, see [SAS Viya Foundation Procedures](#).

\*\* The SAS NUMERIC data type is the same as a DOUBLE data type.

\*\*\* Additional data types are supported by the data connectors. These data types are first converted to data types that can be processed on the CAS server. Check the data connector documentation for your data source to ensure that a data type is supported. For more information, see ["Data Connectors" in SAS Cloud Analytic Services: User's Guide](#).

**Note:** The CVP LIBNAME engine converts the CHAR data type to VARCHAR if you specify the `CVPVARCHAR=YES` LIBNAME option.

## Statistical Procedures Available on SAS Viya

The statistical procedures available on SAS Viya run only on in-memory CAS tables. You must license and install SAS Visual Statistic to use these procedures.

For examples, see the individual procedures listed in the table.

Procedure	Description
<a href="#">ASSESS</a>	Assesses and compares supervised learning models. <ul style="list-style-type: none"> <li>■ For a supervised learning model that has a nominal target, the ASSESS procedure produces lift information and receiver operating characteristic (ROC) information.</li> <li>■ For a regression model, the ASSESS procedure performs a quantile binning of the predictions, and then returns the summary statistics of the response variable for each bin.</li> <li>■ Calculates fit statistics, such as average square error, mean square logarithmic error, mean absolute error, mean consequential error, and multiclass log loss.</li> </ul>
<a href="#">BINNING</a>	Performs binning. The BINNING procedure supports several binning methods and can calculate the weight of evidence (WOE) and information value (IV), based on binning results. <a href="#">Video: Transforming Data</a>
<a href="#">CARDINALITY</a>	Determines a variable's cardinality or limited cardinality. <a href="#">Video: Selecting Features</a>
<a href="#">CORRELATION</a>	Computes Pearson correlation coefficients and probabilities.
<a href="#">FREQTAB</a>	Produces one-way to <i>n</i> -way frequency and crosstabulation (contingency) tables.
<a href="#">GAMMOD</a>	Fits generalized additive models that are based on low-rank regression splines.
<a href="#">GENSELECT</a>	Fits and performs model selection for generalized linear models.
<a href="#">KCLUS</a>	Performs clustering, which is a common step in data exploration. <a href="#">Video: K-Means Clustering</a> <a href="#">Video: Principal Component Analysis</a>
<a href="#">LOGSELECT</a>	Fits and performs model selection for logistic regression models, including binary, binomial, and multinomial response models.
<a href="#">MDSUMMARY</a>	Computes basic descriptive statistics in parallel for CAS tables.

Procedure	Description
NLMOD	Fits nonlinear regression models with standard or general distributions.
PARTITION	Performs simple random sampling, stratified sampling, and oversampling to produce a table that contains a subset of observations or partitioned observations. <a href="#">Video: Exploring Data</a>
PCA	Performs principal component analysis, which is a multivariate technique for examining relationships among several quantitative variables. <a href="#">Video: Principal Component Analysis</a>
PHSELECT	Fits the Cox proportional hazards regression models for survival data and performs variable selection.
PLSMOD	Fits reduced-rank linear models by using any one of a number of linear predictive methods, including partial least squares (PLS).
QTRSELECT	Fits and performs model selection for quantile regression models.
REGSELECT	Fits and performs model selection for ordinary linear least squares models.
SPC	Performs Shewhart control chart analysis.
TREESPLIT	Builds tree-based statistical models for classification and regression. <a href="#">Video: Auto-tuning: Decision Tree</a>
VARIMPUTE	Performs numeric variable imputation. <a href="#">Video: Transforming Data</a>
VARREDUCE	Performs both supervised and unsupervised variable selection. <a href="#">Video: Selecting Features</a>

This example performs a simple random sampling:

```

cas casauto;                                /* 1 */
libname mycas cas caslib=Casuser;          /* 2 */

filename hmeq url 'http://support.sas.com/documentation/
                onlinedoc/viya/exampledatasets/hmeq.csv';    /* 3 */
data mycas.hmeq;                             /* 4 */
    infile hmeq dsd firstobs=2 ;
    input bad loan mortdue value reason $ job $ yoJ
          derog delinq clage ninq clno debtinc;
    keep job reason loan value delinq derog;    /* 5 */
run;

proc partition data=mycas.hmeq samppct=10 seed=10 nthreads=1;    /* 6 */
    output out=mycas.out2 copyvars=(job reason loan value delinq derog); /* 7 */
    display 'SRSFreq';                                           /* 8 */

```

```
run;

proc print data=mycas.out2(obs=20);
run;
```

- 1 Start the CAS session CASAUTO.
- 2 The LIBNAME statement assigns the Mycas libref and binds it to the Casuser caslib.
- 3 Use the FILENAME URL access method to access the Hmeq.csv file at <http://support.sas.com/documentation/onlinedoc/viya/EXAMPLEDATASETS/HMEQ.CSV>.
- 4 Load the Hmeq data set to CAS using the DATA step and the CAS LIBNAME libref, Mycas (Mycas is automatically associated with the Casuser caslib from step 2). Notice that the output is created in your Mycas library as well as in your Casuser caslib.
- 5 Keep a few key variables.
- 6 Perform a simple random sampling. The SAMPPCT=10 option requests that 10% of the input data be sampled.
- 7 Using the OUTPUT statement, request that the sampled data be stored in a table named mycas.out2. The COPYVARS= option lists the variables to be copied from mycas.hmeq to mycas.out2.
- 8 Display the SRSFreq ODS table using the DISPLAY statement.

The PARTITION Procedure

Simple Random Sampling Frequency	
Number of Obs	Number of Samples
5960	596

## Data Mining and Machine Learning Procedures

CAS procedures enable you to have the familiar experience of coding SAS procedures, but behind each procedure statement is one or more CAS actions that run across multiple machines. You must have SAS Visual Data Mining and Machine Learning licensed and installed to use these procedures.

For examples, see the individual procedures listed in the table.

Procedure	Description
<a href="#">ASTORE</a>	Creates score code that can use code from an analytic store, as well as DS2 code. The procedure also has some analytic store management functions.
<a href="#">BNET</a>	Learns a Bayesian network from an input data set.
<a href="#">BOOLRULE</a>	Enables you to extract Boolean rules from large-scale transactional data. The BOOLRULE procedure can automatically generate a set of Boolean rules by analyzing a text corpus that has been processed by the TEXTMINE procedure and that is represented in a transactional format.



Procedure	Description
FACTMAC	<p>Implements the factorization machine model. The flexible factorization machine model has applications in predictive modeling and recommendation.</p> <p><a href="#">Video: Factorization Machines</a></p>
FASTKNN	<p>Implements the <math>k</math>-nearest neighbor (<math>k</math>-NN) search algorithm.</p>
FISM	<p>Performs frequent item set mining, which looks for frequent patterns in a large database. The FISM procedure finds frequent patterns by using the FP-growth (frequent pattern growth) algorithm of Han, Pei, and Yin (2000).</p>
FOREST	<p>Creates a predictive model called a forest, which consists of several decision trees.</p> <p><a href="#">Video: Auto-tuning: Forest</a></p> <p><a href="#">Video: Ensemble Methods: Forest</a></p>
GRADBOOST	<p>Creates a predictive gradient boosting model, which consists of multiple decision trees. The procedure samples the original data without replacement to create the training data for an individual tree, and performs the action of sampling multiple times throughout a run.</p> <p><a href="#">Video: Ensemble Methods: Gradient Boosting</a></p>
GVARCLUS	<p>Performs variable clustering and graphical modeling. The procedure divides a set of variables into disjoint clusters and creates tables that contain the edge and vertex information for defining an undirected graph.</p>
MBANALYSIS	<p>Performs association rule mining on a transaction database.</p>
MWPCA	<p>Implements the moving windows robust principal component analysis. You can use this procedure to capture changes in principal components over time by using sliding windows. Also, you can choose that the principal component analysis that is performed on each window be robust. In a robust analysis, outliers and noise are excluded from each window before the analysis is performed.</p>
NETWORK	<p>Provides a number of network analysis algorithms that use an abstract graph or network as input, help explain network structure, and compute important network measures. Depending on the application, this type of network analysis can stand on its own to provide independent value, or it can support machine learning models.</p> <p><a href="#">Video: Community Detection in Networks</a></p>
NNET	<p>Trains a multilayer perceptron neural network. This procedure can also use a previously trained network to score a data table, which is referred to as stand-alone scoring. Or, it can generate SAS DATA step statements that can be used to score a data table.</p> <p><a href="#">Video: Auto-tuning: Neural Network</a></p>

Procedure	Description
<a href="#">RPCA</a>	Implements the robust principal component analysis. RPCA has applicability in many areas including image processing, latent semantic indexing, ranking, and matrix completion.
<a href="#">SVDD</a>	Implements the support vector data description (SVDD), a machine learning algorithm. SVDD is a one-class classification technique that is useful in applications where data that belongs to one class is abundant but data about any other class is scarce or missing. Fraud detection and process control are some examples of application areas where the majority of the data belongs to one class.
<a href="#">SVMACHINE</a>	Implements the support vector machines (SVM) algorithm. The SVM algorithm computes support-vector machine-learning classifiers for the binary pattern recognition problem. This method has been broadly used in fields such as image classification, handwriting recognition, financial decision-making, and text mining.
<a href="#">TEXTMINE</a>	Integrates natural language processing and statistical analysis to analyze large-scale textual data. <a href="#">Video: Text Mining</a>
<a href="#">TMSCORE</a>	Scores textual data. In text mining, scoring is the process of applying parsing and singular-value decomposition projections to new textual data. <a href="#">Video: Text Mining</a>

The following example of PROC FACTMAC recommends movies that are based on user ratings. It is similar to the [CAS actions factorization example](#) that recommends movies. You can use these two examples to see the differences between programming using procedures and using CAS actions.

```
proc casutil;                                /* 1 */
  load file="~/data/mydata.csv"
  casout="movlens"                            /* 2 */
  importoptions=(filetype="CSV" delimiter="TAB" getnames="FALSE" /* 3 */
    vars=("userid" "itemid" "rating" "timestamp"));
run;

proc factmac data=mycas.movlens nfactors=10 learnstep=0.15
  maxiter=20 outmodel=mycas.factors;          /* 4 */
  input userid itemid /level=nominal;
  target rating /level=interval;
  output out=mycas.out1 copyvars=(userid itemid rating);
run;

proc print data=mycas.factors(obs=10);        /* 5 */
run;
```

- 1 Load the data.
- 2 Specify the name of the in-memory table.
- 3 Specify options for importing a CSV file and the variables to use in the table.

- 4 Use PROC FACTMAC to predict movie ratings.
- 5 Print the first 10 rows of the output table.

To see the results, see [PROC FACTMAC](#).

## SAS Econometrics Procedures

SAS Econometrics provides a set of procedures that provide techniques to model complex business and economic scenarios and to analyze the dynamic impact that specific events might have over time. You must have SAS Econometrics licensed and installed in order to use these procedures.

Procedure	Description
<a href="#">CCDM</a>	Computes an estimate of the compound distribution model by using the distribution models of X and N.
<a href="#">CCOPULA</a>	Performs large-scale multivariate simulation from separate models, each of which can be fitted using different distributional specifications. The specifications can be non-normal.
<a href="#">CNTSELECT</a>	Enables you to analyze regression models in which the dependent variable takes nonnegative integer values or count values.
<a href="#">CPANEL</a>	Analyzes a class of linear econometric models that arise when time series and cross-sectional data are combined.
<a href="#">CQLIM</a>	Analyzes univariate limited dependent variable models in which dependent variables take discrete values or are observed only in a limited range of values. This procedure is a cloud-enabled version of the QLIM procedure in SAS/ETS software.
<a href="#">CSPATIALREG</a>	Analyzes a class of linear spatial econometric models for cross-sectional data whose observations are spatially referenced or georeferenced.
<a href="#">HMM</a>	Supports hidden Markov models (HMMs), which have been widely applied in economics, finance, science, and engineering.
<a href="#">SEVSELECT</a>	Estimates parameters of any arbitrary continuous probability distribution that is used to model the magnitude (severity) of a continuous-valued event of interest.
<a href="#">TSMODEL</a>	Executes user-defined programs on time series data.

## SAS Visual Forecasting Procedures

SAS Visual Forecasting procedures provide automatic variable, event, and model selection. You must have SAS Visual Forecasting licensed and installed in order to use these procedures.

Procedure	Description
<a href="#">TSINFO</a>	Evaluates a variable in an input data table for its suitability as a time ID variable in SAS procedures and solutions that are used for time series analysis.
<a href="#">TSMODEL</a>	Analyzes time-stamped transactional data with respect to time and accumulates the data into a time series format. The TSMODEL procedure forms time series vectors from time-stamped data, and then provides these vectors as array variables for subsequent processing by your program statements.
<a href="#">TSRECONCILE</a>	Reconciles forecasts of time series data at two different levels of a hierarchy in a top-down fashion when the input data are contained in CAS tables.

## SAS Optimization Procedures

SAS Optimization software is a set of procedures for exploring models of distribution networks, production systems, resource allocation problems, and scheduling problems. These procedures use the tools of operations research. You must have SAS Optimization licensed and installed in order to use these procedures.

Procedure	Description
Mathematical Optimization Procedures	
<a href="#">CLP</a>	The CLP procedure is a finite-domain constraint programming solver for constraint satisfaction problems (CSPs) with linear, logical, and global constraints.
<a href="#">OPTLP</a>	Solves linear programming problems that are submitted in a SAS data table that uses a mathematical programming system (MPS) format.
<a href="#">OPTMILP</a>	Solves general and mixed integer linear programming (MILP) problems in which a subset of the decision variables are constrained to be integers. The OPTMILP procedure solves MILP problems with an LP-based branch-and-bound algorithm, augmented by advanced techniques such as cutting planes and primal heuristics.

Procedure	Description
<a href="#">OPTQP</a>	Solves quadratic problems using the interior point algorithm. These are problems with a quadratic objective function and a collection of linear constraints, including general linear constraints along with lower or upper bounds (or both) on the decision variables.
<a href="#">OPTMODEL</a>	Includes the OPTMODEL modeling language and provides a modeling environment for building, solving, and maintaining optimization models.
Network Optimization Procedures	
<a href="#">OPTNETWORK</a>	Includes a number of graph theory and network optimization algorithms that can augment more generic mathematical optimization approaches. Many practical applications of optimization depend on an underlying network. For example, retailers face the problem of shipping goods from warehouses to stores in a distribution network to satisfy demand at minimum cost. Commuters choose routes in a road network to travel from home to work in the shortest amount of time.

## SAS Viya Foundation Procedures

The core product of SAS Viya is SAS Visual Analytics. If you install SAS Visual Analytics only, then you have access to the Base SAS procedures in the following table. If you have SAS 9.4M5 installed, all SAS 9.4 Base procedures run on SAS Viya. If you have SAS Viya with any offering in addition to SAS Visual Analytics licensed and installed, you also have access to all SAS 9.4 Base procedures. For all Base procedure documentation, see [Base SAS Procedures Guide](#).

Most of these procedures run on the SAS Workspace Server and not on the CAS server. The data or metadata that these procedures use transfer data from the CAS server to the SAS Workspace Server. To perform operations on CAS tables, the CAS LIBNAME engine can connect a SAS 9.4 session to an existing SAS Cloud Analytic Services (CAS) session through the CAS session name or through the CAS session UUID. Large tables might cause performance degradations.

Beginning with SAS 9.4M5, some Base SAS procedures are enhanced to process data insideCAS with CAS actions. For more information, see [“CAS Processing of Base Procedures”](#) in [Base SAS Procedures Guide](#).

**Table 1** SAS Foundation Procedures That Are Available for Sites with Only SAS Viya and SAS Visual Analytics Installed

Procedure	Description	Uses CAS Processing
<a href="#">APPEND</a>	Adds rows from a CAS table to the end of a SAS data set, and adds rows from a SAS data set to the end of a CAS table.	Yes
<a href="#">CAS</a>	Provides a programming environment that is based on the CASL language specification. This environment enables you to interact with SAS Cloud Analytic Services (CAS) from the SAS client.	Yes

Procedure	Description	Uses CAS Processing
CASUTIL	Works with tables in SAS Cloud Analytic Services, SAS data sets in SAS libraries, and external files.	Yes
CATALOG	Manages entries in SAS catalogs.	No
COMPARE	Compares the contents of two SAS data sets, selected variables in different data sets, or variables within the same data set.	No
CONTENTS	Shows the contents of a CAS table and prints the directory of the caslib.	Yes
COPY	Copies tables to and from libraries.	Yes
DATASETS	Manages CAS tables.	Yes
DELETE	Deletes SAS data sets and CAS tables.	Yes
DOWNLOAD	In SAS/CONNECT, copies to the client SAS files that are stored on the server.	No
DS2	Runs DS2 code.	Yes
DSTODS2	Converts DATA step code to DS2 code.	No
EXPORT	Writes a SAS data set to delimited files or to JMP files.	No SAS/ACCESS to PC Files must be purchased to add support for other file types, such as Microsoft Access databases for DBMS=ACCESS, Microsoft Excel workbooks, dBase database (DBF) files, and IBM Lotus spreadsheets.
FCMP	Enables you to create, test, and store SAS functions, CALL routines, and subroutines before you use them in other SAS procedures or in DATA steps.	Yes
FMTC2ITM	Converts one or more format catalogs into a single item store.	No
FEDSQL	Submits FedSQL code.	Yes, with <a href="#">limitations on page 39</a>
FORMAT	Creates user-defined formats.	Yes PROC FORMAT stores user-defined formats in a CAS format library, which is associated with a CAS session. CAS user-defined formats must be associated with a variable before CAS processes a table.
HADOOP	Reads and writes Hadoop data.	No
HDMD	Generates XML-based metadata that describes the contents of files that are stored in HDFS.	No

Procedure	Description	Uses CAS Processing
HTTP	Processes data from the web.	No
IMPORT	Reads external data into a SAS data set.	No
JAVAINFO	Shows information about the version of Java on your system.	No
JSON	Reads data from a SAS data set and writes it to an external file in JSON representation.	No
LUA	Enables you to run statements from the Lua programming language within SAS code.	Yes
MAPIMPORT	Produces an output map data set, which can be used with the ODS Graphics SGMAP procedure or other mapping procedures in SAS.	No
MDSUMMARY	Computes basic descriptive statistics for variables across all observations or within groups of observations in parallel for data tables that are stored in SAS Cloud Analytic Services (CAS).	Yes
MEANS	Provides data summarization tools to compute descriptive statistics for variables across all observations and within groups of observations.	Yes, if the DATA= option specifies a data set with a CAS LIBNAME engine libref.
OPTIONS	Lists the current settings of SAS system options in the SAS log.	No
PRINT	Prints SAS data sets and CAS tables.	No. Provides data transfer if the DATA= option specifies a data set with a CAS LIBNAME engine libref.
PRINTTO	Redirects output.	No
PRODUCT_STATUS	Returns a list of the SAS Foundation products that are installed on your system, along with the version numbers of those products.	No
PWENCODE	Encodes passwords.	No
REGISTRY	Maintains the SAS registry.	No
REPORT	Combines features of the PRINT, MEANS, and TABULATE procedures with features of the DATA step in a single report-writing tool that can produce a variety of reports.	Yes, if the DATA= option specifies a data set with a CAS LIBNAME engine libref.
S3	Performs object management for objects in Amazon Simple Storage Service (Amazon S3).	No

Procedure	Description	Uses CAS Processing
SCOREACCEL	Provides an interface to the CAS server for DATA step and DS2 model publishing and scoring.	Yes
SGPANEL	Creates a panel of graph cells for the values of one or more classification variables.	No  Graphing large tables can cause performance degradations. You can summarize or reduce large amounts of data before attempting to graph it.  You can limit the number of rows to graph by using the OBSMAX= option to indicate the maximum number of rows to process.
SGPLOT	Creates statistical graphics such as histograms and regression plots, in addition to simple graphics such as scatter plots and line plots.	No  Graphing large tables can cause performance degradations. You can summarize or reduce large amounts of data before you attempt to graph it.  You can limit the number of rows to graph by using the OBSMAX= option to indicate the maximum number of rows to process.
SGRENDER	Produces graphical output from templates that are created with the Graph Template Language (GTL).	No  Graphing large tables can cause performance degradations. You can summarize or reduce large amounts of data before you attempt to graph it.  You can limit the number of rows to graph by using the OBSMAX= option to indicate the maximum number of rows to process.
SGSCATTER	Creates a paneled graph of scatter plots for multiple combinations of variables.	No  Graphing large tables can cause performance degradations. You can summarize or reduce large amounts of data before attempting to graph it.  You can limit the number of rows to graph by using the OBSMAX= option to indicate the maximum number of rows to process.
SORT	Sorts SAS data sets.	No
SQL	Runs SAS SQL.	No
STREAM	Enables you to process an input stream that consists of arbitrary text and that can contain SAS macro specifications.	No
SUMMARY	Provides data summarization tools that compute descriptive statistics for variables across all observations or within groups of observations.	Yes, if the DATA= option specifies a data set with a CAS LIBNAME engine libref.
TABULATE	Displays descriptive statistics in tabular format, using some or all of the variables in a data set.	Yes, if the DATA= option specifies a data set with a CAS LIBNAME engine libref.



Procedure	Description	Uses CAS Processing
TEMPLATE (see <a href="#">SAS Output Delivery System: Procedures Guide</a> )	Enables you to customize the appearance of your SAS output.	No
TRANPOSE	Creates a CAS table by restructuring the values in a CAS table and transposing selected columns into rows.	If the DATA= and OUT= options point to CAS, the transposition is performed within CAS by invoking the CAS transpose action.

## PROC CASUTIL

The CASUTIL procedure works with tables in SAS Cloud Analytic Services, SAS data sets in SAS libraries, and external files. Here is a list of the table functions and caslib management functions that are part of PROC CASUTIL:

- display table metadata
- delete files from a data source
- unload a table from a caslib
- lists tables in a caslib
- load tables to a caslib
- promote a table to global scope
- save a table in a caslib to a data source

### See Also

- [Use PROC CASUTIL to Load Data on page 15](#)
- [“CASUTIL Procedure” in SAS Cloud Analytic Services: User’s Guide](#)

## Programming with CAS Actions

### PROC CAS and the CAS Language (CASL)

PROC CAS enables you to interact with the CAS server from SAS by using the CAS language and CAS actions. It enables you to program with CAS actions. PROC CAS and CASL are similar to the DATA step and PROC IML because they share support for many of the same language elements. For example, you can use functions and statements such as the DO, GOTO, IF-THEN, and SELECT statements in the CAS procedure. PROC CAS provides built-in functions, or you can create your own functions.

This simple program loads the table Iris.sashdat to the active caslib (default) to perform a correlation:

```
proc cas;                                     /* 1 */
  table.loadtable / path="iris.sashdat" casOut={name="IRIS"};          /* 2 */
  simple.correlation result=x
    / table={groupBy={"Species"}, name="IRIS", orderBy={"SepalLength"}}; /* 3 */
```



```

data mycas.baseball;                                /* 3 */
  set sashelp.baseball;
run;

proc cas;                                           /* 4 */
  quantreg.quantreg/                               /* 5 */
    table={name='baseball'},
    class={'league','division'},
    model={depvars='Salary',
           effects={'nAtBat', 'nHits', 'nHome', 'nRuns',
                    'nRBI', 'nBB', 'yrMajor', 'crAtBat',
                    'crHits', 'crHome', 'crRuns', 'crRbi',
                    'crBB', 'league', 'division', 'nOuts',
                    'nAssts', 'nError'}}};
run;

```

- 1 Use the MySess session and the Casuser caslib. Other examples in this document use a separate CASLIB statement. You can specify caslibs in options in the CAS statement and in the LIBNAME statement.
- 2 The LIBNAME statement associates the CAS LIBNAME libref, Mycas, to the active caslib, Casuser.
- 3 Load the data set sashelp.baseball to the in-memory CAS table Baseball.
- 4 CAS action programming using CASL begins with the PROC CAS statement.
- 5 Use the quantreg action to fit a quantile regression model. When you refer to a table in a caslib within action code, you specify only the table name, not the libref. The class variables are League and Division, and the dependent variable is Salary.

To continue exploring this example and to view the example output, see the [quantreg action](#).

To begin learning about programming with CAS actions, you can review one of the following Getting Started documents:

- [Getting Started with CASL Programming](#)
- [Getting Started with SAS Viya for Python](#)
- [Getting Started with SAS Viya for Lua](#)
- [Getting Started with SAS Viya for R](#)

Document	Action Set Types
<a href="#">SAS Visual Data Mining and Machine Learning: Programming Guide</a>	Machine Learning
<a href="#">SAS Visual Data Mining and Machine Learning: Deep Learning Programming Guide</a>	Deep Learning
<a href="#">SAS Visual Text Analytics Programming Guide</a>	Text Mining
<a href="#">SAS Visual Statistics: Programming Guide</a>	Statistics

Document	Action Set Types
<a href="#">SAS Viya: System Programming Guide</a>	Access Control FCMP Language: DATA Step, DS2, FedSQL, CASL Server Properties Streaming Data Session Methods Session Properties System Table Transpose
<a href="#">SAS Visual Analytics Programming Guide</a>	Aggregation Decision Tree Hypergroup Data Preprocessing Analytics
<a href="#">SAS Econometrics 8.2: Programming Guide</a>	Regression models Dependency structure of multivariate distributions Linear econometric models Univariate limited dependent variable models Estimation of parameters of probability distributions
<a href="#">SAS Visual Forecasting 8.2: Programming Guide</a>	Reconcile forecasts Short-time Fourier transform of a time series Smoothing spectra Automatic time series model identification and forecasting Scripted time series analysis Time-stamped data accumulation for time series
<a href="#">SAS Optimization 8.2: Network Optimization Programming Guide</a>	Graph theory and network optimization algorithms Clique subgraphs to create a complete graph Connected graph components Cycle nodes Combinatorial optimization Sending flow over a network at minimal cost
<a href="#">SAS Optimization 8.2: Mathematical Optimization Programming Guide</a>	Solving linear problems General mixed integer linear programs (MILPS) Quadratic programs Solver techniques and algorithms

## DATA Step

The DATA step can run in your SAS session as well as in the CAS server. In CAS, the DATA step runs in a CAS server session on a single server (called symmetric multiprocessing, or SMP) or across multiple computers in parallel (called massively parallel processing, or MPP). The DATA step can run in multiple threads on each machine. When the DATA step runs in multiple threads, the same DATA step program is run in multiple threads on all nodes in the environment. By default, the DATA step runs in all available threads on every computer node in the cluster. When CAS in-memory tables are used exclusively, the DATA step takes advantage of the in-memory speeds. Some [language element restrictions](#) and [VARCHAR data type restrictions](#) apply when the DATA step runs in the CAS server.

To process data in CAS, you load the data to a [caslib](#).

The following example demonstrates how to use the DATA step to load data into CAS.

```
/*In the SAS Windowing Environment, you must specify the connection information*/
option casport=5570 cashost="cloud.example.com";
cas casauto;                                /* 1 */
libname mycas cas caslib=casuser;          /* 2 */
data mycas.cars (where=(weight>6000));      /* 3 */
    set sashelp.cars;
    keep make model type weight MPG_City;
run;
```

The following DATA step runs in the CAS server. When the input and output librefs use the CAS LIBNAME engine, the DATA step runs in the CAS server.

```
data mycas.cars2;                            /* 4 */
    set mycas.cars;
    if mpg_city > 25 then eff='Y';
    else eff='N';                            /* 5 */
    put 'Thread number: ' _threadid_         /* 6 */
        'on worker node ' _hostname_;
run;
```

- 1 Start a session with the CAS server.
- 2 Create the libref Mycas and bind it to the Casuser caslib. The CASLIB= option in the LIBNAME statement for the CAS engine binds the MyCas library to the Casuser caslib.
- 3 Load a subset of the sashelp.cars data set to the Casuser caslib.
- 4 Create the table cars2 in the Casuser caslib from the CAS table Cars. The SET statement runs in SAS. The SET statement loads a subset of the Sashelp.Cars data set (cars for which the weight is greater than 6000 pounds) to an in-memory table.
- 5 Specify whether the cars in the cars2 table run efficiently.
- 6 Write the thread and worker node where the data was processed.

See [“DATA Step Basics” in SAS Cloud Analytic Services: DATA Step Programming](#) for an overview of running the DATA step in CAS.

Not all DATA step language elements that you used in previous releases of SAS are appropriate for distributed processing in the CAS environment. For a list of language elements that run on the CAS server, see the category table in the respective documentation:

- [Hash objects](#) and [Java objects](#)

- [Data Set Options](#)
- [Formats](#)
- [Functions](#)
- [Statements: DATA step and global](#)

If the language element specifies a category of CAS in the documentation, it can run in the CAS server.

Some restrictions also apply when you use the DATA step in the CAS server. For a list of these restrictions, see the following topics:

- [“Restrictions for Language Elements in the DATA Step” in SAS Cloud Analytic Services: DATA Step Programming](#)
- [“Restrictions for the VARCHAR Data Type in the CAS Engine” in SAS Cloud Analytic Services: User’s Guide](#)

The BY statement in a SAS DATA step divides table data into groups of rows that share the same values for the BY variables. When you use the BY statement in a distributed server, CAS groups the rows based on the first BY variable. When CAS distributes the table across multiple nodes, it keeps the BY groups intact. That is, rows that share the same BY variables are stored together on a single machine. For information about BY processing behavior in CAS, see the following topics:

- [“How CAS Groups Data with BY Variables” in SAS Cloud Analytic Services: DATA Step Programming](#)
- [“How BY Variables Affect Multithreaded DATA Step Execution” in SAS Cloud Analytic Services: DATA Step Programming](#)

**Note:** You can also submit DATA step code by using the CAS action [dataStep.runCode](#) in PROC CAS.

[Video: Using the DATA Step in SAS Viya](#)

[Video: Processing Data in Groups with the DATA Step in SAS Viya](#)

---

## SAS Viya File Service

The SAS Viya file service enables you to store, retrieve, and delete files maintained in the file service repository. The repository is not considered a complete 'file system'. Rather, the repository contains individual files that are directly accessible by their file identifier. This file identifier contains a universally unique identifier (UUID) generated by the file service when a file is created.

The file service also assigns a unique name to each file in the repository, but it is not human-friendly name. A user can change the name, but there is a risk that the name might not be unique within the repository.

You can access a file in the file service using the file identifier. The file identifier is contained in the URI stored in the file information. Use the system-generated name or user-assigned name to find the file URI and the file identifier. Once you have the file identifier, you can then use it to access the file directly in the File Service.

The File Service does not have a concept of 'folders' in its repository. However, you can associate files using a `parentUri`. A `parentUri` is a relative URI for any object in SAS Viya. So you can create a collection of files by specifying the same `parentUri` for each one of those files.

This example creates the file `class.csv` and attaches the file to a job named in the `fileref` jobout.

```
filename jobout filesrvc
  parenturi='/jobExecution/jobs/5a308aa7-1c3a-4465-a14c-fd69a9091926';
data _null_;
  set sashelp.class;
  file jobout('class.csv');
  put name "," sex "," age "," height "," weight;
run;
```

For more information, see [“FILENAME Statement, FILESRVC Access Method” in SAS Global Statements: Reference](#).

---

## Macro Language

Macros are supported in SAS Viya, but only in the SAS session. Macros can be useful in generating code and submitting the code to the CAS server to run. However, the macro language itself does not run on the CAS server.

### See Also

- [SAS Macro Language: Reference](#)

---

## DS2

DS2 runs in SAS Viya as it has in previous releases of SAS. Using a FedSQL query within your DS2 program is not currently supported on the CAS server, with one exception. A FedSQL query is allowed in the SET statement in a DS2 action or PROC DS2.

In addition to using [PROC DS2](#) to run DS2 code, you can also use the [ds2.runDS2 action](#) by using [PROC CAS](#) or a supported third-party language. You can also specify FedSQL statements in the DS2 SET statement in Viya 3.3.

### See Also

- [SAS DS2 Programmer's Guide](#)
- [SAS DS2 Language Reference](#)

---

## FedSQL

In SAS Viya 3.3, FedSQL expands support to include SPD Engine data sets and external data sources that are supported in SAS 9.4 in addition to SAS data sets and CAS libraries. FedSQL supports the same functionality for the additional data sources in SAS Viya that these data sources have in SAS 9.4. The data sources are accessed by using librefs in PROC FEDSQL.

FedSQL supports CAS processing through caslibs. FedSQL supports all SAS Viya data connectors and data connector accelerators. The caslibs enable you to explicitly or automatically load SAS data and external data into CAS for processing. Beginning with SAS Viya 3.3, FedSQL supports implicit pass-through from CAS for the following data sources:

- Amazon Redshift
- DB2 (UNIX)
- Apache Hadoop (Hive and Impala)
- Open Database Connectivity (ODBC)
- Oracle

- PostgreSQL
- SAP Hana
- Teradata (UNIX)

In the initial release, FedSQL supports single-source, full-query SQL pass-through. To be eligible for implicit pass-through, the tables in the FedSQL request cannot have been previously loaded into the CAS session.

FedSQL also provides the following CAS enhancements in SAS Viya 3.3:

- Substantial improvements in CAS query performance.
- The ability to specify control parameters for the FedSQL query planner in CAS. The new CNTL= option enables you to disable implicit pass-through, to require implicit pass-through, and to join tables in the specified order instead of an order chosen by the FedSQL query optimizer.
- Support for new CAS data types INT64 and INT32.

You can access caslibs by using PROC FEDSQL or by using the fedSql.execDirect action. You can submit the fedSql.execDirect action via PROC CAS or by using a supported third-party language.

FedSQL supports a subset of its SAS 9.4 functionality in CAS. FedSQL provides query functionality only in CAS.

In CAS, FedSQL supports these statements:

- CREATE TABLE, with the AS expression
- SELECT (The SELECT operations have some limitations.)
- DROP TABLE

For detailed information about the FedSQL functionality that is supported in CAS libraries, see [SAS Viya: FedSQL Programming for SAS Cloud Analytic Services](#). For information about FedSQL functionality that is supported for SAS libraries, see [SAS FedSQL Language Reference](#).

## See Also

- “CAS Procedure” in [SAS Cloud Analytic Services: CASL Reference](#)
- “FEDSQL Procedure” in [Base SAS Procedures Guide](#)
- “FedSQL Action Set” in [SAS Viya: System Programming Guide](#)
- “Dynamically Executing FedSQL Statements from DS2” in [SAS DS2 Programmer’s Guide](#)

---

## User-Defined Formats

You can store user-defined formats in SAS Viya in catalogs to use in a SAS session, or you can store them in a format library on the CAS server. Format libraries are associated with a CAS session, or they can be promoted to global scope to be available to all CAS sessions. User-defined formats in a format library are server-side formats that the server uses when an analysis is performed according to formatted values. You can migrate existing user-defined formats from SAS to Viya. For more information, see [SAS Cloud Analytic Services: User-Defined Formats](#).

You still use [PROC FORMAT](#) to create formats. A new option, [CASFMTLIB=](#), names the CAS format library for the caslib where the format is stored. Without the CASFMTLIB= option, formats continue to be stored in a format catalog. [PROC FMT2ITM](#) converts one or more format catalogs into a single item store that can be used as input to the addFmtLib action, which adds a CAS format library.



It is a best practice to assign a format to a variable before the table is loaded into the server. After a table is in memory, some procedures cannot assign a format. You can assign a format with the **FORMAT** statement in a **DATA** step or with the **CASUTIL** procedure.

This example adds and saves a user-defined format:

```
/*In the SAS Windowing Environment, you must specify the connection information*/
options casport=5570 cashost="cloud.example.com";
cas casauto sessopts=(caslib="casuser");
proc format library=work.formats casfmtlib="casformats";           /* 1 */
  value enginesize
    low - <2.7 = "Very economical"
    2.7 - <4.1 = "Small"
    4.1 - <5.5 = "Medium"
    5.5 - <6.9 = "Large"
    6.9 - high = "Very large"
;

cas casauto savefmtlib fmtlibname=casformats                       /* 2 */
  table=enginefmt replace;

proc casutil;
  format enginesize enginesize.;                                   /* 3 */
  load data=sashelp.cars casout="cars" replace;
  contents casdata="cars";
quit;
libname mycas cas;                                                /* 4 */

proc mdsummary data=mycas.cars;
  var mpg_highway;
  groupby enginesize / out=mycas.mpg_hwy_by_size;                 /* 5 */
run;

proc print data=mycas.mpg_hwy_by_size;
  var enginesize--_mean_;
run;
```

- 1 The **FORMAT** procedure creates a format that is named **Enginesize**. On the SAS client, the format is temporary and is stored in the Work library. The **CASFMTLIB=** option adds the same format to your CAS session in a format library that is named **Casformats**.
- 2 The CAS statement with the **SAVEFMTLIB** option persists the format library with member **Enginesize** as a **SASHDAT** file in the data source that is associated with the active caslib. The **Casuser** caslib is typically a **PATH** type and uses a file system. The **TABLE=** option specifies the name and results in a file that is named **enginefmt.sashdat**. This enables you to load a format library from a table in future sessions.
- 3 The **FORMAT** statement with the **CASUTIL** procedure assigns the **Enginesize** format to the **Enginesize** variable. The format is applied to the in-memory instance of the **Cars** table.
- 4 A CAS **LIBNAME** engine libref is assigned so that SAS procedures can access the **Cars** table that is in memory on the server. The assignment does not specify a caslib, so the active caslib is used.
- 5 The **MDSUMMARY** procedure provides descriptive statistics for the **Cars** table. The results are grouped by the formatted values of the **Enginesize** variable (five values) instead of by the numeric value. As a result, the output table, **Mpg\_hwy\_by\_size**, has five rows.

[Video: Creating User-Defined Formats with the FORMAT Procedure in SAS Viya](#)

## See Also

- [SAS Cloud Analytic Services: User-Defined Formats](#)