SAS® Cloud Analytic Services 3.5: Fundamentals

Introduction **2**
- What Is SAS Cloud Analytic Services? 2
- About Memory Management 2

Architecture **3**
- Distributed Server 3
- Distributed Server: Controller Fault Tolerance 3
- Single-Machine Server 5

Data **6**
- Data Lifecycle 6
- Client-Side Data Access 7
- Server-Side Data Access 8
- Caslibs, Files, and Tables 10
- More about Tables 10
- Rules for Caslib, Table, and Column Names 14

Memory **15**
- How SAS Cloud Analytic Services Uses Memory 15
- Managing and Monitoring Overview 16
- Special Cases 18

Sessions **19**
- About SAS Cloud Analytic Services Sessions 19
- Why Sessions Are Used 20
- Session Properties 20

Security **21**
- Authentication 21
- Authorization 21
- Encryption for Data at Rest 21
- Encryption for Data in Motion 22

Caslibs **22**
- What is a Caslib? 22

Caslib Scope 23
Personal, Predefined, and Manually Added Caslibs 23
Who Can Add, Delete, and Access a Global Caslib? 24
The Active Caslib 24
Path-Based Caslibs 24

Introduction

What Is SAS Cloud Analytic Services?

- SAS Cloud Analytic Services is a server that is suitable for both on-premises and cloud deployments. The server provides the run-time environment for data management and analytics. By run-time environment, we refer to the combination of hardware and software where data management and analytics take place.
- The server can run on a single machine or as a distributed server on multiple machines. The distributed server consists of one controller, an optional backup controller, and one or more workers. This architecture is often referred to as a massively parallel processing architecture. For both architectures, the server is multi-threaded for high-performance analytics.
- The distributed server has a communication layer that supports fault tolerance. A distributed server can continue processing 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.
- SAS Studio provides a programming environment for developing and submitting SAS programs to the server.
- The SAS Scripting Wrapper for Analytics Transfer (SWAT) enables open-source software such as Python, Lua, and R to run data analysis on the server. For Java, classes are provided to enable connections to the server and classes are provided to run data analysis.

About Memory Management

One of the design principles of the server is to handle large problems and to fully exploit the scalability of modern cluster computing hardware. In order to address this principle, data in the server is managed in blocks.

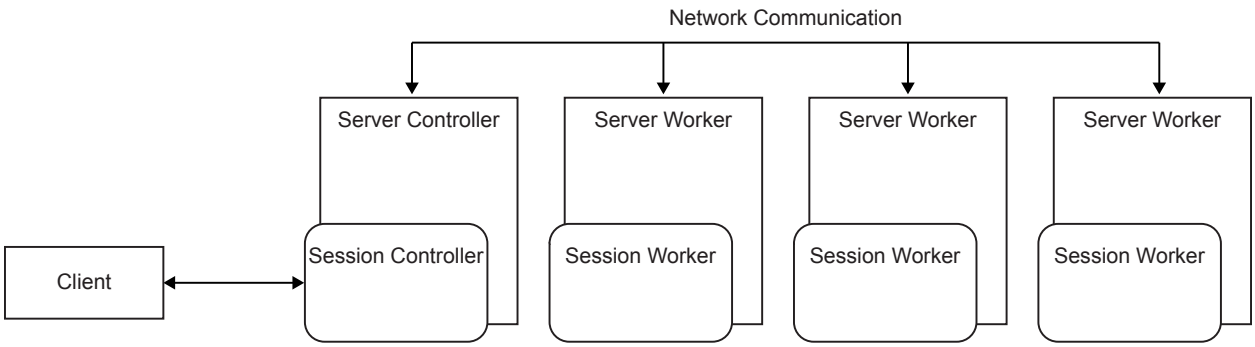
Whenever needed, the server caches the blocks on disk. It is this feature that enables the server to manage memory efficiently, handle large data volumes, and remain responsive to requests. More information is provided in the sections about data access and memory.

Architecture

Distributed Server

A distributed server uses multiple machines to perform massively parallel processing (MPP). A distributed server supports the Linux operating system only. The following figure depicts the server topology for a distributed server:

Figure 1 *Distributed Server Architecture*



- One machine is designated as the controller. Client applications communicate with the controller and the controller coordinates the processing that is performed by the worker nodes.
- One or more machines are designated as worker nodes. Each worker node performs data analysis on the rows of data that are in-memory on the node.
- The server scales horizontally. If processing times are unacceptably long due to large data volumes, more machines can be added as workers to distribute the workload.
- Distributed servers are fault tolerant. If communication with a worker node is lost, a surviving worker node uses a redundant copy of the data to complete the data analysis.
- Whenever possible, distributed servers load data into memory in parallel. This provides the fastest load times.
- For optimal performance it is recommended that all CAS Workers to be identically sized.

Distributed Server: Controller Fault Tolerance

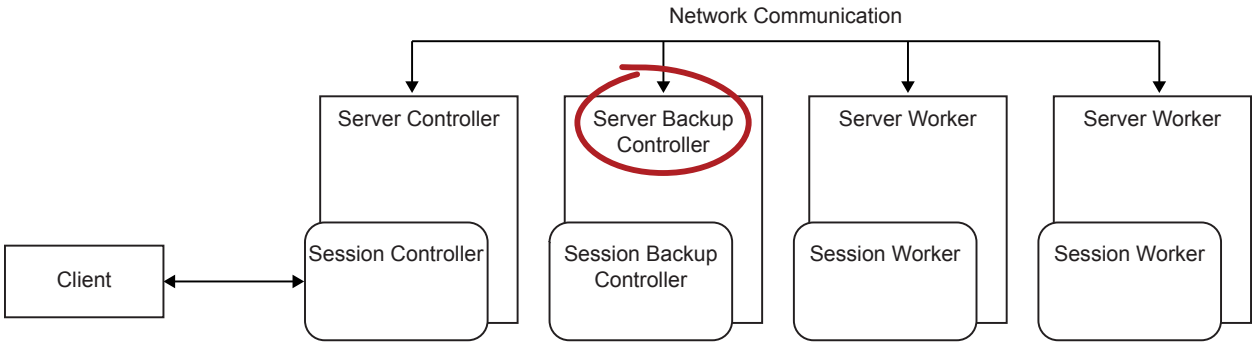
Overview

Support for a backup controller is added in the SAS Viya 3.3 release.

- Controller failover provides continuity of the service for the server and its global state when the primary controller fails. New connections to the server can be made through the backup controller. In some cases, clients can failover transparently—without any indication that connection has changed over to the backup controller.
- Global-scope in-memory tables remain loaded and are unaffected by failover. However, for data that is not in-memory at the time of failover, see “Data Availability Considerations”.
- A backup controller is an optional feature. A backup controller applies to distributed servers only. You cannot add a backup controller to a running server.
- One backup controller is supported. When the primary controller fails, the site operates without fault tolerance for the controller until a planned outage.

The following figure depicts a distributed server with a backup controller:

Figure 2 *Distributed Server with a Backup Controller*



How Failover is Detected

- The primary controller synchronizes with the backup controller continuously. The updates enable the backup controller to provide service rapidly.
- The primary controller sends *heartbeat* messages to the backup controller and workers. A failover occurs when the backup controller does not receive a heartbeat message from the primary controller within the lost heartbeat time-out interval.

For more information, see env.CAS_HEARTBEAT_LOST_TIMEOUT in SAS Cloud Analytic Services: Reference.

After Failover (Binary Protocol)

Most SAS Viya applications use the binary protocol. These applications look up the controller host name from Consul. Servers with a backup controller register themselves in Consul when they start and unregister when they stop. Most significantly, when the controller fails over, the backup controller updates the controller host address in Consul with its own host name. As a result, SAS Viya applications connect to the controller at the new address. For a few minutes during the failover, applications that are running CAS actions can become unresponsive. Connected users might need to discard their sessions (such as closing a browser) and connect again.

Programming clients such SAS, Python, Java, and so on, can use binary protocol too. However, these clients do not use Consul to look up the connection information. These clients connect to the server by specifying the host name and network port of the CAS controller. These clients can accept

both the primary controller and backup controller host names at connection time. Specifying both enables the client to switch automatically from the primary to the backup.

Be aware that the automatic connection to the backup controller for the programming clients requires that the network return an error. When the client attempts to connect to the failed primary controller, the network error is the trigger to connect to the backup controller. However, in some network topologies, when a host is lost, network communication can cause a long delay while a network time-out expires. The duration of the delay is controlled by the network software of the client operating system. In some cases, the delay can be up to 15 minutes. In these cases, the client is unresponsive until the time-out expires.

After Failover (REST and HTTP Protocol)

The REST interface provided by the server and CAS Server Monitor rely entirely on the Apache HTTP proxy to redirect clients from the failed controller to the backup controller. In a full deployment, the HTTP proxy is used and monitors the controller address that is registered in Consul. After failover, the backup controller updates the host name in Consul and HTTP requests are automatically directed to the backup controller.

During the failover, REST clients and CAS Server Monitor are likely to experience errors or unresponsiveness while the network communication software of the client operating system times out. After the failover is complete and the proxy has switched over, connections that use REST and HTTP can resume.

Programming-only Deployment Limitations

SAS offers a programming-only deployment of SAS Viya. This deployment is smaller and simpler. Consul is included in programming-only deployments, but the HTTP proxy is not dynamic. Using a backup controller is not recommended in programming-only deployments if the programming clients use the REST interface or if CAS Server Monitor is used for administration.

For information about a programming-only deployment, see *SAS Viya Administration: Getting Started*.

Data Availability Considerations

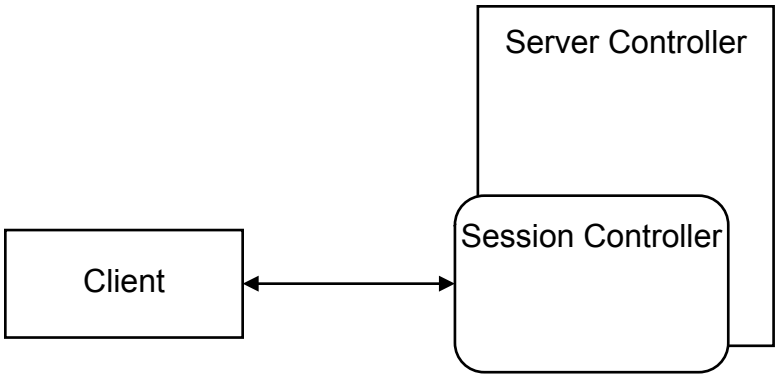
- For caslibs that use PATH or DNFS as the data source, the directory must be a network location that is available at the same path on both controller hosts.
- For server-based caslibs that use database drivers, the drivers must by available on both controller hosts.
- If user-defined format libraries are saved to path-based caslibs, then the directory must be a network location that is available at the same path on both controller hosts.

For more information, see *SAS Viya Administration: SAS Cloud Analytic Services*.

Single-Machine Server

A single-machine server supports Microsoft Windows and Linux operating systems. Support for Microsoft Windows is added in the SAS Viya 3.4 release. The following figure depicts the server topology for a single-machine server:

Figure 3 *Single-Machine Server Architecture*



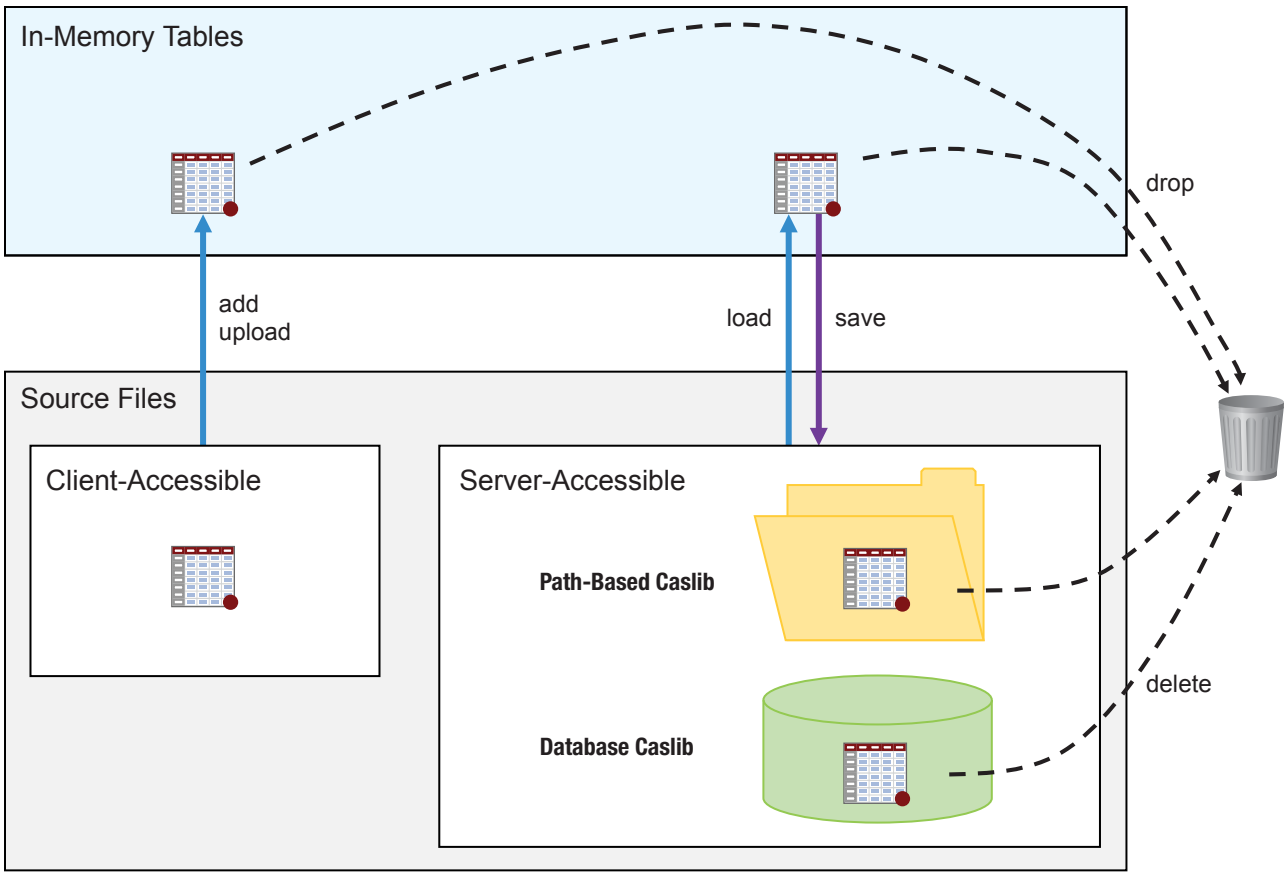
- The single machine is designated as the controller. Because there are no worker nodes, the controller node performs data analysis on the rows of data that are in-memory.
- The single machine uses multiple CPUs and threads to speed up data analysis. This architecture is often referred to as symmetric multi-processing (SMP).
- All the in-memory analytic features of a distributed server are available to the single-machine server.
- Single-machine servers cannot load data into memory in parallel from any data source.

Data

Data Lifecycle

The following graphic summarizes fundamental concepts of the data life cycle for SAS Cloud Analytic Services:

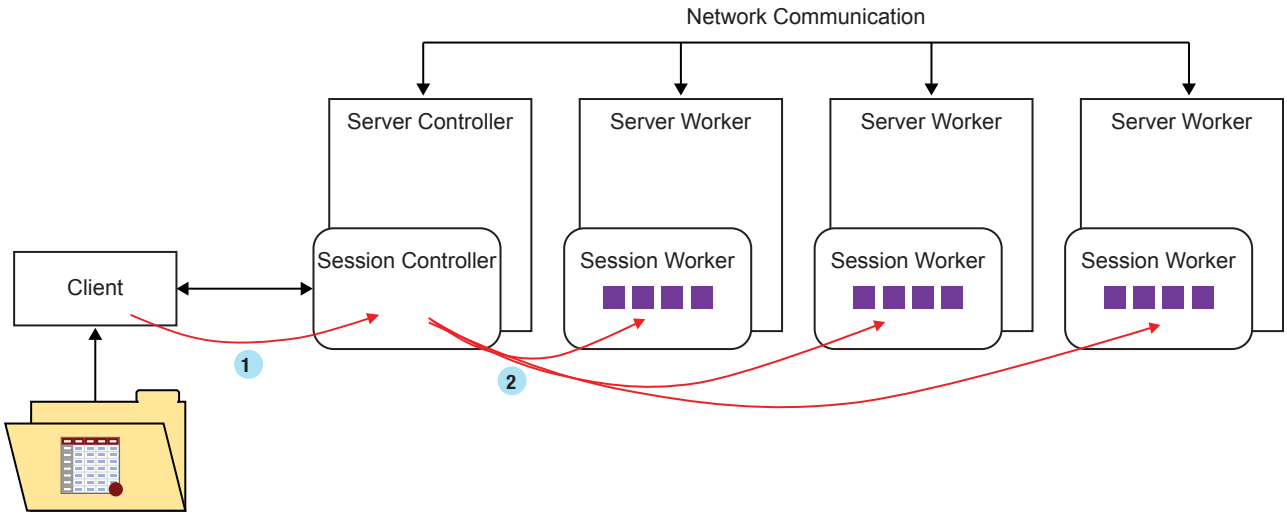
- SAS Cloud Analytic Services operates on in-memory tables. Clients can add or upload data. The server can load data from server-side data sources.
- The server can save and load data from server-side data sources.
- When an in-memory table is dropped, it does not affect a persisted file. Persisted files are removed from the data source by deleting them.



Client-Side Data Access

The following figure depicts a client application, such as SAS Studio, transferring data to a distributed server. The following SAS language elements operate this way:

- A DATA step with the CAS LIBNAME engine
- The LOAD DATA= form of the CASUTIL procedure



1 The client reads the file and transfers the data serially, in chunks, to the session controller.

2 The session controller receives the data from the client and distributes rows to the worker nodes. The rows are distributed in round-robin fashion to the workers.

Table 1 *Advantages and Disadvantages of Client-Side Data Loading*

Advantages	Disadvantages
This is a very straightforward way to load data into the server.	■ Large files can require a long time to transfer between the client and the server. ■ After the data is in memory, you should save it to a caslib's data source.

Server-Side Data Access

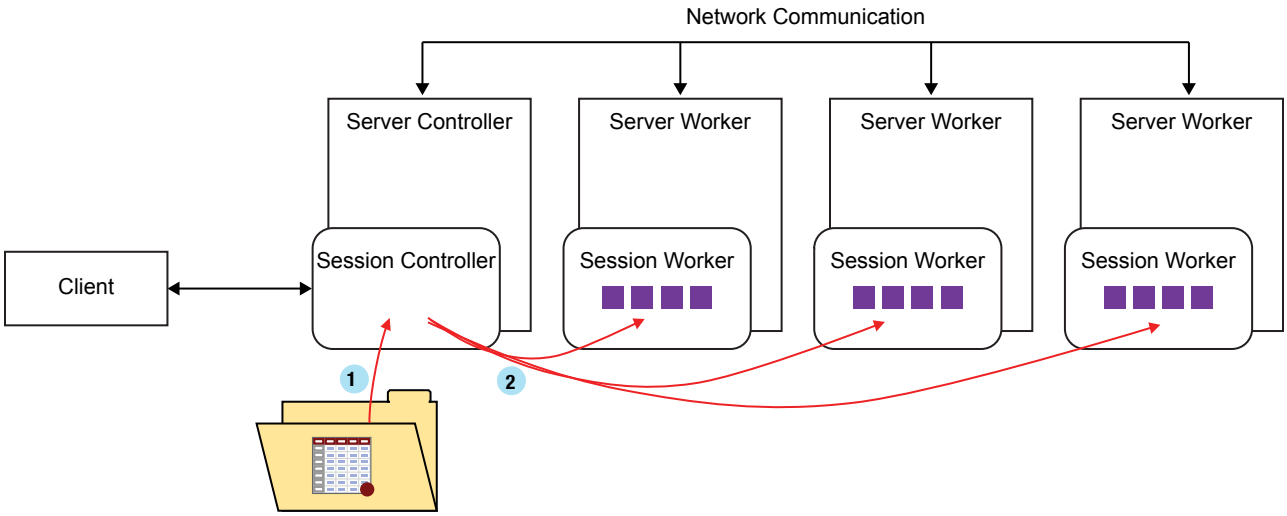
Serial Data I/O

A key feature of SAS Cloud Analytic Services and caslibs is that a caslib has an association with a data source. This is such a valuable feature to the server that caslibs associated with paths and caslibs associated with server-based data sources support server-side loading of data.

The LOAD CASDATA= form of the CASUTIL procedure is used to perform a server-side data load.

The following figure depicts how a server loads a SAS data set (or any file) that is accessible to the controller node only.

Figure 4 *Server-Side Load of a Data File*



1 Because the file is accessible to the controller node only, the controller reads the file from the caslib's data source.

2 The controller node distributes rows to the worker nodes.

Parallel Data I/O

The following figure also depicts a server-side data load. This representation shows parallel data access between the data source and the worker nodes. This pattern applies to all caslib data sources that make the data available to the worker nodes. These data sources are as follows:

- Some path-based data sources:
  - Distributed NFS (DNFS)
  - Hadoop Distributed File System (HDFS)
- Data sources that support a SAS Data Connect Accelerator and SAS Embedded Process is installed.

Figure 5 *Server-Side Parallel Load*

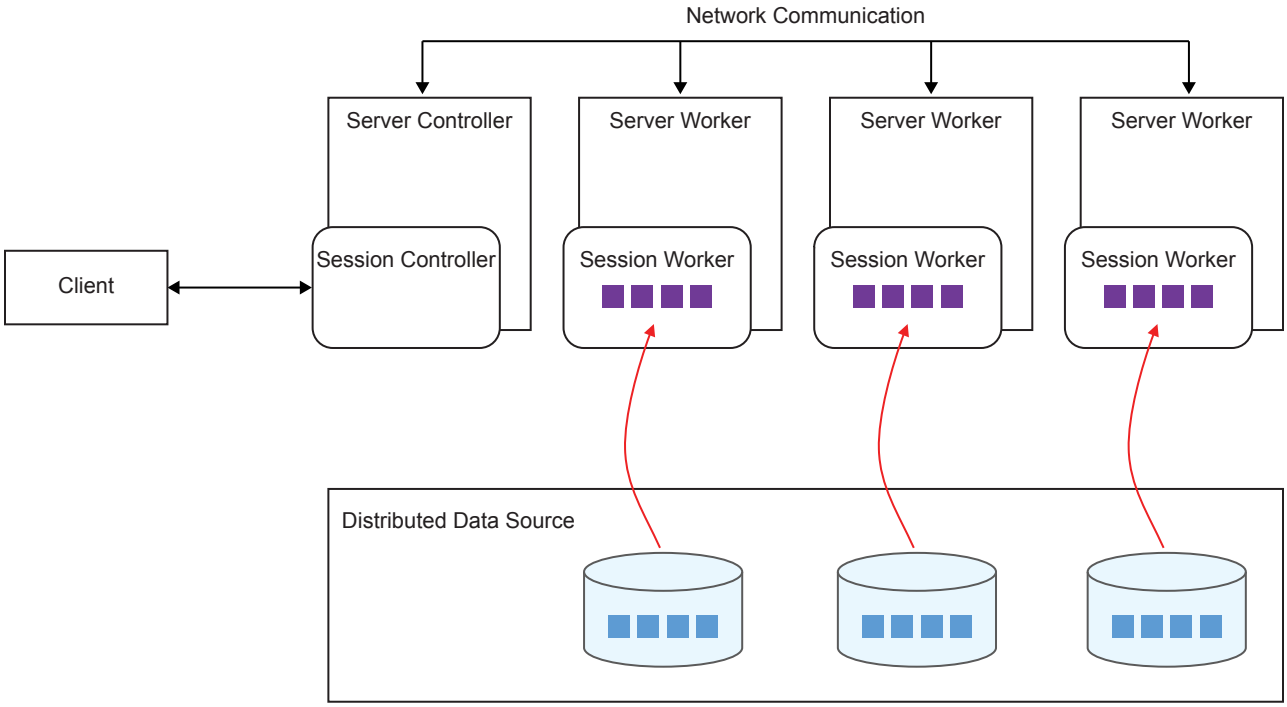


Table 2 *Advantages and Disadvantages of Server-Side Data Loading*

Advantages	Disadvantages
■ Large files and small files are loaded into memory as fast as possible. ■ Files can be loaded into memory after a server restart (or a table is dropped) very quickly.	■ You might not be sure which files are available in the caslib's data source. You can use the LIST FILES statement with the CASUTIL procedure to see which files can be loaded into memory.

Caslibs, Files, and Tables

All access to data with SAS Cloud Analytic Services is through a caslib. At its simplest, a caslib has the following properties:

- A caslib is associated with a data source and includes the connection information for the data source. For example, the data source can be a directory or the host, port, and other connection information for an Oracle database.
- The data in the associated data source is referred to as a file. For path-based caslibs, these files are SASHDAT files, CSV files, SAS data sets, and so on. For server-based caslibs, such as an Oracle database, the term file is still used to create a distinction between data from the caslib's data source, and an in-memory copy of the data.
- SAS Cloud Analytic Services performs analysis on in-memory tables only. In addition to providing an association with a data source, a caslib provides access to in-memory tables that have been loaded into memory.

  Throughout product documentation, the following terms are used interchangeably:

  - data table
  - in-memory table
  - table

Caslibs also provide access control to data. For more information, see *SAS® Viya Administration: CAS Authorization*.

More about Tables

Summary of Table Scope

Scope	In-Memory Duration	Availability	Replace
Global	Long (never automatically dropped)	All sessions[1]	Drop and then load a replacement.
Session	Medium (dropped when the session ends)	Current session	Overwrite with the replacement.
Transient	Short (dropped when the action completes)	Current action	Not applicable.

1 Subject to access controls.

Session and Global Scope

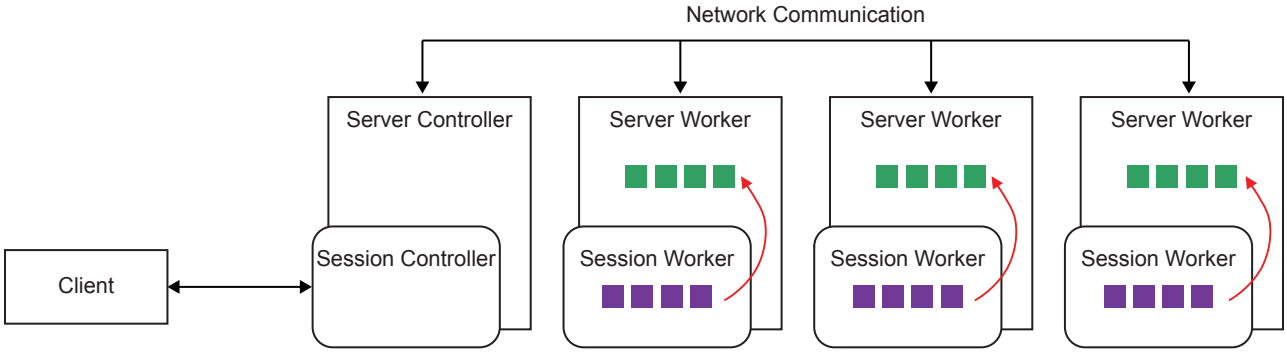
By default, when you load a table into memory, the table has *session scope*. This means that the table is available to that session only. For ad hoc data access and analysis, session-scope tables are preferred because session-scope tables do not require access control checks or any form of locking for concurrent access.

The only disadvantage to a session-scope table is that no other sessions can access the same table. For example, if you want shared access to a single copy of an in-memory table, then a session-scope table does not work. In that case, a global-scope table can provide the shared access.

Table 3 *Best Use and Considerations*

Global-scope tables	■ Best used for tables that are accessed by a large number of users, especially the visual interfaces. ■ A global-scope table cannot be replaced. You must drop it and load the replacement data.
Session-scope tables	■ Best used for general purpose programming. ■ Typically provides better performance than global-scope tables because concurrency locks are not used.

The following figure depicts how the rows of a session-scope table (in purple) are *promoted* to global scope (in green). After the table is promoted, any sessions that have access to the same caslib can access the single copy of the in-memory table.



Transient Scope

A transient-scope table is a table that is loaded into memory only for the duration of an action and then automatically dropped. The table is loaded from the caslib's data source. This is a specialized use case that is available when programming with actions.

- Transient-scope tables are suitable for one-time uses such as scoring.
- A table with the same name must not already exist as a session-scope or global-scope in-memory table. If there is an existing table, then the existing table is used.

- Performance is typically worse than global or session-scope tables because the table is read from the caslib's data source before the action is run. For path-based caslibs, disk I/O is a factor. For server-based caslibs, data is transferred over the network before the action is run.
- Transient-scope tables apply to action programming only where a table is specified by name. For SAS programming, where a CAS LIBNAME engine libref is used, transient-scope tables are not valid.

Fault Tolerance: Data Redundancy

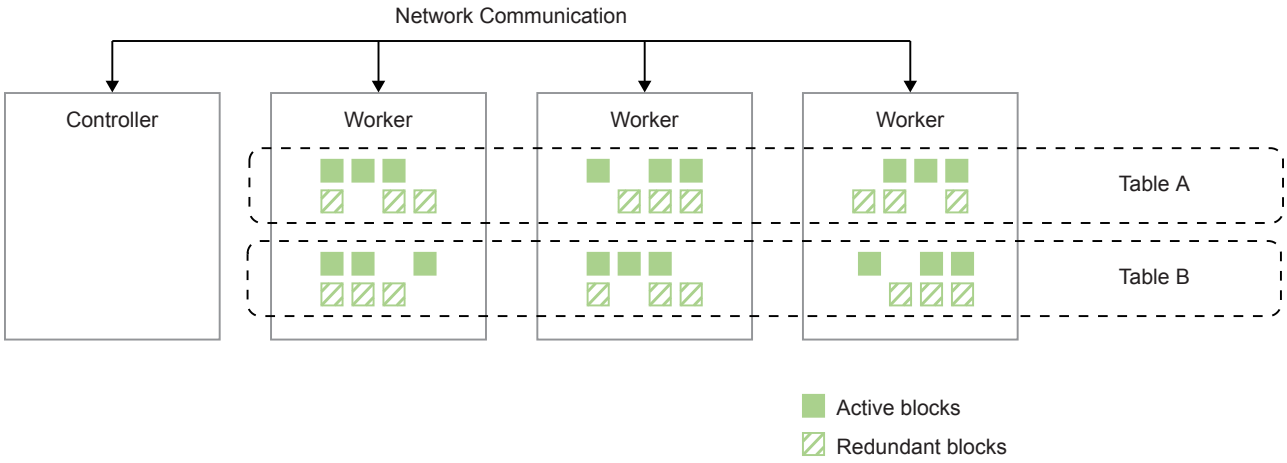
Fault tolerance applies to distributed servers only.

In order to offer fault tolerance, the server performs one of the following when data is loaded into memory:

- If the table was loaded from a SASHDAT file in HDFS, then fault tolerance is provided by the redundant blocks that were created when the file was saved to HDFS. In the event of node failure, a surviving node accesses the data from the redundant block.
- If the table was loaded from a SASHDAT file in a caslib with a source type of DNFS or PATH, then redundant blocks are not loaded preemptively into memory. In the event of a fault, the nodes that remain can access the source file and load copies of the blocks that were used by the failed nodes.
- For all other file and source data types, when you load the file into memory, you can specify the number of redundant blocks to create. In the event of node failure, a surviving node accesses the data from a redundant block.

> **TIP** The redundant copies of blocks are stored in the directories associated with the file system directories associated with the CAS_DISK_CACHE environment variable. Increasing the number of copies increases the amount of disk storage that is used. For more information, see "About the Disk Cache" on page 16.

The following figure depicts how the server uses the system of active blocks and redundant blocks to provide fault tolerance. There are three active blocks for Table A on the first worker node. The redundant blocks for those three blocks are distributed between the second and third worker nodes.



Partitioning and Ordering

By default, the order of rows in a table is not predictable. This is true for both single-machine servers and distributed servers.

One way to introduce order is to partition a table by one or more columns and then specify one or more different columns to use for ordering the rows. When you partition a table, all the formatted values for the partitioning columns are kept in a single partition. For a distributed server, the partition is on a single machine. For a single-machine server, all the partitions are on the single machine.

If you use BY-group processing in a DATA step with the same variables, then it is a performance advantage to partition the table when you initially load the table into memory. Otherwise, the BY groups are formed each time the DATA step is run.

Similarly, if you use a procedure that supports a GROUPBY statement and you specify the partitioning variables, then it is a performance advantage.

You can append records from in-memory tables into another in-memory table. The tables must be partitioned by the same columns. The target table can use partitioning only and not partitioning and ordering.

Indexing

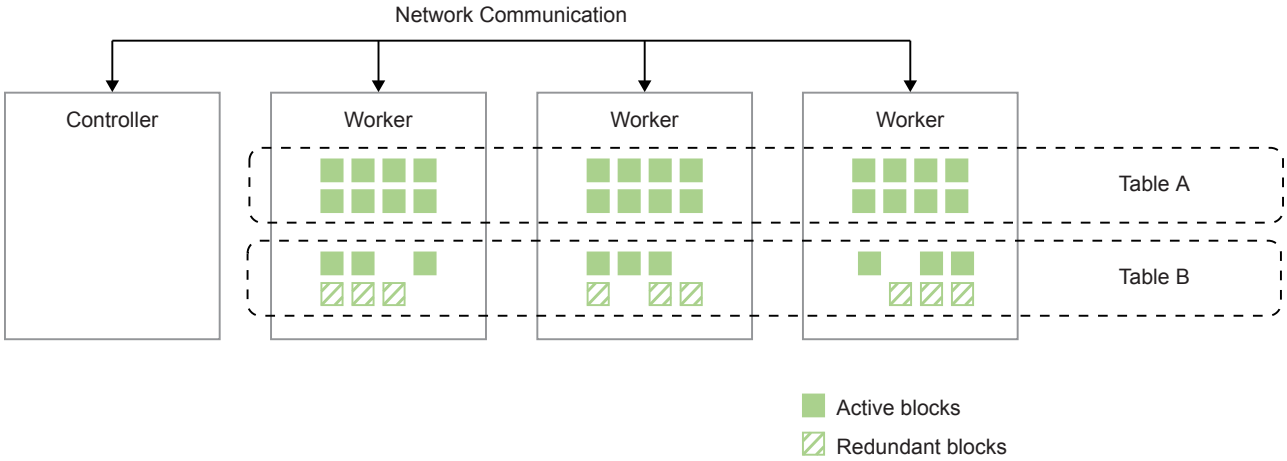
Column indexes are introduced with the SAS Viya 3.3 release.

- The server supports indexes on numeric (DOUBLE), CHAR, and VARCHAR columns.
- Indexes are supported on single-machine and distributed servers.
- Indexes can be used with other table features such as compression, encryption, fault tolerance, and session-scope and global-scope tables.
- When you subset rows with a filter and specify at least one column that is indexed, the index is used to improve performance. The server identifies the indexed column with the highest cardinality and excludes data that can never satisfy the filter criteria.
- When you update values in rows or append rows, the indexes are updated. The index is part of the in-memory table and not a separate object or file. There is no need to re-index an in-memory table.
- When you save an indexed in-memory table to a caslib that uses PATH, DNFS, or HDFS, the index is included in the saved SASHDAT file. The next time you load the table from the file, the index is ready for use.

For more information, see the details in “Indexing Tables” in *SAS Viya: System Programming Guide*.

Repeated Tables

The following figure depicts a repeated table, Table A, and a distributed table, Table B. A repeated table has all of the rows in blocks that are identical on all worker nodes of a distributed server. Put another way, the table is loaded into memory on all the machines.



Repeated tables are useful in some operations, such as table joins, where the rows of a dimension table need to be matched against the keys in the fact table. If all the rows of the dimension table are repeated on each worker node, then the join can be completed without exchanging rows between the worker nodes. Repeated tables are not managed for fault tolerance because each node has all the rows.

Rules for Caslib, Table, and Column Names

Caslib Names

- Names can be 1 to 256 characters long. National and multi-byte characters are accepted.
- Names cannot include the slash character (/).
- Paths that include symbolic links (hard or soft) are not supported.
- Some special characters, such as backslash (\) and semicolon can be specified in a programming interface. However, many SAS Visual Analytics visual interfaces and REST clients do not accept special characters. To ensure that a caslib can be used by the greatest number of interfaces, avoid special characters.
- The following prefixes are reserved.
  - Names cannot begin with CASUSER (in uppercase, lowercase, or mixed case). This is a reserved prefix to prevent confusion with the CASUSER and CASUSERHDFS personal caslibs.
  - Names cannot begin with an underscore character (_). This is a reserved prefix for caslibs that are used for the server.
  - Names cannot begin with #LOCAL_. This is a reserved prefix that is used by the server for automatically named session-scoped caslibs that are used by the server.

Table Names

- Names can include national characters and have no limitation on length. However, most operating systems have limits on filename lengths if you need to save an in-memory table as a file. Similarly,

most databases have limits on table name lengths and can deny saving an in-memory table with a long name as a database table.

- The server does not stipulate limits on lengths. However, the CAS LIBNAME engine, SAS, and most third-party clients have limits. If you intend to access the tables with the CAS LIBNAME engine or SAS, see "Words and Names" in *SAS Programmer's Guide: Essentials*.
- In-memory table names are converted to uppercase. Searches for in-memory table names are case insensitive.
- If the table is loaded from a caslib that enables access to subdirectories, slashes are replaced with periods (.). However, searches for table names do not convert slashes to the period character.
- When you load data from files, the file extensions are stripped by default.
- When saving a table to a path-based caslib that enables access to subdirectories, you can specify a slash in the filename to save the file in a subdirectory.
- UNIX filenames use the character encoding that is based on the server's locale when it is started. A mix of encodings can be an issue when referencing SAS data sets in a shared file system. This is common when data sets are created by SAS in one encoding and referenced by a CAS server that is running a different encoding. For example, there is no way to reference a pathname that was created with the EUC-CN encoding unless all paths from that CAS server are also EUC-CN. The server sets the encoding according to the LANG environment variable at server start-up.
- When the server performs a transient load, the server does not assign a meaningful name to the transient in-memory table. In this case, a name that begins with a _T_ prefix is generated. The rest of the name resembles a UUID and provides uniqueness. _T_ is not a reserved prefix, but avoid naming tables with the prefix to prevent confusion between server-generated names and user-supplied table names.

Column Names

- Names can include national characters and have no limitation on length. However, most databases have limits on column name lengths and can deny saving an in-memory table with a long name as a database table.
- The server does not stipulate limits on lengths. However, the CAS LIBNAME engine, SAS, and most third-party clients have limits. If you intend to access the tables with the CAS LIBNAME engine or SAS, see "Words and Names" in *SAS Programmer's Guide: Essentials*.
- Searches for column names are case insensitive. A table cannot have one column that is named colA and attempt to add a column that is named ColA.

Memory

How SAS Cloud Analytic Services Uses Memory

SAS Cloud Analytic Services (CAS) is an in-memory server and it analyzes in-memory tables. The primary concern for memory use is for in-memory tables. The goal of the server is to use memory

efficiently and provide the best performance for the amount of physical memory available and the data volume to analyze.

To meet the goal, the server uses file-based memory mapping. For SASHDAT files, because the file is already on disk, the server memory maps the file. For other file types and data sources, the server uses the directories associated with the CAS_DISK_CACHE environment variable to store blocks temporarily in files.

Some of the benefits of memory mapping are as follows:

avoid paging to system swap space
: System swap space is small compared to the overall disk space for the host. The swap space is used when memory demand is high. The cost to page out data is high because performance is limited to the write speed of disk drives.

  With memory mapping, the host can write in-memory blocks to the cache during idle time and avoid poor performance when free memory is extremely low. The read speed for disk drives is high, so the cost to read memory mapped blocks is low.

data that exceeds physical memory capacity
: By memory-mapping blocks, the server is able to analyze data that is larger than physical memory capacity. This applies to both single large tables and when the combined size of many tables exceeds memory capacity. Blocks are read from the memory until the physical memory limit is met. Then, because the blocks are memory mapped, some physical memory can be freed (without the performance penalty associated with paging out) and blocks are paged in from the next series of memory-mapped files.

memory efficiency
: For global-scope tables and all tables loaded from SASHDAT files, the use of memory mapping enables multiple sessions to share the same physical memory. If many sessions access the same global-scope table or SASHDAT-backed table, only one instance of the data is in physical memory.

Managing and Monitoring Overview

About the Disk Cache

SAS Cloud Analytic Services organizes data from tables in blocks. With the exception of SASHDAT files and specialized cases, a copy of in-memory blocks are temporarily stored in file system directories. When the server is installed, one or more of these directories are specified for the CAS_DISK_CACHE environment variable.

The disk cache affects performance in two ways. First, the disk space must be sufficient to store the blocks. For distributed servers, keep in mind that additional copies of blocks are stored to provide data redundancy. Secondly, the number of disks is a factor for speed. The server uses the specified directories in a round-robin fashion. If the directories correspond to different physical disks, then contention is reduced and performance is better.

The best practice is to set the directories during deployment. For tuning advice and sample scenarios, see Set Up the CAS Cache Directory in *SAS Viya for Linux: Deployment Guide*.

Tools for Monitoring Memory Use

The builtins.getCacheInfo action provides information about the directory names used for the cache, the capacity, and the free space that is available. This action is supported for all platforms. You must assume a Superuser role to run the action.

For Linux servers, the stand-alone grid monitor shows memory use for the server process and for each session. The metrics include the virtual memory size, the size blocks in the CAS_DISK_CACHE, and the size of blocks that are read from SASHDAT files. You can drill down to view the metrics by host (listed as "rank").

The CAS Server Monitor shows the memory use for the server process, session processes, and the host. You can view per-session resident memory size (physical memory). The CAS Server Monitor applies to Linux and Microsoft Windows. Be aware that some deployments do not enable CAS Server Monitor if SAS Environment Manager is available.

The table.tableDetails action can list the number of mapped blocks, unmapped blocks (backup blocks), and blocks that are allocated from physical memory only (not backed by a file). This information is available at the table level for Linux and Microsoft Windows servers. The action is used by the CASUTIL procedure, so some of the functionality is available with the procedure.

How to Limit Memory Use

Limiting the Address Space Is Not the Best Choice

To enforce memory use limits, an administrator for a Linux server can set an address space ulimit for a user or group. However, this is not universally recommended because ulimit is a single-process control. Memory is used by the server process and each of the many session process that users start. Because each user can start more than one session, the single-process accounting is unlikely to provide the control that is needed.

Finally, the address space ulimit does not distinguish between virtual memory and the memory efficiencies of sharing physical memory. The shared memory size is included in the address space for each session and that does not help manage physical memory.

Linux Cgroup

A Linux cgroup can enforce a physical memory limit for all the session processes that are started on that server. To enable the cgroup memory limit, an administrator starts the server with the cas.memorysize configuration file option. The minimum limit is 3 GB. For distributed servers, a cgroup with the specified limit is created on each host.

The cgroup is used to limit the physical memory use and not the address space use. As a result, users can still operate on tables that exceed the cgroup limit because only a portion of the table is paged in to physical memory at any time. The cgroup enables administrators to manage physical memory utilization instead of virtual memory. This is important because physical memory is the scarce resource and the subject of management and tuning.

Because most tables are memory mapped from SASHDAT files or CAS_DISK_CACHE (the strategy for short-lived output tables is an exception), the server does not use any system swap space. CAS configures the cgroup to prevent use of swap space to avoid poor performance.

Be aware of the following:

- In addition to memory use for tables, some memory is allocated by an action to operate on a table. The memory limit that you set must accommodate this transient memory use.
- Global-scope tables are always memory mapped from a SASHDAT file or CAS_DISK_CACHE. The server process does not maintain the mapped memory that contains the data. Instead, the server passes references to data so that the session processes can map the data. As a result, the operating system often caches the data in physical memory and multiple sessions benefit from accessing the cached memory.
- If the cgroup memory limit and the specialized settings for short-lived tables are used at the same time, the risk of having a session killed is high. The cgroup restricts the amount of physical memory that the server process can use. When all memory has no backing store and the limit is reached, the server terminates one or more sessions. The server reviews the out-of-memory (OOM) score that the operating system assigns for each session. The server terminates the session with the highest OOM score.
- If a session is killed, the server log includes the message "Session 'xxxxx (ID)' terminated due to low memory."

The use of a Linux cgroup to limit memory use can be combined with Hadoop YARN for distributed servers that must share computing resources with other software that is installed on the same hardware. Administrators enable integration with YARN by setting the cas.useyarn configuration file option. When this option is enabled, as the server starts, it makes sure that YARN reserves the memory limit that is specified in the cas.memorysize option on each host. The server does not accept client connections until YARN can reserve the specified memory size. If YARN cannot reserve the memory, the error message "xxxx greater than max container memory" is generated.

Special Cases

SASHDAT Files

When a server has local data access to a SASHDAT file, the server does not copy blocks to the CAS_DISK_CACHE because the server can memory map the file itself.

For a single-machine server, SASHDAT files from a caslib with a data source type of PATH does not use the CAS_DISK_CACHE. For a distributed server, SASHDAT files from a caslib with a data source type of HDFS or DNFS also do not use CAS_DISK_CACHE. In those use cases, the server is already using the memory efficiency and performance strategies that are associated with memory mapping.

Short-lived Output Tables

In some cases, an output table from an action is large and only needed for a very short period of time —until it is used by another action or reporting is complete. In these cases, programmers and SAS applications can set the number of backup copies to zero and set the maxTableMem session option to a large size. (The maxTableMem session option specifies the amount of physical memory to allocate before the CAS_DISK_CACHE is used to store data in files.)

Use this strategy with care:

- An in-memory table with no redundant blocks cannot survive a node failure. There is no fault tolerance in this case.
- Because these settings insist on using physical memory only:

- Unlike tables that memory map from CAS_DISK_CACHE, data for the table might become written to the system swap space when all memory is in use. Use of the system swap space degrades performance severely.
- The session is at greater risk of being killed by the out-of-memory (OOM) killer that is part of the Linux operating system if Linux cgroups are used.

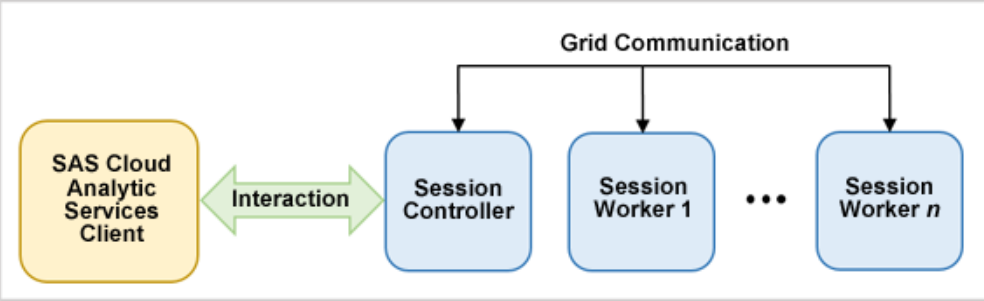
Sessions

About SAS Cloud Analytic Services Sessions

In SAS Cloud Analytic Services, sessions are used to enable clients to communicate with the server to request actions. When you log on to SAS Studio, you must use the CAS statement to create a session in order to connect to SAS Cloud Analytic Services. The code snippet **New Session** in SAS Studio provides the SAS code that is needed to create a session.

When you create your session, the server first authenticates your identity. If your identity is successfully authenticated, the server then starts the session controller process. In a distributed system, the server also sends a message to each machine in the cluster to start a session process for your client. The session processes for a distributed system are shown in the following figure.

Figure 6 *Session Processes in a Distributed System*



The session controller process does the following:

- manages client connections to the session
- monitors the state of the session
- waits for an interface to force a shutdown of the session

The session worker process in each worker machine waits for requests from the session controller process.

After the session processes are started, your client connects to the session. Each client can establish one connection to the session. Multiple clients can connect to the same session. The session is identified by a session name and a UUID. While connected to your session, you can create resources such as caslibs, SAS library references, tables, and user-defined formats. By default, these resources are visible only to your session. You can make these resources visible to all sessions, if needed. You can use the session identity to manage your sessions and session resources. A session executes

requests serially. While a request is processing, subsequent requests are queued until the current request is completed. For each request, the results are returned to the client that made the request.

A client typically terminates the session when it is no longer needed. You can disconnect from your session without terminating it, or you might be disconnected from it unexpectedly due to a loss of network connectivity. In either case, if there are no other client connections to your session, it becomes an orphan session. To conserve system resources, an orphan session is automatically terminated if a client connection is not established within a time-out period (60 seconds by default). The time-out is controlled by the TIMEOUT= session option.

By default, SAS Cloud Analytic Services enforces a limit of 5000 concurrent CAS sessions. For more information, see *SAS Viya Administration: SAS Cloud Analytic Services*.

Why Sessions Are Used

Sessions are used to provide the following:

- identification
- fault isolation
- efficiency
- resource tracking

Identification is provided by the use of user credentials. When you connect to SAS Cloud Analytic Services, the server authenticates your user credentials before it creates the session processes. The processes that are created for your session retain your identity and use it to access resources on the server.

Fault isolation is provided for each session through the isolation of its processes from those other client sessions and those of the server itself. If a problem occurs in your session, it does not impact other clients or the server.

Efficiency is achieved through the use of session-scope resources and through concurrent processing, when needed. By default, the resources that you create in your session have session scope. That is, they are visible within your session but not to other client sessions. Other users cannot access and modify your session-scope resources. Concurrent processing provides greater efficiency when processing large amounts of data, especially in distributed systems.

Finally, resource tracking is enabled through the use of resource metrics. Session option METRICS= enables the display of information about the system resources that are consumed by a session. By default, metrics information is disabled. When METRICS= is set to True for your session, system resource metrics are displayed after each request is executed. Metrics include real time, CPU time, memory usage, and so on. Using the metrics information, you can see the system resources that your session is using and make adjustments, if necessary.

Session Properties

Each session has configurable properties that control various session characteristics. A default value is provided for those properties that require a value. You can use session options to manage the properties for your session. See “Session Options” in *SAS Cloud Analytic Services: User’s Guide*.

Security

Authentication

Authentication is the process of verifying the identity of a user that is attempting to log on to or access software.

The server is configured to use an authentication provider during the deployment process. There are several interfaces that are authentication points, including SAS Studio, CAS Server Monitor, batch processing programs, and SAS Drive.

For details, see Authentication in *SAS Viya Overview*.

Authorization

Authorization is the aspect of security that determines which resources are available to each identity. The primary task is to provide appropriate access to any global caslib that you add. You can also manage access at the table, column, and row levels.

See SAS Viya Administration: Cloud Analytic Services Authorization.

Encryption for Data at Rest

SASHDAT Files

The SASHDAT file format is used when SAS Cloud Analytic Services saves tables to disk. This file format supports encryption for data at rest in two ways:

- file-by-file encryption with passwords that are supplied by programmers for encryption and decryption.
- all SASHDAT files in a caslib can be encrypted and decrypted seamlessly by specifying a password when the caslib is added to the server.

For more information, see Encryption in SAS Viya: Data at Rest.

Encrypted Volumes for CAS_DISK_CACHE

For Linux systems, the server can use directories for CAS_DISK_CACHE that correspond to volumes that are encrypted with Linux unified key setup (LUKS). Information that describes LUKS and how to encrypt a volume is provided by your Linux operating system vendor.

Encryption for Data in Motion

Data in motion is data that is being transmitted to another location. Data is most vulnerable while in transit. Sensitive data in transit should be encrypted.

In a full deployment of SAS Viya, all network connections are secured by default. SAS Viya is deployed with Transport Layer Security (TLS) to secure network connections and is fully compliant with SAS security standards.

In a SAS Viya programming-only deployment, the basic framework for security is included by default, but is not enabled by default.

For more information, see Encryption in SAS Viya: Data in Motion.

Caslibs

What is a Caslib?

A caslib is an in-memory space to hold tables, access control lists, and data source information. All data is available to CAS through caslibs and all operations in CAS that use data are performed with a caslib in place. Authorized users can add or manage caslibs with the CASLIB statement in SAS Studio.

Caslibs perform the following functions:

- provide access to data from the data source and access to in-memory tables that are copied from the data source. For example, a caslib named HPS might be defined as the HDFS path `/hps` and its subdirectories. All files with that path are potential data sources that the server can access.
- provide a space to hold temporary, in-memory tables that can have operations performed on them.
- provide a space to hold connection information to the data source. For example, a caslib that accesses an Oracle database contains connection information such as password, schema information, and data source type.
- provide a common interface into data providers.
- create an association with access controls that define what groups and individual users are authorized to do with the contents of the caslib.

You can also hide caslibs. A hidden caslib is omitted from most lists of caslibs. Tables in a hidden caslib are omitted from most lists of tables. For more information, see Reduced Visibility: Hidden Caslibs.

Caslib Scope

One property of a caslib is scope. A caslib can have one of two scopes: session scope or global scope. A session-scope caslib is accessible only from the session that adds it. This enables straightforward server-side data access to a programmer and does not interfere with other sessions. When you add a session caslib, your permission to do so is checked. If the permission is granted, the caslib exists only in the session where the caslib was added. By default, when a session is added with the CASLIB statement or the table.addCaslib action, it is added with session scope. Session-scope caslibs are useful if you do not need to share tables, but you only need to access them.

TIP Caslib names must be unique. If you add a session-scope caslib with the same name as a global-scope caslib, the global-scope caslib is effectively hidden because session-scope is searched first.

Global-scope caslibs can be accessible to any session on the server. Depending on access controls, users can share access to in-memory tables. Personal caslibs and pre-defined caslibs are global caslibs. Names of global caslibs must be unique across all sessions within a server. Global-scope caslibs are useful if you want other users to have access to the table, subject to access controls. Global caslibs are also useful if you want to access the table from a second client session.

Global-scope caslibs support both session-scope tables (for use in one session only) and global-scope tables (for use in many sessions, subject to access controls). -Session-scope caslibs support session-scope tables only.

For information about promoting tables with the CASUTIL procedure, see the documentation for PROC CASUTIL. You can also promote a table with the DATA step. For information, see *SAS Cloud Analytic Services: DATA Step Programming*.

Personal, Predefined, and Manually Added Caslibs

Caslibs can be personal, predefined, or manually added. Your level of authorization determines your interaction with each type of caslib. For complete information about caslib authorization, see *SAS® Viya Administration: CAS Authorization*.

personal caslibs
: When you start a session, a personal caslib is always available, is the active caslib, and has global scope. This enables you to access files and in-memory tables from any session that you start. However, they are personal and only your user ID can access the data. The initially active caslib is one of the following:

  CASUSER(*user-ID*)
  : The host CAS user location is `~/casuser/` where the `~` represents the user's home directory. This personal caslib applies to single-machine servers and distributed servers.

    For more information about the default CASUSER location, see The CASHostAccountRequired Custom Group in *SAS Viya Administration: Identity Management*.

  CASUSERHDFS(*user-ID*)
  : For distributed servers that use HDFS as a co-located or remote data provider for SASHDAT files, an administrator can specify an expression, such as `/user/%USER`, to map directories in

HDFS to personal caslibs. This enables users to access data from HDFS from directories like /user/sasdemo. This personal caslib applies to distributed servers only and is optional.

predefined caslibs
: Predefined caslibs are automatically created during deployment, managed by an administrator, and have global scope. They are created with the appropriate data access controls and are available to multiple users as defined by the administrator.

manually added caslibs
: Only administrators and authorized users can add caslibs with the CASLIB statement. Manually added caslibs are typically used in a program for ad-hoc data access that might not be generally available to all programmers that use the server.

  In most deployments, administrators configure caslibs that are associated with commonly used data sources that users need to access.

Note: On Windows, when a session is launched under the service identity, the path to CASUSER is `C:\casuser`. This invalid path is used to prevent sessions that are launched under the CAS service account from saving and loading data into a personal caslib.

Who Can Add, Delete, and Access a Global Caslib?

In a new deployment, only administrators can add and delete global caslibs. To extend those abilities to other users, see Caslib Management Privileges.

Each new global caslib is initially available to the user who added it. To expand access to other users, see Examples: Manage Access to a Caslib.

If you have a programming-only deployment, see Manage Access to a Global Caslib.

The Active Caslib

The active caslib is used to access data in CAS. If your administrator has enabled personal caslibs, then, when you start a session, the initially active caslib is your personal caslib. When you add a caslib, the newly added caslib becomes the active caslib. If you drop the active caslib, the initially active caslib is set as the active caslib again.

Authorized users can change the active caslib programmatically by adding a caslib with the CASLIB statement or by specifying the CASLIB= session option. To see what caslib is active, you can use the CASLIB _ALL_ LIST statement. For documentation about the CASLIB statement and CASLIB= option, see *SAS Cloud Analytic Services: User's Guide*.

Path-Based Caslibs

A path-based caslib references a data source that is a directory accessible to the SAS Cloud Analytic Services controller. For a distributed server, performance can be improved when the workers can access the directory with the DNFS, HDFS, or S3 data sources.

Path-based caslibs must meet the following criteria:

- The path must be fully qualified when you add a path-based caslib.
- For a session-scope caslib, the specified path must already exist.
- For a global-scope caslib, you can specify an existing path, or you can create a directory. If you create a directory, then the path up to the last directory must be an existing path.

The following table identifies the path-based caslib types and supported file types.

Table 4 *Supported File Types by Caslib*

Caslib Type	Supported File Types
ADLS (Azure Data Lake Storage, Gen 2)	■ CSV ■ ORC [6]
PATH	■ AUDIO [5] ■ CSV [3] ■ DOCUMENT [5] ■ DTA [2] ■ IMAGE [5] ■ JMP [2, 5] ■ ORC [6] ■ PARQUET [4] ■ SASHDAT ■ SAS7BDAT (specify BASESAS for the fileType parameter) [1] ■ SAV (specify SPSS for the fileType parameter) [2, 5] ■ VIDEO [5] ■ XLS [2] ■ XLSX (specify EXCEL for the fileType parameter) [2]
DNFS	■ AUDIO [5] ■ CSV [3] ■ DOCUMENT [5] ■ IMAGE [5] ■ PARQUET [4] ■ SASHDAT [7] ■ VIDEO [5]
S3	■ AUDIO [5] ■ CSV [3] ■ DOCUMENT [5] ■ IMAGE [5] ■ PARQUET [4] ■ SASHDAT

Caslib Type	Supported File Types
	■ VIDEO [5]

1 Requires the SAS Data Connector to SAS Data Sets.

2 Requires the SAS Data Connector to PC Files.

3 The CSV file type is used to identify any delimited file. For example, to read a semicolon delimited file with a .txt suffix, you can specify CSV as the file type and specify the delimiter.

4 Apache Parquet files are supported for single-machine and distributed servers that use Linux or Linux on Power.

5 These file types are read-only.

6 Apache ORC files are supported for single-machine and distributed servers that use Linux.

7 For a SASHDAT file in DNFS, treat the file as read-only unless you set the backing store to CASDISKCACHE. See “Controlling Use of CAS_DISK_CACHE” in *SAS Viya: System Programming Guide*.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies. Copyright © 2019, SAS Institute Inc., Cary, NC, USA. All Rights Reserved. June 2022 3.5-P2:casfun

THE POWER TO KNOW®