SAS® Viya® 3.4 Administration: Using the Command-Line Interfaces

Command-Line Interface: Overview

Introduction

SAS Viya contains administrative command-line interfaces (CLIs). In SAS Viya, a CLI is a user interface to the SAS Viya REST services where you enter commands on a command line and receive a response from the system. You can use a CLI to interact directly with SAS Viya programmatically without a GUI.

How to Use This Document

This document describes administrative tasks that enable you to run the CLIs, gives information about how to use the integrated help to use the CLIs, and includes examples of how to use each CLI. Here is a reading strategy for performing these tasks:

- Set up your environment to run the CLIs.
- Familiarize yourself with the CLI syntax and the integrated help within the CLIs.
- See “Inventory” on page 2 for information about each CLI plug-in, and links to examples.

Inventory

The following administrative CLIs are available in SAS Viya:

Name	Scope and Examples
admin	Hosts other CLIs that run as plug-ins to this one. The top-level administrative command-line interface that is used to initialize, authenticate, and execute other plug-ins.
audit	Gets SAS audit information. See “CLI Examples: Audit” on page 22.
authorization	Gets general authorization information and manages rules. See “CLI Examples: General Authorization” on page 26.

Name	Scope and Examples
backup	Manages backups. See “CLI Examples: Backup” on page 30.
cas	Manages CAS administration and authorization. See “CLI Examples: CAS Administration”, “CLI Examples: CAS Authorization” on page 23, and “CLI Examples: Formats” on page 40. For information about cas environment variables, see CAS Administration: Details on page 34.
compute	Manages the operations of the compute service. See “CLI Examples: Compute” on page 35.
configuration	Manages the operations of the configuration service. See “CLI Examples: Configuration” on page 35.
dagentsrv	Manages interactions with the SAS Data Agent server. See “Command-Line Interface” in *Cloud Data Exchange for SAS Viya: Administrator’s Guide*.
dataexplorer	Automates data management capabilities from the Data Explorer UI. See “CLI Examples: Data Explorer” on page 36.
devices	Manages mobile device blacklist and whitelist actions and information. See “CLI Examples: Device Management” on page 37.
folders	Gets and manages SAS folders. See “CLI Examples: Folders” on page 38.
fonts	Manages fonts that are provided by SAS as well as custom fonts that are registered in SAS Visual Analytics 8.3. See “CLI Examples: Fonts” on page 39.
identities	Gets identity information, and manages custom groups. See “CLI Examples: Identities” on page 44.
job	Manages the operations of the job flow scheduling service. See “CLI Examples: Job” on page 46.
launcher	Manages the operations of the launcher service. See CLI Examples: Launcher on page 46.
licenses	Manages SAS product license status and information. See “CLI Examples: Licensing” on page 47.
qkbs	Manages the SAS Quality Knowledge Bases (QKBs) on a Cloud Analytics Services (CAS) server. See “CLI Examples:Quality Knowledge Bases (QKBs)” on page 47.
reports	Manages SAS Visual Analytics 8.2 reports. See “CLI Examples: Reports” on page 49.

Name	Scope and Examples
restore	Manages restore operations. See “CLI Examples: Restore” on page 52.
scoreexecution	Manages resources that are no longer used by the Score Execution service. See CLI Examples: Score Execution on page 52.
tenant	Manages tenants in a multi-tenant deployment. See “CLI Examples: Tenant Administration” on page 52.
transfer	Promotes SAS content. See “CLI Examples: Transfer” on page 53.

Key Points

Here are key points for using the CLIs:

- SAS recommends that users with administrative privileges run the admin CLI in order to ensure optimal results. Users without administrative privileges are able to run the admin CLI, but the results might not always be as expected.
- To prepare to use the admin CLI plug-ins, see “Command-Line Interface: Preliminary Instructions” on page 5.
- Commands and subcommands are case-sensitive.
- You must precede the options of the commands with two hyphens (`--`) if the option is a word. If the option is a single letter, you can precede the option with two hyphens (`--`) or one hyphen (`-`). For example, you can use **`--help`** or **`-h`** for the Help global option.
- Linux special characters that are specified on the command line must be preceded (or escaped) with a backslash (\) so that they are not interpreted by the Linux shell. Linux special characters that are included in JSON template files do not need to be escaped.
- If a single parameter contains spaces (such as in long file names), you must enclose it in quotation marks in order to pass it as one item. Depending on what can be parsed by your operating system, other parameter values might need to be enclosed in quotation marks or escaped accordingly.
- Use the Help from within each CLI for information about the available commands, subcommands, and options. For the admin CLI plug-ins, see “Command-Line Interface: Syntax” on page 10.
- Timestamps that are returned from the sas-admin CLIs are in Universal Time Coordinated (UTC) format rather than the local time. By contrast, the timestamps returned from SAS Environment Manager are in local time.

  For example, in SAS Environment Manager, you might notice that a SAS Visual Analytics report has a modified date of April 26, 2018 09:23:47. This is the local time. However, the reports CLI will show that the modified date for the same report is 2018-04-26T13:23:47.314Z. This is UTC time.

Note: This document reflects sas-admin CLI functionality in the initial release of SAS Viya 3.4. The integrated Help supersedes this documentation and provides the most current information about any expanded or enhanced functionality.

About the Examples

The following points apply to all of the examples in this document:

- The examples assume that you have signed in to SAS Viya using the command line. See "Command-Line Interface: Preliminary Instructions" on page 5.
- The examples explicitly specify all necessary options. In practice, you might find it more efficient and concise to use environment variables where available. Remember to clear values for any environment variables when appropriate.
- The examples generally were run in a Linux environment.
- The examples include line breaks within commands to enhance readability. Do not include line breaks when you submit a command.
- The examples generally include single quotation marks when quotation marks are required. Use the quotation marks that are appropriate for your platform.
- The examples generally assume that you are using the default profile rather than a named profile.

  Note: If you logged in using a named profile, you must do one of the following to use that profile:

  - set the **SAS_CLI_PROFILE** environment variable to the name of the profile. This will remain in affect until you log off from the environment.

    For example, suppose that you have set the **SAS_CLI_PROFILE** environment variable to **Target1**.

    Then, you can log on to the environment with the "Target1" profile with this command: `sas-admin auth login`

    If you are signing in to an environment that has been configured for Kerberos, use this command: `sas-admin auth kerberos`.

  - include the **profile** global option on each CLI command as follows:

    ```
    sas–admin --profile profile-name CLI-name CLI-commands
    ```

    Then, you can sign in to the environment with the "Target1"profile with this command: `sas-admin --profile Target1 auth login`

    If you are signing in to an environment that has been configured for Kerberos, use this command: `sas-admin --profile Target1 auth kerberos`

Command-Line Interface: Preliminary Instructions

Complete the following required preliminary tasks before you use a CLI.

Set the SSL_CERT_FILE Environment Variable

If your environment is enabled for Transport Layer Security (TLS), you must set the **SSL_CERT_FILE** environment variable to the path location of the trustedcerts.pem file (if using the SAS default truststore) or the path location of your site-signed certificate (if using an internal truststore).

Note: For CLI users on Linux who are running the CLIs directly on the SAS machine, you might be able to source the consul.conf file rather than set the SSL_CERT_FILE environment variable.

Set the SSL_CERT_FILE environment variable to the following value:

Truststore	Operating System	Path
SAS default truststore	Linux	`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/trustedcerts.pem`
	Windows	`\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\cacerts\trustedcerts.pem`
internal truststore	Linux or Windows	*path_to_certificate*

Here is an example of how to set the environment variable in a Linux environment: `export SSL_CERT_FILE=/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/trustedcerts.pem`

Download the CLI Executable

Overview

You can download the admin CLI directly from the SAS Support website, and install the plug-ins that you need. This is the recommended approach for running the admin CLI from client machines. The most current plug-ins are available with the downloaded version of the admin CLI.

Important: If you are installing (or upgrading to) the Windows version of SAS Viya 3.4 for the first time, you must download and install the latest version of the admin CLI to client machines.

All the plug-ins are available for installation with the downloaded admin CLI with the exception of the following plug-ins:

backup	Will be available as a plug-in for installation with the admin CLI in a future release.
restore	Will be available as a plug-in for installation with the admin CLI in a future release.
compute	Might be available as a plug-in for installation with the admin CLI in a future release.
tenant	Is not available as a plug-in for installation with the admin CLI.

Note: Alternatively, the admin CLI, along with the plug-ins, is installed on the SAS Viya server during deployment. If you are not downloading the admin CLI to run on a client machine or you want to run a plug-in that is not available with the downloaded admin CLI, you can run the admin CLI directly from this location on the SAS Viya server:

- Linux: **`/opt/sas/viya/home/bin`**
- Windows: **`\Program Files\SAS\Viya\bin`**

For SAS Viya 3.4, the dataexplorer CLI plug-in is only available for download. It is not installed in the above directory location on the SAS Viya server.

How To

Here are the steps to download the admin CLI and install the plug-ins:

Note: You must have a valid SAS profile to download the admin CLI.

1 Go to the Support / Downloads and Hot Fixes page at http://support.sas.com/downloads/package.htm?pid=2133, and download the appropriate file to your machine.

2 Prepare the files for your environment:

UNIX	Expand the file to a directory from which you plan to run the CLI:
	`tar -xvzf /CLI-directory/sas-admin-cli-1.1.2-download-linux.tgz`
	From the directory, make sure that execution permission is set:
	`chmod +x sas-admin`

Windows or Macintosh	Unzip the file and place it in a directory from which you plan to run the CLI.

CAUTION! Do not place the admin CLI download file in the following directory:

Table 1 *CLI Default Directory Prohibited from Download*

Linux	**`/opt/sas/viya/home/bin`**
Windows	**`\Program Files\SAS\Viya\bin`**

3 Navigate to the directory location where you saved the CLI file. You must run the CLI commands directly from this directory, or you can add this directory to your system path to make the sas-admin CLI available to all CLI users.

4 Create a profile and sign in.

5 Install the CLI plug-ins by running these commands:

```
1 sas-admin plugins
2 sas-admin plugins list
3 sas-admin plugins list-repo-plugins
4 sas-admin plugins install --repo SAS plugin-name
```

1 Display the commands that are available to the admin CLI plug-ins command.

2 Display the admin CLI plug-ins that are currently installed.

3 List the plug-ins in the SAS repository that are available for installing.

4 Install a plug-in from the SAS repository.

Note: The plug-ins are installed in the home directory of the user who ran the **`sas-admin plugins`** command: ***`home-directory`*`/.sas/admin-plugins`**.

Run the Admin CLI Plug-In from a Shared Location

Important: SAS recommends that each admin CLI user should download a unique instance of the admin CLI and install a unique instance of a plug-in from the SAS repository. If your organization has a policy that enforces the download of only a single instance, it is still possible to make the CLI plug-ins available from a shared location as follows:

1 Navigate to the config.json file according to the operating system being used:

Table 2 *CLI Directory for config.json File*

Linux:	***home-directory*/.sas/admin-plugins**
Windows:	***home-directory*\.sas\admin-plugins**
Macintosh:	***home-directory*/.sas/admin-plugins**

2 Open the config.json file, and for each CLI plug-in, find the line that contains the location, and modify the directory path to point to a shared location that contains the CLI admin plug-in directory. Here is an example of this section in the config.json file for the transfer CLI plug-in:

```
"transfer": {
      "location": "\shared-location\admin-plugins\\sas-transfer-cli-1.3.5-20180330.1522434840.exe",
      "description": "tool for promoting contents across environments",
      "version": "1.3.5"
```

3 Propagate this change to the config.json file for all users:***home-directory*/.sas/admin-plugins**

Create at Least One Profile

If you have not already created a profile for the environment that you want to use, complete the following steps:

1 If you downloaded the CLI executables separately, navigate to the directory on the machine that contains the CLIs. If you run CLIs directly from the SAS Viya server and you have not downloaded the CLI executables separately, from a command prompt on the SAS Viya server, navigate to the appropriate directory:

Table 3 *CLI Default Directory Prohibited from Download*

Linux	**/opt/sas/viya/home/bin**
Windows	**\Program Files\SAS\Viya\bin**

2 At the command prompt, enter a command to initialize a new profile. Here are examples:

To create a default (unnamed) profile, enter:	`sas-admin profile init`
To create a profile called **prod**, enter:	`sas-admin --profile prod profile init`

You can use a named profile to access different environments using the same set of CLIs. See “Default Profile and Named Profile” on page 14 for information about why you might want to use a named profile.

Note: Running the **profile** global command creates a config.json file in this directory: ***home-directory*/.sas**. For more information, see “Overview” on page 13.

3 Respond to the subsequent prompts as follows:

`Service Endpoint`	Specify the URL for the SAS Viya environment. Use the following format: *communications-protocol*://*web-server-host-name*:*web-server-port* For example: `https://host.example.com`

`Output type`	Specify your preferred format for CLI output (`text`, `json`, or `fulljson`). **Note:** For more information about the output types, see "Output Type" on page 13.
`Enable ANSI colored output`	Specify whether to enable colored output (`y` or `n`).

4 Repeat steps 2 and 3 for any additional profiles that you want to create.

Use a Profile to Sign In

1 If you downloaded the CLI executables separately, navigate to the directory on the machine that contains the CLIs. If you run CLIs directly from the SAS Viya server and you have not downloaded the CLI executables separately, from a command prompt on the SAS Viya server, navigate to the appropriate directory:

Table 4 *CLI Default Directory Prohibited from Download*

Linux:	**`/opt/sas/viya/home/bin`**
Windows:	**`\Program Files\SAS\Viya\bin`**

2 At the command prompt, enter a command to initiate the sign-in process. Here are examples:

To use your default (unnamed) profile (assuming that the **`SAS_CLI_PROFILE`** environment variable is not set), enter:	`sas-admin auth login`
To use a profile called **`prod`**, enter:	`sas-admin --profile prod auth login`

Note: Kerberos is supported for Linux and Windows environments. If you are signing in to an environment that has been configured for Kerberos, replace `auth login` with `auth kerberos` in the preceding examples.

3 At the subsequent prompts, enter your user ID and password. No prompts appear in environments that have been configured for Kerberos.

By default, your authentication remains active for 12 hours. You can use the **`auth logout`** command to sign out.

Note: When you run the **`auth login`** global command, a bearer token is written to the credentials.json file in this directory: ***`home-directory`*`/.sas`**. For more information, see "Overview" on page 13.

Note: If you logged in using a named profile, you must do one of the following to use that profile:

- set the **`SAS_CLI_PROFILE`** environment variable to the name of the profile. This will remain in affect until you log off from the environment.

  For example, suppose that you have set the **`SAS_CLI_PROFILE`** environment variable to **`Target1`**.

  Then, you can log on to the environment with the "Target1" profile with this command: `sas-admin auth login`

If you are signing in to an environment that has been configured for Kerberos, use this command: `sas-admin auth kerberos`.

- include the **`profile`** global option on each CLI command as follows:

  ```
  sas–admin --profile profile-name CLI-name CLI-commands
  ```

  Then, you can sign in to the environment with the “Target1”profile with this command: `sas-admin --profile Target1 auth login`

  If you are signing in to an environment that has been configured for Kerberos, use this command: `sas-admin --profile Target1 auth kerberos`

See Also

“Command-Line Interface: Overview” on page 2

Command-Line Interface: Syntax

Structure

The basic structure of a command-line interface (CLI) command is:

sas-admin *interface-name [global options] command [command options] [subcommand] [subcommand options] [arguments]*

sas-admin
specifies the sas-admin CLI.

interface name
specifies the CLI plug-in.

[global options]
specifies options that are applicable to all CLIs.

command
specifies a command that is specific for the CLI that you are using.

[command options]
specifies options for the CLI-specific command that you are using.

[subcommand]
specifies a subcommand for the CLI command that you are using.

[subcommand options]
specifies options for the subcommand.

[arguments]
specifies arguments for options.

Here are some basic examples of issuing CLI commands:

Example: Change the output type that is used with the audit CLI to JSON.

```
sas-admin audit --output json
```

Example: Show more detailed information for the mobile device CLI **`blacklist list`** command.

```
sas-admin --verbose devices blacklist list
```

For information about the CLIs that are available, see “Inventory” on page 2.

Integrated Help

Global Commands

Use the integrated help within the CLI to learn about the available global commands, plug-ins, and global options. The global options apply to each CLI plug-in.

Example: List all the global commands, plug-ins, and global options for the admin CLI.

```
sas-admin help
```

Here is the output from this command in a Linux environment:

```
NAME:
   sas-admin - SAS Administrative Command Line Interface

USAGE:
   sas-admin [global options] command [command options] [arguments...]

VERSION:
   1.1.13

COMMANDS:
   authenticate, auth, authn    Handles authentication to the target environment.
   help, h            Shows a list of commands or help for one command.
   plugins            Manages plugins.
   profile, prof        Shows and updates options.

PLUGINS:
   audit
   authorization
   backup
   cas
   compute
   configuration
   devices
   folders
   fonts
   identities
   job
   launcher
   licenses
   qkbs
   reports
   restore
   scoreexecution
   tenant
   transfer

GLOBAL OPTIONS:
   --colors-enabled        Enables or disables ANSI colored output. [$SAS_CLI_COLOR]
   --help, -h            Shows help.
   --insecure, -k        Allows connections to TLS sites without validating the server certificates.
   --locale "en"        Specifies a locale to use. [$LC_ALL, $LANG]
   --log-file            Specifies the file to write log events to. [$SAS_LOG_FILE]
   --output            Specifies output format - text, json, fulljson. [$SAS_OUTPUT]
```

```
      --profile, -p "Default"    Specifies a named profile to use. [$SAS_CLI_PROFILE]
      --quiet, -q            Quiets spurious output, data only.
      --sas-endpoint         Sets the URL to the SAS services. [$SAS_SERVICES_ENDPOINT]
      --verbose            Shows detailed processing information and output.
      --version, -v        Prints the version.

   COPYRIGHT:
      (c) 2016-2018 SAS Institute Inc. All Rights Reserved.
```

Example: Show the version of the CLI.

```
sas-admin --version
```

Here is the output from this command in a Linux environment:

```
sas-admin version 1.1.13
```

CLI Plug-in

Use the integrated help within the CLI to learn about the available commands, subcommands, and options for each CLI plug-in. Use the same syntax for each CLI, substituting the CLI plug-in name or command that you are getting help for.

Example: List all the commands for the devices plug-in to the admin CLI.

```
sas-admin devices --help
```

Here is the output from this command in a Linux environment:

```
NAME:
   sas-devices

USAGE:
   sas-admin devices command [command options] [arguments...]

COMMANDS:
   authorized-devices    Manages the authorization of devices.
   blacklist        Manages the list of blacklisted devices.
   enforcement        Manages the policy enforcement of mobile devices.
   help, h        Shows a list of commands or help for one command.
   last-access        Manages the set of history records of the devices that use the mobile application.
   whitelist        Manages the list of whitelisted devices.
```

Example: List the subcommands of the **`blacklist`** command that is a part of the devices plug-in to the admin CLI.

```
sas-admin devices help blacklist
```

Here is the output from this command in a Linux environment:

```
NAME:
   sas-admin devices blacklist - Manages the list of blacklisted devices.

USAGE:
   sas-admin devices blacklist [arguments...]

COMMANDS:
   add          Adds a device to the blacklist.
   clear        Clears the blacklist of all device IDs that are present.
   delete       Deletes a device from the blacklist.
```

```
   list         Lists the devices in the blacklist.
```

Example: List the options of the **list** subcommand of the **blacklist** command that is a part of the devices plug-in to the admin CLI.

```
   sas-admin devices blacklist help list
```

Here is the output from this command in a Linux environment:

```
NAME:
   sas-devices blacklist list - Lists the devices in the blacklist.

USAGE:
   sas-devices blacklist list [command options] [arguments...]

OPTIONS:
   --all    Returns all of the devices in the blacklist.
   --limit "10"    Specifies the maximum number of devices to return. The default value is 10.
   --start "0"    Specifies the 0-based offset of the first device to return. The default value is 0.
```

Command-Line Interfaces: Concepts

Output Type

You must specify an output type for your CLI when you create your profile. The output types for CLIs are as follows:

- text

  Specifies that the output from the CLI is in text format. This is the default format.

- JSON

  Specifies that the output from the CLI is in JSON format.

- fulljson

  Specifies that the output from the CLI is the entire JSON response. This option is useful when writing scripts in which you need access to the entire response in order to complete a task.

Global Command: Output

Use the **output** global command to specify the output format for a CLI command. Doing so overrides any output type that was specified when you created your profile or that was set in the **SAS_OUTPUT** environment variable.

Global Command: Profile

Overview

Use the **profile** global command to create the connection profile that defines your SAS Viya deployment. This process creates the following two files in the directory ***home-directory*/.sas**:

- config.json

Contains information about your SAS Viya deployment, including the name of the connection profile, the service endpoint, and the output type (**text**, **json**, or **fulljson**).

Note: If you downloaded the CLI executables, the config.json file is created in this directory:

Table 5 *CLI Directory for config.json File*

Linux	***home-directory*/.sas/admin-plugins**
Windows	***home-directory*\.sas\admin-plugins**
Macintosh	***home-directory*/.sas/admin-plugins**

- credentials.json

  Will contain the authentication tokens for your session that are created after you issue the **auth login** global command.

Default Profile and Named Profile

You can create a default profile or a named profile. If you do not specify a named profile in the **sas-admin auth login** command, and the **SAS_CLI_PROFILE** environment variable is not set, then the default profile is used.

You can use a named profile if you need to work with two or more environments simultaneously from the same machine using the same set of CLIs. When you log on to an environment with a named profile, the associated token is stored for that specific profile. You can work in different environments at the same time by specifying different profile names in the commands. For example, suppose that you have different development, test, and production environments. You can create a separate, named profile for each environment to help distinguish the environment that you are connecting to.

You can also use a named profile to eliminate the requirement to create a profile with the correct settings every time you log on. Suppose that you know that you want to use the JSON output type and a certain endpoint. You can create a named profile with these settings. Then when you want to log on, you can specify this profile name and eliminate the need to create a new profile with these settings

Note: If you logged in using a named profile, you must do one of the following to use that profile:

- set the **SAS_CLI_PROFILE** environment variable to the name of the profile. This will remain in affect until you log off from the environment.

  For example, suppose that you have set the **SAS_CLI_PROFILE** environment variable to **Target1**.

  Then, you can log on to the environment with the “Target1” profile with this command: `sas-admin auth login`

  If you are signing in to an environment that has been configured for Kerberos, use this command: `sas-admin auth kerberos`.

- include the **profile** global option on each CLI command as follows:

  ```
  sas–admin --profile profile-name CLI-name CLI-commands
  ```

  Then, you can sign in to the environment with the “Target1”profile with this command: `sas-admin --profile Target1 auth login`

  If you are signing in to an environment that has been configured for Kerberos, use this command: `sas-admin --profile Target1 auth kerberos`

Examples

Here are typical examples of creating default and named profiles in an environment that has not been configured to use Kerberos:

Example: Create a default connection profile, specify the **text** output type, and specify the path to your SAS Viya deployment as the service endpoint.

Note: The **SAS_CLI_PROFILE** environment variable must not be set in order for this example to work.

```
sas-admin profile init
```

- At the prompt for **Service Endpoint**, enter the URL for the SAS Viya environment as follows:**https://*endpoint URL***
- At the prompt for **Output type**, enter **text**.
- At the prompt for **Enable ANSI colored output**, enter **y** or **n**.

Here is the config.json file that was created. **"Default"** indicates that a default profile was created.

```
{
  "Default": {
    "ansi-colors-enabled": "false",
    "output": "text",
    "sas-endpoint": "https://endpoint URL"
  }
}
```

Example: In the same environment, create a connection profile that is named **Target1**, specify the **json** output type, and specify the path to your SAS Viya deployment as the service endpoint.

```
sas-admin --profile Target1 profile init
```

Here is the config.json file that was created. Notice that there are now two profiles: "Default" and "Target1". Notice that the output type for the "Target1" profile is JSON.

```
{
  "Default": {
    "ansi-colors-enabled": "false",
    "output": "text",
    "sas-endpoint": "https://endpoint URL
  },
  "Target1": {
    "ansi-colors-enabled": "false",
    "output": "json",
    "sas-endpoint": "https://endpoint URL
```

See "Command-Line Interface: Preliminary Instructions" on page 5 for more information about creating profiles.

Example: In the same environment, log on using the **Target1** profile that you created in the previous example.

```
sas-admin --profile Target1 auth login
```

Global Command: Authenticate

Use the **authenticate** global command to log on or log off from the environment. This process stores a token in the credentials.json file.

Note: The token expires after 12 hours, so you might need to re-execute the command at a later time to reconnect to the environment.

To log on , run the **auth login** command. For syntax information, see "Structure" on page 10.

Note: Kerberos is supported for Windows and Linux environments. Sign in to environments that have been configured for Kerberos with this command: `auth kerberos`

Here are typical examples from a Linux environment that has not been configured for Kerberos:

- Use this command to log on with a default profile: `sas-admin auth login`. The **`SAS_CLI_PROFILE`** environment variable must not be set in order for this example to work.
- Use this command to log on with the profile named "Target1": `sas-admin --profile Target1 auth login`.

  If the **`SAS_CLI_PROFILE`** environment variable is set to **`Target1`**, then you can log on to the "Target1" environment as follows: `sas-admin auth login`.

To log off, issue the **`auth logout`** command. For syntax information, see "Structure" on page 10. Here are typical examples:

- Use this command to log off from the SAS Viya environment that is specified in your default profile: `sas-admin auth logout`. The **`SAS_CLI_PROFILE`** environment variable must not be set in order for this example to work.
- Use this command to log off from a SAS Viya environment that is specified in the profile named "Target1": `sas-admin --profile Target1 auth logout`.

  If the **`SAS_CLI_PROFILE`** environment variable is set to **`Target1`**, then you can log off from the "Target1" environment as follows: `sas-admin auth logout`.

Source Files: JSON Templates

Overview

For some CLI commands, you can include the command options in a separate JSON file by using the **`source-file`** option. This file is referred to as a template or source file. You might choose this option for the following reasons:

- to accelerate the process for entering CLI commands by including common options in a template
- to automatically create a template by piping output to a file

Details

- If a duplicate CLI option is specified on the command line and in the source file, the command-line option takes precedence over the option in the source file.
- Although some CLI options can be specified in the source file and on the command line, other CLI options can be specified only on the command line.

Create Caslibs from JSON Templates

Templates that are used for creating caslibs must be JSON files. You can generate sample JSON templates for creating caslibs with the cas CLI **`generate-cas-samples`** command.

Example: Generate a template for creating a caslib. In this example, a template is generated for creating a path caslib. You will generate the path.json sample template, and use it to create the new template.

```
1 sas-admin cas generate-cas-samples --output-location sample-template-path

2 sas-admin cas caslibs create --help

3 sas-admin cas caslibs create path --server serverA  --path path --name caslibA
 --source-file /sample-template-path/path.json
```

```
4 sas-admin --output json cas caslibs show-info --name caslibA > pathA.json
```

1 Generate sample templates. The path.json template is one of the files that are created. Specify the folder location for the samples templates in the **output-location** option.

2 List the types of caslibs that can be specified. You will use this value with the **create** command in the next step.

3 Create the new path caslib with the name caslibA. In the **source-file** option, specify the path.json sample template that you just created. In the next step, you will use the information about the new caslib to create a JSON template.

   On the command line, specify any options for overriding the values in the sample template. In this example, the server and path were specified.

4 Redirect the JSON output for caslibA to a file named pathA.json, which you can then use as a template for creating new caslibs. Specify the use of JSON output with the **output json** global option.

Example: Create a caslib from a template. This example uses the template that you created in the previous example.

```
1 sas-admin cas caslibs create --help

2 sas-admin cas caslibs create path --server serverB  --path pathB --name caslibB --source-file pathA.json

3 sas-admin --output json cas caslibs show-info --name caslibB --server serverB
```

1 List the types of caslibs that can be specified. You will use this value with the **create** command in the next step.

2 Create a new caslib named caslibB by specifying the pathA.json file that you created in the previous example in the **source-file** option. On the command line, specify any options for overriding the values in the template. In this example, the server and path were specified.

3 Show information about caslibB in JSON format. Note that the server and path that were specified on the command line overwrote the server and path values that were in the template.

Create User-Defined Formats from JSON Templates

In templates that are used for adding a user-defined format, templates that include command options as well as the format ranges must be JSON files. You can generate sample JSON templates for creating user-defined formats with the cas CLI **generate-cas-samples** command.

Example: Create a JSON template for adding a user-defined format.

```
1 sas-admin cas generate-cas-samples --output-location sample-template-path

2 sas-admin cas format-libraries add-format json --format-library format_libraryA --server serverA
 --source-file  /sample-template-path/gender.json

3 sas-admin cas --output json format-libraries show-format-ranges --format 'en_us-$gender'
 --format-library format_libraryA --server serverA  > formatA_template.json

4 sas-admin cas format-libraries add-format json --format-library format_libraryB
 --server serverA --source-file formatA_template.json
```

1 Generate sample templates. The gender.json template is one of the files that are created. Specify the folder location for the samples templates in the **output-location** option.

2 Add a new format to a format library. In the **source-file** option, specify the gender.json sample template that you just created. In the next step, you will use the information about the new format to create a JSON template.

   On the command line, specify any options for overriding the values in the sample template. In this example, the format library was specified. If the command is successful, the server returns a message that identifies

the name of the format that you added. In this example, the **en_us-$gender** file is created. The **en_us-** prefix identifies the English locale.

3 Redirect the JSON output for the new format to a JSON file using the **format-libraries show-format-ranges** command. Specify the use of JSON output with the **output json** global option. The format name to use in the **format** option was returned from the server when you created the format. Since the gender format is a character format, you must precede it with a dollar sign ($) and enclose it in quotation marks.

You can use the JSON file as a template for creating new caslibs. Note that the format library that you specified when you created the new format is saved in the new JSON file.

4 Create a new format using the new JSON file as the **source-file** template. On the command line, specify any options for overriding the values in the template. In this example, a new format library was specified.

Create Policies from JSON Templates

Templates that are used for creating CAS server policies must be JSON files.

Important: For CAS to use resource management policies, the environment variable **CAS_ENABLE_CONSUL_RESOURCE_MANAGEMENT** must be set on the CAS server. For more information, see Environment Variables.

Global Caslib Policies

Example: Create a policy from a sample JSON template.

```
1 sas-admin cas generate-cas-samples --output-location sample-template-path

2 sas-admin  cas servers policies define global-caslibs
  --server serverA --source-file /sample-template-path/policies-examples/globalCaslibs.json
```

1 Generate sample templates. Specify the folder location for the sample templates in the **output-location** option. The policies-examples folder is created, and one of the files in the folder is globalCaslibs.json.

Edit the globalCaslibs.json file for your environment. You can add new policy options or change values of existing policy options. Save the file.

Note: As an alternative to manually editing the JSON file, you can add additional options by using the **attributes** option.

2 Define a global caslib policy by specifying the edited globalCaslibs.json sample template file in the **source-file** option. The values in the globalCaslibs.json file are used to define the new policy. Information about the new policy is returned from the server after the policy is created. Make sure that the policy is correct.

Example: Create a JSON template for creating a global caslib policy.

```
1 sas-admin cas generate-cas-samples --output-location sample-template-path

2 sas-admin  cas servers policies define global-caslibs
  --server serverA --source-file /sample-template-path/policies-examples/globalCaslibs.json

3 sas-admin --output json cas servers policies show-info --policy globalCaslibs
 --server serverA > /path/global-caslib-template.json

4 sas-admin  cas servers policies define global-caslibs
 --attributes sessionTables:1000000000 --server serverA --source-file /path/global-caslib-template.json

5 sas-admin cas servers policies show-info --policy globalCaslibs --server serverA
```

1 Generate sample templates. Specify the folder location for the sample templates in the **output-location** option. The policies-examples folder is created, and one of the files in this folder is globalCaslibs.json.

Edit the globalCaslibs.json file for your environment. You can add new policy options or change values of existing policy options. Save the file.

2 Define a global caslib policy by specifying the edited globalCaslibs.json template file in the **source-file** option. The values in the globalCaslibs.json file are used to define the new policy. Information about the new policy is returned from the server after the policy is created. You will use this new policy to create a JSON template.

3 To create the template, redirect the JSON output for the new policy to a JSON file using the **servers policies show-info** command. Specify the use of JSON output with the **output json** global option. The policy name to use in the **policy** option was returned from the server when you created the policy in the previous step. Open the new JSON file to verify that it is correct.

4 Create a new global caslib policy using the new JSON template in the **source-file** option. On the command line, specify any new attributes or attributes to override the values in the template. In this example, a session tables value was specified. The value that you assign to the session tables attribute must be specified in bytes.

5 Display information about the new global caslib policy to verify the presence of the desired attributes and the additional session tables attribute.

Priority Assignments Policies

Note: The groups to which you are assigning a priority-level must exist in LDAP or as a custom group in order for the policy to work.

Example: Create a policy from a sample JSON template.

```
1 sas-admin cas generate-cas-samples --output-location sample-template-path

2 sas-admin  cas servers policies define priority-assignments
  --server serverA --source-file /sample-template-path/policies-examples/priorityAssignments.json
```

1 Generate sample templates. Specify the folder location for the sample templates in the **output-location** option. The policies-examples folder is created, and one of the files inside this folder is priorityAssignments.json.

   Edit the priorityAssignments.json file for your environment. You can add new policy options or change values of existing policy options. Save the file.

   Note: As an alternative to manually editing the JSON file, you can add additional options by using the **attributes** option in the next step.

2 Define a priority assignments policy by specifying the edited priorityAssignments.json sample template file in the **source-file** option. The values in the priorityAssignments.json file are used to define the new policy. Information about the new policy is returned from the server after the policy is created. Make sure that the policy is correct.

Example: Create a JSON template for creating a priority assignment policy.

```
1 sas-admin cas generate-cas-samples --output-location sample-template-path

2 sas-admin cas servers policies define priority-assignments
--server serverA --source-file /sample-template-path/policies-examples/priorityAssignments.json

3 sas-admin --output json cas servers policies show-info --policy priorityAssignments
 --server serverA > /path/priority-assignments-template.json

4 sas-admin cas servers policies define priority-assignments
--attributes userM:1 --server serverA --source-file /path/priority-assignments-template.json

5 sas-admin cas servers policies show-info --policy priorityAssignments --server serverA
```

1 Generate sample templates. Specify the folder location for the sample templates in the **output-location** option. The policies-examples folder is created, and one of the files in the folder is priorityAssignments.json.

   Edit the priorityAssignments.json file for your environment. You can add new policy options or change values of existing policy options. Save the file.

2 Define a priority assignments policy by specifying the edited priorityAssignments.json template file in the **source-file** option. The values in the priorityAssignments.json file are used to define the new policy. Information about the new policy is returned from the server after the policy is created. You will use this new policy to create a JSON template.

3 To create the template, redirect the JSON output for the new policy to a JSON file using the **servers policies show-info** command. Specify that JSON output is used with the **output json** global option. The policy name to use in the **policy** option was returned from the server when you created the policy in the previous step. Open the new JSON file to verify that it is correct.

4 Create a new priority assignments policy using the new JSON template in the **source-file** option. On the command line, specify the users and their priority levels with the **attributes** option. In this example, userM is added to the policy with priority 1. The server returns information about the new policy after it is defined. Verify that userM is now included in the policy as well as the groups that were already in the source file.

5 Display information about the new priority assignments policy to verify the presence of the desired attributes and the additional user.

Priority-Level Policies

Note:

Priority-level policies can be used to assign resources based on a user's group membership or by explicit assignment with the priority assignment policy. To facilitate assigning resources by group membership, define custom groups with the same names as the priority-level policy and assign users to the groups.

Example: Create a policy from a sample JSON template.

```
1 sas-admin cas generate-cas-samples --output-location sample-template-path

2 sas-admin  cas servers policies define priority-levels --server serverA --priority 3
--source-file /sample-template-path/policies-examples/cas-shared-default-priority-2.json
```

1 Generate sample templates. Specify the folder location for the sample templates in the **output-location** option. The policies-examples folder is created, and one of the files in the folder is cas-shared-default-priority-2.json.

   Edit the cas-shared-default-priority-2.json file with values appropriate for your environment. You can add new policy options or change values of existing policy options. Save the file.

2 Define a priority-level policy by specifying the edited cas-shared-default-priority-2.json template file in the **source-file** option. Use the **priority** option to specify the priority for the policy. In this example, the new policy is priority 3. Information about the new policy is returned from the server after the policy is created. Make sure that the policy is correct.

Example: Create a JSON template for creating a priority-level policy.

```
1 sas-admin cas generate-cas-samples --output-location sample-template-path

2 sas-admin cas servers policies define priority-levels --priority 3
--server serverA --source-file  /sample-template-path/policies-examples/cas-shared-default-priority-2.json

3 sas-admin --output json cas servers policies show-info --policy serverA-priority-3
 --server serverA > /path/priority-levels-template.json

4 sas-admin cas servers policies define priority-levels --priority 4 --cpu 30 --session-tables 2000000000
 --server serverA --source-file /path/priority-levels-template.json
```

1 Generate sample templates. Specify the folder location for the sample templates in the **output-location** option. The policies-examples folder is created, and one of the files in the folder is cas-shared-default-priority-2.json.

   Edit the cas-shared-default-priority-2.json file with values appropriate for your environment. You can add new policy options or change values of existing policy options. Save the file.

2 Use the edited cas-shared-default-priority-2.json template file in the **source-file** option to define a new policy. You will use this new policy to create a JSON template.

   In this example, a policy for priority level 3 is specified with the **priority** option. Information about the new policy is returned by the server after the policy is created. The name of priority-level policies is generated as follows: *server*-**priority**-*priority*. Therefore, in this example, the policy is named serverA-priority-3.

3 To create the template, redirect the JSON output for the new policy to a JSON file using the **servers policies show-info** command. Specify the use of JSON output with the **output json** global option. Open the new JSON file to verify that the contents are correct.

4 Create a new policy using the new JSON template in the **source-file** option. This example a shows a policy for priority level 4.

   Specify the appropriate values for priority level 4. This example specifies different values for the CPU and session-tables attributes with the **cpu** and **session-tables** options. The value that you assign to the **session-tables** attribute must be in bytes. The server returns information about the new policy after it is defined. Verify that the serverA-priority-4 policy was created with the correct values for CPU and session tables.

   Note: To enable the policy resource settings by group membership, create a custom group that has the same name as the priority-level policy. For example, if the policy is named serverA-priority-4, you must create a custom group with the same name. Then, add the users to the group who will be granted this priority level. These users will automatically be granted the resource settings that are specified in the priority-level policy by being a member of this custom group.

See Also

"CAS Resource Management Policies" in *SAS Viya Administration: SAS Cloud Analytic Services*

Source Files: Data

Overview

For some CLI commands, you can pass data to the CLI by including it in a separate file by using the **source-file** option. This file is referred to as a template or source file. You might want to use this option to enter multiple data items at the same time rather than entering multiple CLI commands to enter data items.

Examples

Format Ranges

Templates that are used for adding the ranges for new user-defined formats must be CSV files or JSON files. You can generate sample templates for adding ranges to user-defined formats with the cas CLI **generate-cas-samples** command.

Here is an example of a formats source file in CSV format:

```
F,female
M,male
f,female
m,male
```

See Also

See "Creating Formats" on page 40 for examples of creating user-defined formats with source files.

CAS Server Paths Lists

Files that are used for adding paths to the paths list must be text files.

Here is an example of a paths list source file:

```
/opt/sas/viya/config/folderA
/opt/sas/viya/config/folderB
/opt/sas/viya/config/folderC
```

Example: Add multiple paths to the paths list for the specified CAS server using a template. This example assumes that you have a template text file that is formatted so that each path is on a separate line, as shown in the preceding source file.

```
1 sas-admin cas servers paths-list list --server serverA

2 sas-admin cas servers paths-list add-paths --server serverA --source-file /path_to_template/templateA.txt

3 sas-admin cas servers paths-list list --server serverA
```

1 Display the current paths list for serverA.

2 Add the paths in the template text file to the paths list for serverA.

3 Display the current paths list for serverA to confirm that the paths were added.

Devices Blacklist

Files that are used for adding mobile devices to the blacklist must be text files.

Here is an example of a mobile devices source file:

```
deviceID4
deviceID5
deviceID6
```

Example: Add multiple devices to the blacklist using a template. This example assumes that you have a template text file that is formatted with each device on a separate line, as shown in the preceding source file.

```
sas-admin devices blacklist add --source-file /path_to_template/templateA .txt
```

Command-Line Interface: Examples

CLI Examples: Audit

The following examples assume that you have already signed in to SAS Viya at the command line. See “Command-Line Interface: Preliminary Instructions” on page 5.

Examples

Example: List the records of audit entries for reports.

```
sas-admin audit list --application reports
```

Example: List the records of audit entries of type security and sort by user.

```
sas-admin audit list --sort-by user --type security
```

Example: List the records of audit entries of type security and state of success, and then write the results as CSV to an output file:

```
sas-admin audit list --state success --type security --csv /tmp/outputfile.text
```

See Also

- “Command-Line Interface: Overview” on page 2
- SAS Viya Administration: Auditing

CLI Examples: CAS Authorization

The following examples assume that you have already signed in to SAS Viya at the command line. See “Command-Line Interface: Preliminary Instructions” on page 5.

Getting Access Information

Example: List tableA’s direct access controls.

```
sas-admin cas tables list-controls --server serverA --caslib caslibA --table tableA
```

Example: List tableA’s direct access controls and inherited settings.

```
sas-admin cas tables list-controls --server serverA --caslib caslibA --table tableA --list-type all
```

Example: Show effective (net) access to tableA for userA.

```
sas-admin cas tables list-controls --server serverA --caslib caslibA --table tableA
--control-type effective --user userA
```

Example: Show effective access to tableA for groupA and groupB.

```
sas-admin cas tables list-controls --server serverA --caslib caslibA --table tableA
--control-type effective --group 'groupA|groupB'
```

Example: Show the source of userA’s access to tableA.

```
sas-admin cas tables list-controls --server serverA --caslib caslibA --table tableA
--control-type origin --user userA
```

Managing Access Controls

Example: Remove all direct access controls from tableA.

```
sas-admin cas tables clear-controls --server serverA --caslib caslibA --table tableA
```

Example: Set a simple row-level access control on the CARS table so that members of groupA can see only those rows where the value in the Make column is **`Ford`**.

```
sas-admin cas tables add-control --server serverA --caslib caslibA --table CARS --group groupA
--grant Select --where "make='Ford'"
```

Example: Set an identity-based, row-level access control on the Salary table. The reason is so that each authenticated user can see only those rows where the value in the User column is his or her own user ID.

```
sas-admin cas tables add-control --server serverA --caslib caslibA --table salary --group "*"
--grant Select --where "User='SUB::SAS.Userid'"
```

Example: Enable guests to read data in the Public caslib.

```
1 sas-admin cas caslibs add-control --server serverA --caslib Public --guest --grant ReadInfo
 --superuser
```

```
2 sas-admin cas caslibs add-control --server serverA --caslib Public --guest --grant Select
 --superuser
```

1 This example is applicable to only a deployment where guest access is enabled.

In the standard configuration, because only a privileged user can modify access to the Public caslib, the superuser option is specified here. Only a member of the Superuser role for the specified CAS server can obtain elevated privileges by specifying the superuser option.

2 The grants in this example support reading of data, but do not support just-in-time loading of data. Instead of granting LimitedPromote to guest, consider using a different technique for ensuring that data is loaded.

Because this example does not create and reuse a dedicated superuser session, you must specify the superuser option in each command where elevated privileges are needed.

Example: Enable groupA to read data (and perform just-in-time data loading) in a new caslib.

```
1 sas-admin cas caslibs add-control --server serverA --caslib caslibA --group groupA --grant ReadInfo

2 sas-admin cas caslibs add-control --server serverA --caslib caslibA --group groupA --grant Select

3 sas-admin cas caslibs add-control --server serverA --caslib caslibA --group groupA
 --grant LimitedPromote
```

Example: Make the same changes as in the preceding example, but use an access control transaction so that you can review your changes before you commit them to the server.

```
1 sas-admin cas sessions create --name mysess --server serverA --superuser

2 sas-admin cas transactions checkout --session-id XYZ --server serverA --caslib caslibA

3 sas-admin cas caslibs add-control --session-id XYZ --server serverA --caslib caslibA
--group groupA --grant ReadInfo

4 sas-admin cas caslibs add-control --session-id XYZ --server serverA --caslib caslibA
--group groupA --grant Select

5 sas-admin cas caslibs add-control --session-id XYZ --server serverA --caslib caslibA
--group groupA --grant LimitedPromote

6 sas-admin cas caslibs list-controls --session-id XYZ --server serverA --caslib caslibA

7 sas-admin cas transactions commit --session-id XYZ --server serverA

8 sas-admin cas sessions delete --session-id XYZ --server serverA
```

1 Start a session. If you are a member of the Superuser role for the associated CAS server, give the session Superuser status.

2 Check out the caslib into the session that you just started. Use the session ID that is returned from the preceding command. The session-id value **XYZ** is used here for simplicity. Checking out an object automatically starts a transaction.

3 Grant access within the transaction.

4 Grant access within the transaction.

5 Grant access within the transaction.

6 Review the results. Because you supply the session ID, the output reflects the uncommitted changes in your session.

7 After you review the output from the list-controls command, commit your changes.

8 If you are finished, it is a good practice to delete your session.

Example: Replace any direct access controls on tableA with access controls from an external JSON file. In this example, the replacement access controls are derived from tableB and are then applied to tableA.

```
1 sas-admin cas tables list-controls --server serverA --caslib caslibA --table tableB > ac.json

2 sas-admin cas tables replace-controls --server serverA --caslib caslibA --table tableA
 --source-file ac.json
```

```
3 sas-admin cas tables list-controls --server serverA --caslib caslibA --table tableA
```

1 This example writes the direct access controls for tableB to the file **ac.json** in the directory from which you are running your CLI.

   Note: This example assumes that the profile that you are using specifies **json** as your default output type. Otherwise, you must use the global option --output to specify **json** as the output type for this command. That option must immediately follow sas-admin.

   Note: You can reference an absolute path or a relative path.

2 Delete any direct access controls on tableA, and replace them with the access controls that you wrote to the ac.json file.

3 Review the new set of direct access controls on tableA.

Note: You can modify the output from tableB before you use it to replace direct access controls on tableA.

Details and Tips

Basics

- Throughout this topic, the term *access control* refers to an access control in the CAS authorization system. To manage access to content objects and functionality, see "CLI Examples: General Authorization" on page 26.
- You can add, delete, and replace only direct access controls.
- A request to delete a direct access control that does not exist does not generate an error.
- To modify access that a table inherits, set direct access controls on the parent caslib.
- You cannot modify access that a caslib inherits.
- Use of access control transactions is optional. You do not have to check out an object in order to modify its access controls.
- In the list-controls command, use the control-type and list-type options as follows:
  - Use the control-type option only if you want to obtain net access information (**effective**) or source information (**origins**).
  - The list-type options are not relevant if the control-type is **effective** or **origin**.
  - Use the list-type option only if you want to obtain inherited settings, in addition to direct access controls (**all**) or instead of direct access controls (**inherited**).
- You can obtain origins information for only one identity at a time.
- The value **serverA** is used in the examples for simplicity. A more typical server name is **cas-shared-default**.
- See "Details" on page 34 for information about using environment variables with the CAS commands.

Principals

- To specify a particular identity (where supported), you must provide a user ID or a group ID rather than a name.

  CAUTION! The user ID and the group ID that you provide are not validated. Make sure the IDs that you provide are accurate.

- To specify multiple users or multiple groups (where supported), use the pipe character (|) as a delimiter and enclose the string in single quotation marks (for example: --user 'userA|userB'). You cannot specify both users and groups in a single request.

- The group * corresponds to Authenticated Users. To specify that principal, enter `--group '*'`.
- The Guest principal represents all users who connect as guests. To specify that principal, enter the option `--guest` and do not specify a value.

Permissions

- To specify a particular permission (where supported), use one of the following case-insensitive values: ReadInfo, Select, LimitedPromote, Promote, CreateTable, DropTable, DeleteSource, Insert, Update, Delete, AlterTable, AlterCaslib, or ManageAccess. You cannot specify multiple permissions in a single request.
- For information about the scope and purpose of each permission, see "CAS Authorization: Concepts" in *SAS Viya Administration: Cloud Analytic Services Authorization*.

Fine-Grained Controls

- Row-level grants are always for the Select permission on a table. The syntax for row-level permission filters is the same as in other CAS authorization interfaces.
- You cannot set column-level permissions using this interface.

See Also

- "Command-Line Interface: Overview" on page 2
- SAS Viya Administration: Cloud Analytic Services Authorization

CLI Examples: General Authorization

The following examples assume that you have already signed in to SAS Viya at the command line. See "Command-Line Interface: Preliminary Instructions" on page 5.

Getting Access Information

Example: Show detailed properties of a specified rule.

```
sas-admin authorization show-rule --id d85144aa-79dc-4852-b949-645cc5ff8ffc --details
```

Example: Show effective access for a specified object URI. To return information about contributing rules, you must specify fulljson output in your profile. Specifying the **`output`** global option in the CLI command inline is insufficient.

```
sas-admin authorization explain --target-uri /SASHome/**
```

Example: List all the rules in the deployment.

```
sas-admin authorization list-rules
```

Managing Rules

Example: Give groupA Read access to reportA.

```
sas-admin authorization authorize --permissions Read --group groupA
--object-uri /reports/reports/33db163a-716e-4980-a5bc-6c42a0278c40
```

Example: Provide guest access to reportA.

```
sas-admin authorization authorize --permissions Read --guest
--object-uri /reports/reports/33db163a-716e-4980-a5bc-6c42a0278c40
```

Example: Grant Authenticated Users Read access to folderA and its child members.

```
sas-admin authorization authorize --permissions Read --authenticated-users
--object-uri /folders/folders/2414f911-d276-4357-8550-fcf03753c9e7/**
--container-uri /folders/folders/2414f911-d276-4357-8550-fcf03753c9e7
```

Example: Delete a rule.

```
1 sas-admin authorization show-rule --id d85144aa-79dc-4852-b949-645cc5ff8ffc --details
2 sas-admin authorization remove-rule --id d85144aa-79dc-4852-b949-645cc5ff8ffc
```

1 Review the rule's properties so that you are certain you are deleting the correct rule.
2 Delete the rule.

Example: Change the principal in an existing rule so that the rule is assigned to Group B, which has **groupB** as its ID.

```
1 sas-admin authorization show-rule --id cd75a376-c5d4-4951-9e57-cf441610628c --details
2 sas-admin authorization update-rule --id cd75a376-c5d4-4951-9e57-cf441610628c --group groupB
```

1 Review the rule's properties so that you are certain you are modifying the correct rule.
2 Modify the rule.

Example: Include the Update and Delete permissions in an existing rule that already grants the Read permission.

```
1 sas-admin authorization show-rule --id cd75a376-c5d4-4951-9e57-cf441610628c --details
2 sas-admin authorization update-rule --id cd75a376-c5d4-4951-9e57-cf441610628c
--grant --permissions Read,Update,Delete
```

1 Review the rule's properties so that you are certain you are modifying the correct rule.
2 Modify the rule.

Example: Edit the description in an existing rule.

```
1 sas-admin authorization show-rule --id 0e8a6ce7-e51a-40cc-aeda-ee2a5efb53ca --details
2 sas-admin authorization update-rule --id 0e8a6ce7-e51a-40cc-aeda-ee2a5efb53ca
--description 'This is a revised description.'
```

1 Review the rule's properties so that you are certain you are modifying the correct rule.
2 Modify the rule.

Managing Rules (Bulk Approach)

Adding Rules

Example: Add multiple rules, as specified in a referenced JSON file.

```
sas-admin authorization create-rules --file newrules.json
```

Note: The path to the input file is relative to the location of the CLI executable (sas-admin).

The input file is in JSON format. Here is an example that adds two rules:

```
[
   {
    "op": "add",
    "value": {
      "description": "Description for this rule.",
      "objectUri": "/folders/folders/156f833a-31ac-40f8-bf78-82e738daef36",
      "permissions": [
```

```
        "Secure"
      ],
      "principalType": "user",
      "principal": "userC",
      "type": "grant"
      }
   },
   {
    "op": "add",
    "value": {
      "description": "Description for this rule.",
      "objectUri": "/folders/folders/156f833a-31ac-40f8-bf78-82e738daef36",
      "permissions": [
       "Read",
       "Update",
       "Delete"
      ],
      "principalType": "group",
      "principal": "groupD",
      "type": "grant"
    }
   }
]
```

Because an ID for each rule is not specified in the preceding example, the rule ID is generated. For details about available parameters and values, see the Authorization REST API documentation on developer.sas.com.

Deleting Rules

Example: Delete multiple rules, as specified in a referenced JSON file.

```
sas-admin authorization remove-rules --file oldrules.json
```

Note: The path to the input file is relative to the location of the CLI executable (sas-admin).

The only required content for the remove-rules command is the ID of each rule that you want to remove. Here is an example of the input file for a remove-rules command:

```
[
   {
    "op": "add",
    "value": {
      "id": "rule123456788"
    }
   },
   {
    "op": "add",
    "value": {
      "id": "rule123456789"
    }
   }
]
```

Notice that even though this file is used in a removal command, the value of the op parameter is **add**.

> **TIP** If you specify a rule ID in an input file that you use to create rules, you can reference the same file to remove the rules. The remove-rules command ignores values other than the rule ID.

Details and Tips

- Throughout this topic, the term *rule* refers to an authorization rule in the general authorization system. To manage access to CAS objects (such as caslibs and tables), see "CLI Examples: CAS Authorization" on page 23.
- To assign a rule to a principal type, use one of the following options:

Principal Type	Option
Guest	`--guest`
Authenticated Users	`--authenticated-users`
Everyone	`--everyone`

- To assign a rule to a particular identity, you must provide a user ID or a group ID, not a name. For example, to assign a rule to the SAS Administrators custom group, specify: `--group SASAdministrators`

  CAUTION! The user ID and the group ID that you provide are not validated. Make sure the IDs that you provide are accurate.
- You can obtain the ID for a user or group from the **Users** page in SAS Environment Manager.
- You can obtain the objectURI for a content object (such as a report) from the **Content** page in SAS Environment Manager. Select the object in the navigation pane. On the right, the **URI** field in the **Basic Properties** section contains the object URI. To target the object URI for a content object (such as a report) or a container (such as a folder), append a suffix. See "Rule Targets" in *SAS Viya Administration: General Authorization*.
- You can obtain the ID for a rule from the **Rules** page in SAS Environment Manager. Right-click a rule and select **Properties**. The last field in the Properties window contains the rule's ID.
- When you use the show-rule command, always specify that you want details to be returned. Some of the fields that can be essential to interpreting a rule are excluded from the default response. For example, a condition is not included in the default response.
- When you use the update-rule command, specify only the options for the rule properties that you want to modify. For any option that you specify in the update-rule command, provide the complete replacement value or values.
- When you use the explain command, the returned information indicates the effective (net) access of each relevant principal for all permissions.

  Note: In the output from the explain command, the `grant` and `prohibit` values indicate effective (net) access, not direct settings. For example, a `prohibit` value in the output from the explain command is usually caused by the lack of any relevant grant, rather than by the existence of a relevant Prohibit rule. See "Authorization Decisions" in *SAS Viya Administration: Cloud Analytic Services Authorization*.
- Enabling or disabling guest access involves more than running the enable-guest-access or disable-guest-access command. See the guest access documentation.

See Also

- "Command-Line Interface: Overview" on page 2
- SAS Viya Administration: General Authorization

CLI Examples: Backup

The following examples assume that you have already signed in to SAS Viya at the command line. See "Command-Line Interface: Preliminary Instructions" on page 5.

Examples

Example: List the first 50 backup jobs.

```
sas-admin backup list --limit 50
```

Example: Start a binary backup named backupA.

```
sas-admin backup start --slug backupA
```

See Also

- "Overview" in *SAS Viya Administration: Backup and Restore*
- "Command-Line Interface: Overview" on page 2

CLI Examples: CAS Administration

The following examples assume that you have already signed in to SAS Viya at the command line. See "Command-Line Interface: Preliminary Instructions" on page 5.

Important: For Windows users of the CAS CLI who are using it for CAS administration and who are CAS administrators, CAS sessions are launched under the CAS service account. Also, any user who has credentials that are stored can use those credentials to run the CAS CLI for any purpose. See "External Credentials: How To" in *SAS Viya Administration: External Credentials*.

Generate CAS Samples

Example: Generate sample template files for creating caslibs, user-defined formats, and resource management policies.

```
sas-admin cas generate-cas-samples --output-location sample-location
```

Note: The **`cas generate-cas-samples`** command also generates an md5.txt file that contains checksums for each sample file. SAS Technical Support can use this file to validate the integrity of the sample files.

See Also

"Source Files: JSON Templates " on page 16

Enable Guest Access

Note: This information does not apply to Windows environments or to Linux environments that have been configured for Kerberos.

To enable guest access on the specified server, modify the direct access controls for the predefined caslibs on the server. To do this, use the controls that are defined in the specified source file. If you do not have a preexisting source file, you can generate one that contains the default access controls, make modifications to it, and use it as the source file.

Perform this action with elevated privileges if the user is able to assume the Superuser role. This is one of several required steps to enable guest access. For a complete set of instructions on how to enable guest access, see "Authentication: Guest Access" in *SAS Viya Administration: Authentication*.

Example: Enable guest access for the predefined caslibs on the server by using a preexisting source file.

```
sas-admin cas facilitate-guest --source-file path-to-source-file --server serverA --superuser
```

Example: Enable guest access for the predefined caslibs on the server by creating a source file and using it to modify the direct access controls.

```
1 sas-admin cas generate-guest-controls --output-location /path/

2 sas-admin cas facilitate-guest --source-file path-to-controls-file --server serverA --superuser
```

1 Generate a source file from the default access controls. The generated source file is named **facilitate-guest-controls.txt**.

   Make the desired modifications to access controls to the source file that you just generated.

2 Modify the direct access controls using the source file that you just modified.

Note: You can also remove guest access with the **cas remove-guest-controls** command. However, this command removes only the default set of direct access controls and not any other guest access controls that you might have applied. To view which direct access controls are removed, see the **facilitate-guest-controls.txt** file that is generated by the **cas generate-guest-controls** command.

Manage CAS Role Memberships

Example: List the administrative users on the specified CAS server.

```
sas-admin cas admin-users list --server serverA
```

Example: Add the user user1 to the Superuser role on the specified CAS server.

```
sas-admin cas admin-users add --user user1 --server serverA
```

Example: Remove the group with the ID group1 and name group1_name from the Superuser role on the specified CAS server. The action is performed without prompting the user for confirmation since the **force** option is used.

```
sas-admin cas admin-users remove --group group1 --server serverA --name group1_name --force
```

> **TIP** To remove a user or a group from the administrative users, you must specify the name of the user or the group as well as the identity of the user or the group. Use the name option to specify the name. Use the user option to specify the ID for a user. Use the group option to specify the ID for a group. In order to obtain the ID, use the **admin-users list** command.

Manage SAS Sessions

Example: List the sessions of which you are the owner on the specified CAS server.

```
sas-admin cas sessions list --server serverA
```

Example: Using elevated privileges, list 50 sessions for which the owner is user1 on the specified CAS server, and sort by **state** . You must be able to assume the Superuser role to run the command with elevated privileges.

```
sas-admin cas sessions list --server serverA --superuser --limit 50 --owner user1 --sort-by state
```

Example: Using elevated privileges, list all sessions for which the name contains the string dataExplorer on the specified CAS server. You must be able to assume the Superuser role to run the command with elevated privileges.

```
sas-admin cas sessions list --server serverA --superuser --all --name-contains dataExplorer
```

Example: Using elevated privileges, show additional information about the session with ID 12345 on the specified CAS server. You must be able to assume the Superuser role to run the command with elevated privileges.

```
sas-admin cas sessions show-info --superuser --session-id 12345  --server serverA
```

Manage Tables

Note: These examples explicitly specify the **server** and **caslib** required options. However, using environment variables might be more efficient for these options. For more information about the environment variables, see "Details" on page 34.

Example: For the specified caslib and CAS server, list all tables with names that contain the string visual, sort by **state**, and return a maximum number of 50 tables.

```
1 sas-admin cas help tables

2 sas-admin cas tables help list

3 sas-admin cas tables list --caslib caslibA --server serverA --name-contains visual
--sort-by state --limit 50
```

1. Review the Help for the cas tables command.
2. Review the Help for the list subcommand of the cas tables command.
3. Issue the command to list the tables for the specified caslib and CAS server with the appropriate subcommand and options.

Example: Load the given table for the specified caslib and CAS server.

```
sas-admin cas tables load --table airlines --server serverA --caslib caslibA
```

Example: Unload the given table for the specified caslib and CAS server.

```
sas-admin cas tables unload --table airlines --server serverA --caslib caslibA
```

Example: Show information about the given table for the specified caslib and CAS server.

```
sas-admin cas tables show-info --table airlines --server serverA --caslib caslibA
```

Manage Caslibs

Important: The caslibs command prompts for the required options for the different types of caslibs that you can create. For information about the available options for each type of caslib, see the **dataSource** option for the **addCaslib** action here: addCaslib Action. Be aware that there are more required options for the cas caslibs CLI command than there are for the addCaslib action set.

Note: These examples explicitly specify the **server** required option. However, using an environment variable might be more efficient for this option. For more information about the environment variables, see "Details" on page 34.

Example: Generate sample template files for creating caslibs.

```
sas-admin cas generate-cas-samples --output-location /sample-template-path
```

Example: On the specified server, create a caslib that is based on a file path.

```
sas-admin cas caslibs create path --name caslibA --path /tmp/dept --server serverA
```

Example: On the specified server, create a caslib that is based on an instance of an Impala server.

```
sas-admin cas caslibs create impala --authentication-domain domain  --server serverA
--name caslibA --schema schema --impala-server Impala-server
```

Example: On the specified server, create a caslib that is based on an instance of a Postgres server.

```
sas-admin cas caslibs create postgres --authentication-domain domain --server serverA
 --name caslibA --postgres-server Postgres-server --postgres-database Postgres-database
```

Example: On the specified server, list the first 20 global caslibs. You must be able to assume the Superuser role to run the command with elevated privileges.

```
sas-admin cas caslibs list --server serverA --scope global --limit 20 --superuser
```

Example: On the specified server, list the global caslibs starting at caslib number 21. You must be able to assume the Superuser role to run the command with elevated privileges.

```
sas-admin cas caslibs list --server serverA --scope global --start 21 --superuser
```

Example: Delete the given caslib from the specified server.

```
sas-admin cas caslibs delete --server serverA --name caslibA
```

See Also

"Create Caslibs from JSON Templates" on page 16

Manage Servers

Note: These examples explicitly specify the **`server`** required option when applicable. However, using an environment variable might be more efficient than this option. For more information about the environment variables, see "Details" on page 34.

Example: List all the CAS servers.

```
sas-admin cas servers list --all
```

Example: Adds multiple paths to the paths list on the specified server so that a caslib that is based on the paths can be created. The active list in this example is the whitelist. You can specify multiple paths in the file that is referenced in the **`source-file`** option. See caslib paths list for more information.

```
sas-admin cas servers paths-list add-paths --server serverA --source-file /path-to-source-file/source.txt
```

Example: On the specified server, list the identities who can create and delete session and global caslibs.

```
sas-admin cas servers privileges list --server serverA
```

Example: On the specified server, grant the ability to manage global caslibs to a group.

Note: A reference to the group * corresponds to Authenticated Users. To specify the Authenticated Users group, enter `--group '*'`

```
sas-admin cas servers privileges modify --server serverA --grant --global --group group-name
```

Example: Show detailed information about the specified server. Show the dates and times in the local time zone.

```
sas-admin cas servers show-info --server serverA --use-local-datetime
```

Example: Create a new resource management policy for a global caslib. You can also create a global caslib policy from a JSON template on page 18.

```
sas-admin cas servers policies define global-caslibs
--attributes cpu:30,Public:1000000000,HPS:4000000000 --server serverA
```

Example: Create a new resource management policy for priority assignments. You can also create a priority assignment policy from a JSON template on page 19.

```
sas-admin cas servers policies define priority-assignments
 --attributes userA:4,userB:3 --server serverA
```

Example: Create a new resource management policy for priority level 5 on serverA. This creates a policy with the name **`serverA-priority-5`**. You can also create a priority level policy from a JSON template on page 20.

```
sas-admin cas servers policies define priority-levels --cpu 10 --global-casuser 500000000
--global-casuser-hdfs 100000000 --session-tables 2500000000 --priority 5  --server serverA
```

Example: Create a new resource management policy for priority level 4 on serverA. Specify a source file that contains values for priority level 3, but override them with the values for priority 4 by including them on the command line. This example overrides the value for CPU in the source file, by specifying the **cpu** option on the command line. This example assumes that you have a preexisting JSON file for a priority level 3 policy named priority3-levels.json. You can also create a priority level policy from a JSON template on page 20.

```
sas-admin cas servers policies define priority-levels --priority 4 --server serverA --cpu 15
 --source-file /path/priority3-levels.json
```

Note:

- For CAS to use resource management policies, the environment variable **CAS_ENABLE_CONSUL_RESOURCE_MANAGEMENT** must be set on the CAS server. For more information, see Environment Variables.
- When you create a resource management policy for priority levels, the policy is named ***CAS-server-priority-priority-level***.
- When you create a resource management policy for priority levels and you use the **source-file** option to specify values, you must include the **priority** option on the command line.
- CPU sharing is not supported on Windows. As a result, when creating a resource management policy for priority levels on Windows, the **cpu** option is allowed, but will not be honored.
- The values that you assign to the **global-casuser**, **global-casuser-hdfs**, and **local-tables** attributes must be specified in bytes.
- Options that are specified the command line take precedence over the options in the source file.

See Also

- “Create Policies from JSON Templates” on page 18
- “CAS Resource Management Policies” in *SAS Viya Administration: SAS Cloud Analytic Services*

Details

- When running the CAS CLI on a UNIX machine or a Macintosh machine, with the CAS server running on a Windows machine, a Windows pathname that contains backslashes must be enclosed in single quotation marks. Here is an example: **'\\tmp\sas'**.
- The CAS CLI supports the following environment variables:
  - SAS_CLI_DEFAULT_CAS_SERVER
  - SAS_CLI_DEFAULT_CASLIB
  - SAS_CLI_DEFAULT_CAS_SESSION

  You can assign values to the environment variables that you want to remain in effect throughout your session. If the CAS CLI command requires the **server**, **caslib**, or **session-id** options, and the environment variables are set, then you can omit the required options from the CAS CLI command.

  For example, suppose the following:

  - **SAS_CLI_DEFAULT_CAS_SERVER** is set to **serverA**.
  - **SAS_CLI_DEFAULT_CASLIB** is set to **caslibA**.

  You can then run this command without specifying the required **server** and **caslib** options: `./sas-admin cas tables show-info --table airlines`.

Note: Some commands do not support the use of some of the environment variables. For example, the CAS CLI ignores the CAS environment variables for the following commands:

- caslib remove-control
- caslib delete
- tables remove-control
- sessions delete

You must explicitly specify all required options when using these commands.

See Also

- "Command-Line Interface: Overview" on page 2
- SAS Viya Administration: SAS Cloud Analytic Services
- SAS Viya Administration: Identity Management
- SAS Viya Administration: Data

CLI Examples: Compute

The following examples assume that you have already signed in to SAS Viya at the command line. See "Command-Line Interface: Preliminary Instructions" on page 5.

Examples

Example: List the compute contexts.

```
sas-admin compute contexts list
```

Example: Validate the compute context session with the specified ID.

```
sas-admin compute contexts validate --id 389fee7a-e164-4e45-b836-a301638e9945
```

Example: Delete the compute context session with the specified name.

```
sas-admin compute contexts delete --name "SAS Job Execution compute context"
```

Example: List the launcher contexts.

```
sas-admin compute launchers list
```

Example: Delete the launcher context with the specified ID.

```
sas-admin compute launchers delete --id 8fbdd5f8-a2ee-42a5-a228-8737a0cf778f
```

Example: List the compute sessions.

```
sas-admin compute sessions list
```

See Also

"Command-Line Interface: Overview" on page 2

CLI Examples: Configuration

The following examples assume that you have already signed in to SAS Viya at the command line. See "Command-Line Interface: Preliminary Instructions" on page 5.

Examples

Example: List all the configurations that exist in the Configuration service.

```
sas-admin configuration configurations list
```

Example: Download the configurations with the specified definition name (**spring** in this example) from the SASLogon service, and write the output to a file.

```
sas-admin configuration configurations download --definition-name spring --service SASLogon
--target path-to-output-file
```

Example: List the expectation objects in the Configuration service.

```
sas-admin configuration expectations list
```

Example: Show the expectation with the specified ID.

```
sas-admin configuration expectations show --id 185a046e-4e88-4e29-86cf-61d04b9abd07
```

Details

- You can delete a configuration with the configuration CLI.

  CAUTION! Do not use the **delete** subcommand of the configurations command unless you are sure that you want to delete your configuration.

See Also

"Command-Line Interface: Overview" on page 2

CLI Examples: Data Explorer

The following examples assume that you have already signed in to SAS Viya at the command line. See "Command-Line Interface: Preliminary Instructions" on page 5.

Examples

Example: List the data sources.

```
sas-admin dataexplorer data-sources list
```

Example: List the connections for the specified data source.

```
sas-admin dataexplorer data-sources connections list --source dsName
```

Example: List the tables for the specified data source and connection.

```
sas-admin dataexplorer data-sources tables list --source dsName --connection connection
```

Example: List the rows for the specified table for the specified data source and connection.

```
sas-admin dataexplorer data-sources rows list --source dsName --connection connection --table tableName
```

Example: Delete a table from the specified data source and connection.

```
sas-admin dataexplorer data-sources tables delete --source dsName --connection connection --table tableName
```

Example: Create a new table from the specified data source and connection.

```
sas-admin dataexplorer data-sources tables create --target-data-source dsName
--target-connection connection --target-table-name tableName --source-data-source dsName
--source-connection connection --source-table-path path-to-table
```

Example: Create a new table from a local file.

```
sas-admin dataexplorer data-sources tables create --target-data-source dsName
--target-connection connection --target-table-name tableName --local-file path-to-file
```

Example: Create a new table from a table URI.

```
sas-admin dataexplorer data-sources tables create --target-data-source dsName
--target-connection connection --target-table-name tableName
--source-table-uri uri-for-table
```

Example: Query for the specified job.

```
sas-admin dataexplorer jobs query --job-id jobID
```

See Also

- "Command-Line Interface: Overview" on page 2
- SAS Data Explorer: User's Guide

CLI Examples: Device Management

The following examples assume that you have already signed in to SAS Viya at the command line. See "Command-Line Interface: Preliminary Instructions" on page 5.

Examples

Example: Determine whether the device with ID device1 is authorized for use in the environment.

```
sas-admin devices authorized-devices validate --device-id device1
```

Example: Show whether the blacklist or whitelist is being enforced.

```
sas-admin devices enforcement status
```

Example: Add the device with ID device1 to the whitelist.

```
sas-admin devices whitelist add --device-id device1
```

Example: List the devices that are enabled in the whitelist.

```
sas-admin devices whitelist list
```

Example: Add the device with ID device1 to the blacklist.

```
sas-admin devices blacklist add --device-id device1
```

Example: Remove the device with ID device1 from the blacklist.

```
sas-admin devices blacklist delete --device-id device1
```

Example: From a list of the devices of type iPhone that have connected or attempted to connect to the server, add a specific iPhone to the blacklist.

```
1 sas-admin devices last-access list --device-type iPhone

2 sas-admin devices blacklist add --device-id device-id
```

1 List the last-access attempts of all devices of type iPhone.

2 Add a device to the blacklist using the device ID that was identified in the previous step.

Example: List in fulljson output the last access attempts to the server for all devices.

```
sas-admin --output fulljson devices last-access list
```

See Also

- "Command-Line Interface: Overview" on page 2
- SAS Viya Administration: Mobile

CLI Examples: Folders

The following examples assume that you have already signed in to SAS Viya at the command line. See "Command-Line Interface: Preliminary Instructions" on page 5.

Examples

Example: Add subfolderA as a child folder to folderA. In this example, the path to folderA is **`/parent-folder/folderA`**.

```
sas-admin folders create --name subfolderA --parent-path /parent-folder/folderA
```

Example: Update the name of folderA to departmentA. In this example, the path to folderA is **`/parent-folder/folderA`**.

```
sas-admin folders update --path /parent-folder/folderA --name departmentA
```

Example: Delete the departmentA folder including any non-folder members. In this example, the path to departmentA is **`/parent-folder/departmentA`**.

```
sas-admin folders delete --path /parent-folder/departmentA --recursive
```

Example: Move folderA to folderB. In this example, the path to folderA is **`/parent-folder/folderA`**, and the path to folderB is **`/parent-folder/folderB`**.

```
3 sas-admin folders move --path /parent-folder/folderA --parent-path /parent-folder/folderB

4 sas-admin folders list-members --path /parent-folder/folderB

5 sas-admin folders list-members --path /parent-folder/folderB --tree
```

1 Move folderA to folderB.

2 List the members of folderB to confirm that folderA is a member.

3 Display the members of folderB in a visual tree structure using the **`tree`** option.

Example: List the members of folderA using the ID of folderA.

```
sas-admin folders list-members --id folderA-ID
```

Example: Create folderA. Add subfolderA as a child folder to folderA using the ID of folderA.

```
1 sas-admin folders create --name folderA

2 sas-admin folders create --name subfolderA --parent-id parent-folder-ID
```

1 Create folderA and note the ID.

2 Issue the command to create subfolderA and specify the ID of folderA as the parent folder.

Details

- Many of the folders commands require the ID of a folder as an argument. The ID of a folder is displayed when you create the folder. The ID of folders is also displayed when you list folders.
- When you delete a folder that contains non-folder content (such as reports), the CLI displays a message that you cannot delete the folder, because it is not empty. To delete all contents of folders including non-folder content, you must use the **`recursive`** option of the **`delete`** subcommand.

See Also

- “Command-Line Interface: Overview” on page 2
- SAS Viya Administration: Content Management

CLI Examples: Fonts

The following examples assume that you have already signed in to SAS Viya at the command line. See “Command-Line Interface: Preliminary Instructions” on page 5.

Examples

Example: Add and register the open-source OpenSans-Bold.tff Google font.

```
sas-admin fonts add --uri https://server:port/fonts/OpenSans-Bold.tff
```

Note: In a Windows environment, to add a font with a file uniform resource indicator (URI), copy the file to the Windows server and then add the font to the environment with this command: `sas-admin fonts add --uri file://hostname/path-to-font`

Example: List the currently registered fonts in the system.

```
sas-admin fonts list
```

Example: Show information about the Arial Symbol font.

```
1 sas-admin fonts list --name "Arial Symbol"
2 sas-admin fonts show-info --id font-ID-of-Arial-Symbol-font
```

1 Identify the ID of the Arial Symbol font.

2 Show information about the Arial Symbol font using the specified font ID.

Example: Delete fontA from the system.

```
1 sas-admin fonts list --name fontA
2 sas-admin fonts delete --file-id file-id of fontA
```

1 Identify the file ID of fontA.

2 Delete fontA from the system using the specified file ID.

Details

- The fonts CLI can be used to add web open font format (WOFF), TrueType (TTF), and TrueType Collection (TTC) fonts to the Fonts service. Once added, the fonts are available for the following purposes:
  - To render content in web browsers with web open font format (WOFF) and TrueType (TTF) fonts.

    TrueType Collection (TTC) fonts are not displayed in web browsers. Therefore, users cannot select text that uses TrueType Collection (TTC) fonts if they are included in a SAS web application such as SAS Visual Analytics.
  - To render contents for printing with TrueType (TTF) and TrueType Collection (TTC) fonts.

    Web open font format (WOFF) fonts are not supported for printing PDF or SVG output. Therefore, WOFF fonts must be paired with a matching TTF or TTC font for print support.

  Note: The following SAS system fonts are WOFF only, and are not supported for printing:

  - Noto Sans
  - Noto Sans JP

- Noto Sans KR
- Noto Sans SC
- Noto Sans TC
- Noto Sans Thai

Given the TTC and WOFF font limitations, you might need to upload a combination of font formats in order to satisfy both of the preceding purposes.

- A font might not be available from a web server that is accessible from the middle tier without authentication. If so, you can upload the font to the SAS Viya server and reference it with the file:/// URI scheme.
- In multi-tenant environments, the fonts are shared across all the tenants. Maintenance is supported only on the provider tenant.
- You are responsible for obtaining licensing of any fonts that are registered with the system. In a multi-tenancy configuration, this includes licensing the fonts for use by all tenants.

See Also

"Command-Line Interface: Overview" on page 2

CLI Examples: Formats

The following examples assume that you have already signed in to SAS Viya at the command line. See "Command-Line Interface: Preliminary Instructions" on page 5.

Creating Format Libraries

Note: By default, the tables for newly created format libraries are stored in the **FORMATS** library. You can specify a different caslib to contain the tables with the **format-libraries create --caslib** option.

Example: Create a format library from a SAS catalog.

```
sas-admin cas format-libraries create --format-library fmtlib1 --server serverA
 --source-path /path-to-catalog/catalog-name.sas7bcat --su
```

Example: Create a format library from a CAS table.

```
sas-admin cas format-libraries create --format-library fmtlib1 --server serverA
 --source-path /path-to-table/table-name.sashdat --su
```

Example: Create a format library from a CAS item store.

```
sas-admin cas format-libraries create --format-library fmtlib1 --server serverA
 --source-path /path-to-table/item-store-name --su
```

Creating Formats

Note: In the following examples, a dollar sign ($) is used to indicate a character format. SAS character format names must be quoted.

Example: Generate sample templates for creating user-defined formats.

```
sas-admin cas generate-cas-samples --output-location /sample-template-path
```

Example: Create a user-defined format from a JSON template. For more information, see "Source Files: JSON Templates " on page 16. This example assumes that you already have a JSON template file, as follows:

```
{
    "libraryName": "userformats1",
    "locale": "en_US",
```

```
    "name": "$ynm",
    "ranges": [
        "1=yes",
        "2=no",
        "3=maybe"
    ],
    "version": 1
}
```

```
1 sas-admin cas format-libraries add-format json --server serverA --source-file /path-to-template/ynm.json

2 sas-admin cas format-libraries show-format-ranges --server serverA
 --format '$ynm' --format-library userformats1
```

1 Create the ynm format from the JSON template. Since the **format** and **format-library** options are not included on the command, the library and name values in the template are used.

2 Display information about the ynm format to verify that the userformats1 library and ynm name from the template were used.

Example: Create a user-defined format from a JSON template, but specify a different format and format library on the command line. For more information, see “Source Files: JSON Templates ” on page 16. This example uses the JSON template from the previous example.

```
1 sas-admin cas format-libraries add-format help json

2 sas-admin cas format-libraries add-format json --server serverA
 --format '$ynmA' --format-library userformats2 --source-file /path-to-template/ynm.json

3 sas-admin cas format-libraries show-format-ranges --server serverA
 --format '$ynmA' --format-library userformats2
```

1 Review the options that are available for the **json** subcommand. Since you are specifying a format and format library that is different from those that were specified in the JSON template, you will use the **format** and **server** options on the command line.

2 Create the ynmA format in the userformats2 library using the JSON template. Specify the ynmA format and userformats2 format library on the command line with the **format** and **format-library** options. These options override the options that are specified in the JSON template.

3 Display information about the format to verify that the userformats2 library and ynmA name from the command line overwrote the values in the template.

For more information, see “Create User-Defined Formats from JSON Templates ” on page 17.

Example: Create a user-defined format and specify the ranges from a CSV template. For more information, see “Source Files: Data” on page 21. This example assumes that you already have a CSV template file, as follows:

```
F,female
M,male
f,female
m,male
```

```
1 sas-admin cas format-libraries add-format help csv

2 sas-admin cas format-libraries add-format csv --server serverA --format '$gender'
 --format-library userformats1 --source-file /path-to-template/gender.csv

3 sas-admin cas format-libraries show-format-ranges --server serverA --format '$gender'
 --format-library userformats1
```

1 Review the options that are available for the **csv** subcommand.

2 Create the gender format in the userformats1 library using the CSV template to specify the ranges. Since the gender format is a character format, you must precede it with a dollar sign ($) and enclose it in quotation marks. The other command options must be specified on the command line.

3 Display information about the format.

Example: Create the same user-defined format for different locales. Use the **show-format-ranges** option with the formats that you created. This example uses the CSV file from the previous example.

```
1 sas-admin cas format-libraries add-format csv --server serverA --format '$gender'
 --format-library userformats1 --source-file /path-to-template/gender.csv

2 sas-admin cas format-libraries add-format csv --server serverA --format '$gender'
 --format-library userformats1 --source-file /path-to-template/gender.csv --format-locale ja_JP

3 sas-admin cas format-libraries add-format csv --server serverA --format '$gender'
 --format-library userformats1 --source-file /path-to-template/gender.csv --format-locale ko_KR

4 sas-admin cas format-libraries show-formats --server serverA --format-library userformats1

5 sas-admin cas format-libraries show-format-ranges--server serverA
--format-library userformats1 --format '$gender'

6 sas-admin cas format-libraries show-format-ranges --server serverA
--format-library userformats1 --format '$gender' --ignore-locale

7 sas-admin cas format-libraries show-format-ranges --server serverA
--format-library userformats1 --format 'ja_JP-$gender'
```

1 Create the gender format in the userformats1 library with no specified locale. Since the gender format is a character format, you must precede it with a dollar sign ($). Since no locale is specified, the format will be created with no locale.

2 Create another gender format in the userformats1 library, but specify the Japanese locale.

3 Create another gender format in the userformats1 library, but specify the Korean locale.

4 List the SAS formats in the userformats1 library to verify that the formats were created. You should see the following formats: $gender, ja_jp-$gender, and ko_kr-$gender.

5 Display information about the $gender format that was created with no specified locale. The locale defaults to the operating system of the CAS server. If the CAS server is physically located in Japan and the default locale has not been changed, the locale will default to Japanese. If the CAS server is physically located in Korea and the default locale has not been changed, the locale will default to Korean. If the CAS server is physically located in the United States and the default locale has not been changed, the locale will default to English.

6 Display information about the $gender format that was created with no specified locale, but specify the **ignore-locale** option. You will see that no locale is displayed.

7 Display information about the ja_JP-$gender format that was created with the Japanese locale. The Japanese locale is displayed, and takes precedence over the locale of the operating system of the CAS server.

Example: Create a user-defined format. Then save the format library that contains the new format as a SASHDAT file (table) in the data source that is associated with the active caslib. You can load a format library from the table in future sessions. This example assumes that you already have a CSV template file, as follows:

```
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"

1 sas-admin cas caslibs create path --name caslibA --server serverA --path /path-to-caslib
```

```
2 sas-admin cas format-libraries add-format csv --format '$enginesize' --format-library Casformats
 --server serverA --source-file /path-to-source-file/enginesize.csv

3 sas-admin cas format-libraries export --caslib caslibA --format-library Casformats
 --server serverA --table enginefmt --su

4 cas tables list --caslib caslibA
```

1 Create a path caslib called caslibA to contain the new format. The SASHDAT file that is created will be placed in the location that you specify in the **--path** option.

2 Create a new character format called enginesize by importing its ranges from a CSV file. Add the new format to the Casformats library.

3 Export the Casformats library that contains the enginesize format to the table enginefmt in caslibA. The **--table** option specifies the name of the table and the results in a file named enginefmt.sashdat. The file is placed in the location that you specified in the **--path** option when you created caslibA. You must specify the superuser option in commands where elevated privileges are needed.

4 List the tables in caslibA to verify that you see the enginefmt table.

Details and Tips

- You can create a SAS format library from an existing SAS catalog, CAS table, or item store file. The extension of the file that you specify in the **source-path** option of the **cas format-libraries create** command identifies the source of the SAS format library as follows:

Extension	Source of the Format Library
.sas7bcat	SAS catalog
.sashdat	CAS table
no extension or any other extension	CAS item store file

- If you run the show-format-ranges command for a format, the results are displayed according to these rules:
  - If you request information about a non-locale format (such as **$charfmt**) and you do not specify the **ignore-locale** option, the default locale format will be returned. If the default system locale is en_US and the **ignore-locale** option was not specified, the returned format for **$charfmt** is en_US-$charfmt.
  - If you request information about a non-locale format (such as **$charfmt**), you must specify the **ignore-locale** option in order to return the actual format with no locale. If the **ignore-locale** option was specified, the returned format for **$charfmt** is $charfmt.
  - If you request information about a format that has a non-existent locale, the format for the default system locale is returned. Suppose that you request information about en_bogus-$charfmt, which is a format that does not exist. If the default system locale is en_US, the returned format is **en_US-$charfmt**.
  - If you request information about an invalid format, an error is returned. Suppose that you request information about en-US-$charfmt. The hyphen in the locale name (en-US) should have been an underscore (en_US). No format is returned, and an error is displayed.
- When you create a format library, the search order defaults to none. If you want to specify how the new format library appears in the SAS format search order, you must do the following:
  - specify one of the following values for the **search-order** option in the **format-libraries create** command: append, prepend, or replace.

- specify the **superuser** option in the **format-libraries create** command.

See Also

See "Overview" in *SAS Viya Administration: Data* for more information about user-defined formats.

CLI Examples: Identities

The following examples assume that you have already signed in to SAS Viya at the command line. See "Command-Line Interface: Preliminary Instructions" on page 5.

Examples

Example: Add user1 to the group that has the ID 4444.

```
sas-admin identities add-member --user-member-id  user1 --group-id 4444
```

Example: Create a group with the name Salesgroup, the group ID 8888, and the description of "Custom sales group".

```
sas-admin identities create-group --id 8888 --name Salesgroup --description "Custom sales group"
```

Example: Remove user1 from the sales and marketing groups.

```
1 sas-admin identities list-memberships --user-id user1
2 sas-admin identities remove-member --group-id sales-group --user-member-id user1
3 sas-admin identities remove-member --group-id marketing-group --user-member-id user1
```

1 Verify the groups that user1 belongs to.

2 Remove user1 from the group that has the group ID sales-group.

3 Remove user1 from the group that has the group ID marketing-group.

Example: Show details about the group that has the group ID ABC.

```
sas-admin identities show-group --id ABC
```

Example: Validate the LDAP configuration values.

Note: These values are not used for setting up LDAP.

You can use any of the following commands to validate the LDAP configuration values:

Note: The identities validate-config command is available beginning with SAS Viya 3.4 in July 2019 and later deployments.

```
1 sas-admin identities validate-config --ldap-url "url" --ldap-userDN "userDN"
--ldap-password password --ldap-group-baseDN "group-baseDN"
 --ldap-user-baseDN "user-baseDN"
2 sas-admin identities validate-config --file path-to-file
3 sas-admin identities validate-config  --ldap-userDN "userDN" --ldap-password password
 --file path-to-file
```

1 Specify the LDAP configuration values via the command line.

2 Specify a file that contains the LDAP configuration values.

Note: You can include the LDAP configuration values in a separate file by using the file option. The file that you specify can be of the type JSON or YML.

The password that is used to connect to the LDAP server must be Base64-encoded. If you specify a password in the file, you must explicitly encode the password.

3 Specify a file that contains the LDAP configuration values. Use the ldap-userDN and ldap-password options to specify a different LDAP user and password to override these values from the file. If a command contains LDAP options and the file option, the LDAP option values override the values in the file.

Note: You can include the LDAP configuration values in a separate file by using the file option. The file that you specify can be of the type JSON or YML.

The password that is used to connect to the LDAP server must be Base64-encoded. If you specify a password in the file, you must explicitly encode the password.

Details

- If a group is created with no name, the specified ID is used for the name.
- The following identities commands list only 50 items at a time by default:
  - list-groups
  - list-members
  - list-memberships

  To list more than 50 items, you can use the `--limit` option.

  Example:

  ```
  sas-admin identities list-members --group-id ABCD --limit 100
  ```

- Consider the following key points about the identities validate-config command:
  - You can use this command only on a secure (TLS-enabled) environment.
  - The password that is used to connect to the LDAP server must be Base64-encoded. If you specify the password with the **`--ldap-password`** option, the CLI encodes it for you. If you specify the password in the JSON or YML file, you must encode the password explicitly.
  - The file that you specify with the **`--file`** option can be of the type JSON or YML.
  - Here is a typical file of the type YML that is used for validating your LDAP configuration values:

    Note: Enter the URL definition on a single line. Multiple lines are used here to improve readability.

    ```
    sas.identities.providers.ldap.connection:
        host: multi-tenant-ldap.sun.com
        password: xxxxxxxxxxx=
        port: 389
        url: ldap://${sas.identities.providers.ldap.connection.host}:
            ${sas.identities.providers.ldap.connection.port}
        userDN: cn=admin,dc=sun,dc=com
    sas.identities.providers.ldap.group:
        baseDN: DC=SUN,DC=com
        accountId: cn
        objectClass: groupOfUniqueNames
        objectFilter: (objectClass=groupOfUniqueNames)
    sas.identities.providers.ldap.user:
        baseDN: DC=SUN,DC=com
        accountId: uid
        objectClass: inetOrgPerson
        objectFilter: (objectClass=inetOrgPerson)
    ```

See Also

- “Command-Line Interface: Overview” on page 2

- "Identity Management: Overview" in *SAS Viya Administration: Identity Management*

CLI Examples: Job

The following examples assume that you have already signed in to SAS Viya at the command line. See "Command-Line Interface: Preliminary Instructions" on page 5.

Examples

Example: List the job flows.

```
sas-admin job flows list
```

Example: Generate a template for a flow and write the output to a file.

```
sas-admin job flows generate-template --file-out /tmp/output.txt
```

Example: List job flow scheduling service objects.

```
sas-admin job schedulers list
```

Details

Here is an example of a template file that is generated from a flow that you created:

```
{
    "name": "Replace with name of the flow",
    "description": "(Optional) Replace with description of the flow",
    "triggerType": "Replace with trigger type, select one of: runnow, manual, event",
    "triggerCondition": "Replace with either any or all",
    "flowProperties": {},
    "defaultJobProperties": {}
}
```

See Also

"Command-Line Interface: Overview" on page 2

CLI Examples: Launcher

The following examples assume that you have already signed in to SAS Viya using the command line. See "Command-Line Interface: Preliminary Instructions" on page 5.

Examples

Example: List all launcher contexts.

```
sas-admin launcher contexts list --all
```

Example: Show detailed information about the SAS Launcher context with the specified ID.

```
sas-admin launcher contexts show --id context-id
```

Example: List the SAS Launcher options set mappings.

```
sas-admin launcher options-set-mappings list --all
```

Example: Show detailed information about the SAS Launcher options that are set with the specified ID.

```
sas-admin launcher options-sets show --id options-set-ID
```

See Also

- “Command-Line Interface: Overview” on page 2
- “SAS Launcher Server and Launcher Service” in *SAS Viya Administration: Programming Run-Time Servers*

CLI Examples: Licensing

The following examples assume that you have already signed in to SAS Viya at the command line. See “Command-Line Interface: Preliminary Instructions” on page 5.

Examples

Example: List site information, which is the site name, the site number, the operating system name, the release number, the server date, the grace period, and the warning period.

```
sas-admin licenses site-info list
```

Example: List all products in the system whose name contains “Visual Analytics”.

```
sas-admin licenses products list --name-contains "Visual Analytics"
```

Example: List all products in the system that are expired.

```
sas-admin licenses products list --expired
```

Example: List the products in the system that have these identifiers: 827, 921, and 985.

```
sas-admin licenses products list --product-ids 827,921,985
```

Example: List the number of products with a current license that are deployed.

```
sas-admin licenses count --current
```

Example: List the Data-Connector products whose licenses are covered within the grace period.

```
sas-admin licenses data-connectors list --grace
```

See Also

- “Command-Line Interface: Overview” on page 2
- SAS Viya Administration: Licensing

CLI Examples:Quality Knowledge Bases (QKBs)

The following examples assume that you have already signed in to SAS Viya at the command line. See “Command-Line Interface: Preliminary Instructions” on page 5.

Examples

Example: List the details about all environments.

```
sas-admin qkbs environments list --all
```

Example: List all the QKBs that are registered in the system.

```
sas-admin qkbs qkbs list --all
```

Example: Delete a specified QKB.

```
sas-admin qkbs qkbs delete --context serverA --environment CAS --qkb qkbName
```

Example: Delete a specified import job.

```
sas-admin qkbs jobs delete --id QKB-job-ID
```

Example: List the contexts of a particular environment type. Currently, **CAS** is the only supported environment type.

```
1 sas-admin qkbs contexts help list

2 sas-admin qkbs contexts list --environment CAS -all

3 sas-admin qkbs contexts list --environment CAS --superuser-role TRUE --state running
```

1 Display the options that you can use to filter the contexts list.

2 Notice the values that are returned for type, state, and Superuser role. These are examples of valid values that you can use for these options when filtering the search results. For example, you will see examples of valid values for **type** and **state**. You will also see that the valid values for **Superuser role** are **TRUE** or **FALSE**.

3 Search for the QKBs using appropriate values for type, state, and Superuser role.

Example: Submit a job to import a QKB to the CAS server. Enter a separate command to check the status of the job to determine whether it ran successfully.

```
1 sas-admin qkbs qkbs import --context serverA --environment CAS --destination qkbname
 --source /path-to-qarc/demoqkb.qarc

2 sas-admin qkbs jobs show-info --id QKB-job-ID
```

1 Submit a job to import a QKB to a CAS environment using the serverA context. Specify the path and filename of the source QKB archive file and the name of the QKB after the import is complete. When the job is successfully submitted, you will receive a message that includes the ID of the import job. Running the command in this manner submits the job, but does not determine whether the job ran successfully.

2 Display detailed information about the import job that you just ran to determine whether it ran successfully. You must specify the ID that you obtained in the previous step.

Example: Submit a job to import a QKB to the CAS server. Use the **poll** option to show whether the QKB was successfully registered and whether the import job ran successfully.

```
sas-admin qkbs qkbs import --context serverA --environment CAS --destination qkbname
 --source /path-to-qarc/demoqkb.qarc --poll
```

Example: Check the status of a long running QKB job from another Linux shell. This example assumes that you have the ID of the import job for which you want to check the status. The ID should have been returned after you submitted the job.

```
1 ssh userID@server

2 sas-admin qkbs jobs show-info --id QKB-job-ID
```

1 Open a new shell or terminal on the CAS server that your QKB job is running on.

2 Display information about the import job that you just ran. You must specify the ID of the import job that you received when you submitted the job.

Details

- Due to the size of the imported QKBs, you cannot use the **verbose** option. If you attempt to use the **verbose** option, an error results.
- Because QKB files can be large, the processing time for an import job might be lengthy. To check the status of import jobs, you can open another shell and run the **qkbs jobs show-info** command.
- If an import job fails while it is being registered, you must repeat the import process.

- To import or delete a QKB, you must be logged in as a user who has the ability to assume the Superuser role for the CAS server where the QKB is being imported to or deleted from. There is no separate **superuser** or **su** option.

See Also

- "Command-Line Interface: Overview" on page 2
- SAS Viya Administration: QKB Management

CLI Examples: Reports

The following examples assume that you have already signed in to SAS Viya at the command line. See "Command-Line Interface: Preliminary Instructions" on page 5.

Single Command Examples

Example: Show information about the report that has the ID a85235e7-fad1-4f8a-9ad9-ea0d576619e1.

```
sas-admin reports show-info  --id a85235e7-fad1-4f8a-9ad9-ea0d576619e1
```

Example: List the detailed output of all reports that were created after 2017-05-23.

```
sas-admin reports list --created-after 2017-05-23 --details
```

Example: List the reports in the SAS Viya system that were modified by user1.

```
sas-admin reports list --modified-by user1
```

Example: List a maximum of 50 reports that are sorted by ID and in descending order.

```
sas-admin reports list --details --sort-by ~id --limit 50
```

Example: Delete the report that has the ID a85235e7-fad1-4f8a-9ad9-ea0d576619e1.

```
sas-admin reports delete --id a85235e7-fad1-4f8a-9ad9-ea0d576619e1
```

Multiple Command Examples

Note: Some of these tasks must be performed by a user with administrative privileges.

Example: Update the theme that is used by a SAS Visual Analytics report. This example assumes that you already have a SAS Visual Analytics report that is using the Marine theme (the default).

```
1 sas-admin reports themes list

2 sas-admin reports themes show-info --theme-id theme_ID

3 sas-admin reports list --name reportA

4 sas-admin reports themes update --report-id report_ID --theme-id theme_ID
```

1. List the themes that are available for SAS Visual Analytics reports, and record the ID of the theme that you want to use.
2. Show detailed information about the identified theme for updating the report. You use the ID of the theme that was identified in the previous step.
3. List information about reportA that you want to update, and record the ID of the report.
4. Update the theme that is used in reportA to the theme that you identified in step 1. You use the ID of the report that you identified in the previous step.

Note: You can update a report's theme only to a theme of type **system** or **custom**. Themes of types **legacy** and **retired** are allowed in existing reports, but cannot be used to update a theme.

Example: Export the translation worksheets for reports that need to be translated, translate the worksheets, and then import the translated worksheets into the report. Some of these tasks must be performed by a user with administrative privileges whereas the actual translation might be performed by another user.

You need to determine what reports need to be translated and what languages that the reports need to be translated into. You also need to identify the base language that the reports were created in. The translation worksheets that are exported will contain strings to translate, and these strings are written in the base language that the report was created in. You must save the translation worksheets using the UTF-8 character encoding.

For this example, you learn that you need to translate the reports that were created after 01MAY2018 and whose names contain the text “VAN”. You also learn that the reports need to be translated into French, Spanish, and Japanese. You determine that the base language that the reports were written in is English. See “Details” on page 51 for additional information about translation worksheets.

```
1 sas-admin reports list --created-after 2018-05-01 --limit "100" --name-contains VAN

2 sas-admin reports translations export --report-id report_ID
--output-location /output-path-for-worksheet --report-locale fr-FR

3 sas-admin reports translations export --report-id report_ID
--output-location /output-path-for-worksheet --report-locale es-ES

4 sas-admin reports translations export --report-id report_ID
--output-location /output-path-for-worksheet --report-locale ja-JP

5 ######## LOCALE FOR THIS TRANSLATION WORKSHEET ########
fr-FR
############## BEGIN TRANSLATABLE STRINGS ##############
property.label = translated value
property1.label = translated value
property2.label = translated value

6 sas-admin reports translations import --source-file /output-path-for-worksheet/report-name_fr-FR.reports

7 sas-admin reports translations import --source-file /output-path-for-worksheet/report-name_es-ES.reports

8 sas-admin reports translations import --source-file /output-path-for-worksheet/report-name_ja-JP.reports
```

1 List the reports that were created after 01MAY2018 and whose names contain the text “VAN”. Use the **limit** option to specify the maximum number of reports to list. The default value is 20. To make sure that you list all the reports that were created after 01MAY2018, specify a value of 100. If you receive a message at the end of the report list that indicates that more reports are available, list the reports again with a higher value for the **limit** option.

   After running the code to list the reports, you find that there are four reports that you need to translate. In other words, there are four reports that were created after 01MAY2018 and whose names contain the text “VAN”. Note the report ID of each report. You will need the report ID to export the translation worksheets.

   If you do not have administrative privileges, you might be able to obtain the report ID from SAS Environment Manager.

2 Export the translation worksheet for the first of the four reports in French (fr-FR),

3 Export the translation worksheet for the first of the four reports in Spanish (es-ES).

4 Export the translation worksheet for the first of the four reports in Japanese (ja-JP).

   Repeats steps 2, 3, and 4 for the second, third, and fourth reports. Be sure to update the report ID of the report in the command when you move to the second, third, and fourth report. You will issue the command a total of 12 times in this example. (There are four reports, and each report is translated into three languages). Afterward, you should have 12 translation worksheets in the location that is specified in the output location option.

   Note: If another user is performing the translation, then provide them with the translation worksheets. They can proceed with step 5.

5 Here is an example of the lines that exist in each translation worksheet. The line following **LOCALE FOR THIS TRANSLATION WORKSHEET** indicates the language of the translation worksheet.

In the French translation worksheets (*report_name*_fr-FR), locate the line that contains **BEGIN TRANSLATABLE STRINGS**. In the following lines, edit the values to the right of the equal sign (=) and provide the appropriate values, in French. Save the file.

In the Spanish translation worksheets (*report_name*_es-ES), locate the line that contains **BEGIN TRANSLATABLE STRINGS**. In the following lines, edit the values to the right of the equal sign (=) and provide the appropriate values, in Spanish. Save the file.

In the Japanese translation worksheets (*report_name*_ja-JP), locate the line that contains **BEGIN TRANSLATABLE STRINGS**. In the following lines, edit the values to the right of the equal sign (=) and provide the appropriate values, in Japanese. Save the file.

Note: If the translation was performed by another user, then this user must provide the translated worksheets to the user who is performing the import.

6 Import the French translation worksheet (*report_name*_fr-FR).

7 Import the Spanish translation worksheet (*report_name*_es-ES).

8 Import the Japanese translation worksheet (*report_name*_ja-JP).

Details

- The **translations export** command appends the locale value that is specified as the report locale to the name of the translation report. For example, if you export a translation worksheet for reportA and specify the US English locale as the value of the **report-locale** option, the translation worksheet will be saved in a file with the name:

  ```
  reportA_en-US.reports
  ```

  The **translations export** command includes a locale section in the translation worksheet that includes the locale name that you specified as the report locale. Here is an example of the locale section for the locale en-US:

  ```
  ######## LOCALE FOR THIS TRANSLATION WORKSHEET ########
  en-US
  ```

  Note: Before importing a translation worksheet, you must make sure that the language in the locale section of the worksheet matches the language in the name of the translation worksheet. The name of the file that you are importing must be appended with a locale value.

- The character encoding of the files that contain the translation worksheets must be UTF-8.
- The **clear-results-cache** option clears the results cache of report data for SAS Visual Analytics reports in SAS Viya. In multi-tenancy environments, this command can be run only on the provider tenant, but it clears the results cache across all tenants.
- The reports CLI lists only 20 reports at a time by default. To list more than 20 reports, you can use the `--limit` option. To list 50 reports, enter `--limit 50`.
- To specify what report number to start the list with, you can use the `--start` option. Suppose that you have listed the first 20 reports, and you want to list the next 20 reports, starting with report number 21, enter `--start 21`.

See Also

“Command-Line Interface: Overview” on page 2

CLI Examples: Restore

The following examples assume that you have already signed in to SAS Viya at the command line. See "Command-Line Interface: Preliminary Instructions" on page 5.

Examples

Example: Show the history of restore operations.

```
sas-admin restore list
```

Example: Start a restore operation of a specified backup, and specify that the name of the restore operation is restoreA.

```
sas-admin restore start --backup-name backup-ID --slug restoreA
```

See Also

- "Overview" in *SAS Viya Administration: Backup and Restore*
- "Command-Line Interface: Overview" on page 2

CLI Examples: Scoreexecution

The following examples assume that you have already signed in to SAS Viya using the command line. See "Command-Line Interface: Preliminary Instructions" on page 5.

Examples

Example: Display detailed information about the unused resources.

```
sas-admin scoreexecution --output fulljson list-hanging-resources
```

Example: Display information about unused resources in a table format.

```
sas-admin scoreexecution --output text list-hanging-resources
```

Example: Display only the URIs of the unused resources.

```
sas-admin scoreexecution --output text --quiet list-hanging-resources
```

Example: Write the URIs of the unused resources to a file named uris.txt.

```
sas-admin scoreexecution list-hanging-resources --file uris.txt
```

Example: Delete the unused resources listed in the file uris.txt.

```
sas-admin scoreexecution remove-hanging-resources --file uris.txt
```

See Also

- "Command-Line Interface: Overview" on page 2
- SAS Decision Manager: User's Guide

CLI Examples: Tenant Administration

The following examples assume that you have already signed in to SAS Viya at the command line. See "Command-Line Interface: Preliminary Instructions" on page 5.

Examples

Note: The **create**, **delete**, **onboard**, and **offboard** commands are currently reserved for use only in Ansible playbooks. Do not use these commands at an operating system prompt.

Here are typical examples of commands that can be used with the CLI:

Example: List the tenants.

```
sas-admin tenant list
```

Example: Show the properties for the specified tenant.

```
sas-admin tenant show --id tenantID
```

Example: Check the health of the specified tenant.

```
sas-admin tenant health-check --id tenantID
```

Example: Enable the specified tenant.

```
sas-admin tenant enable --id tenantID
```

Example: Disable the specified tenant.

```
sas-admin tenant disable --id tenantID
```

Details

- The Help for the ID of a tenant states that the string must match this pattern: ^[a-z]+[a-z0-9]*

  This pattern means that the ID of a tenant must start with a lowercase letter, followed by any number of lowercase letters. Use of numbers is optional.

- When you enable a tenant so that users can sign in to it, the access policy of the tenant is changed from **providerTenantUsersOnly** to **allUsers**.

  When you disable a tenant so that users can no longer sign in to it, the access policy of the tenant is changed from **allUsers** to **providerTenantUsersOnly**.

  For more information about the access policy of a tenant, see "Edit the Properties of a Tenant" in *SAS Viya Administration: Multi-tenancy*.

See Also

- "Command-Line Interface: Overview" on page 2
- SAS Viya Administration: Multi-tenancy

CLI Examples: Transfer

The following examples assume that you have already signed in to SAS Viya at the command line. See "Command-Line Interface: Preliminary Instructions" on page 5.

Examples

Example: See the commands and subcommands that are available for the transfer import command.

```
sas-admin transfer import --help
```

Example: Export a new transfer package from an object that has the resource URI "/reports/reports/faa7f5f2-0822-4ca0-9f92-23bda3e02738", and name the package "Export Report".

```
sas-admin transfer export --name "Export Report"
```

```
--resource-uri "/reports/reports/faa7f5f2-0822-4ca0-9f92-23bda3e02738"
```

Example: Display the contents of a transfer package that is named “Export Report” and include import and export history of the package.

Note: Beginning with SAS Viya 3.4 in July 2019 and later deployments, the transfer show command was updated to show more information about the package contents.

```
1 sas-admin transfer list --name "Export Report"
2 sas-admin transfer show --id transfer-package-ID --history
```

1 List information about the transfer package, and record the transfer package ID.

2 Show import and export history of the specified package. You will see a list of import and export tasks for the package. Choose the number of the import or export task for which you want history. In this example, task 1 was selected. Here is typical output:

```
Num   Type     StartedTimeStamp           CompletedTimeStamp         State
1     Import   2019-07-12T12:39:35.855Z   2019-07-12T12:39:50.435Z   Completed
2     Import   2019-07-12T12:39:35.855Z   2019-07-12T12:39:50.435Z   Completed

Enter (q)uit or a number to get task details> 1
```

You will see a summary of results for the selected task. Choose the number of the item that you want information about. In this example, item 4, SAS Environment Manager, was selected to get import details about. Here is typical output:

```
Summary results of import:

        Total tasks: 4  Succeeded: 4  Failed: 0  Completed with errors: 0  Completed with warnings: 0
 Skipped: 0

Num   Name                      State       Result
1     Dashboard Items           Completed   /folders/folders/16d6016b-ca0a-438a-b502-538742bef506
2     Application Activity      Completed   /reports/reports/ecec39ad-994f-4055-8e40-4360f410bc6e
3     Products                  Completed   /folders/folders/707b2a8c-b1f8-4253-9a07-32d10234a26c
4     SAS Environment Manager   Completed   /folders/folders/ac7814c4-eb57-45fd-ae8a-d3add0b57075

Enter (q)uit or a number to get task details> 4
```

You will see details about the selected item. Here is typical output:

```
Name        SAS Environment Manager
State       Completed
Started     2019-07-12T12:39:38.515Z
Completed   2019-07-12T12:39:41.055Z
Result      /folders/folders/ac7814c4-eb57-45fd-ae8a-d3add0b57075
Location    /Products/SAS Environment Manager
Message
The resource was successfully promoted.
```

Example: Get the mapping information for the transfer package that is named “Export Report” from the previous example.

```
1 sas-admin transfer list --name "Export Report"

2 sas-admin transfer get-mapping --id transfer-package-ID
```

1 Locate the ID for the transfer package that you want to export.

2 Issue the command to retrieve the mapping information for the “Export Report” transfer package that has the specified ID.

Example: Upload a package source environment to the target environment, and write the mapping file to the specified location.

```
1 sas-admin --profile source transfer download --id transfer-package-ID --file /tmp/MyPackage.json
2 sas-admin --profile target transfer upload --file /tmp/MyPackage.json --mapping /tmp/map.txt
```

1 Download the package to your local machine and store it in a package file that is named MyPackage.json.

2 Upload the MyPackage.json file to the target environment, and write the mapping file to the file map.txt.

Details

You can specify information about the export or import operation that you want to perform using the Transfer service REST API standards, as follows:

- export

  You can specify information about the export operation that you want to perform using the request option of the transfer export command. The option accepts JSON input of type ExportRequest. The content of the input can be contained in a quoted string or in a file. The filename must begin with the at sign (@). The filename can be specified in either of two forms:

  ```
  @filename.txt
  ```

  ```
  @/path/.filename.txt
  ```

  The content of this option makes up the POST request through which the export package is sent.

  Table 6 *HTTP POST Export Request Members*

Name	Type	Description
version	integer	The schema version number of the JSON media type. This is version 1.
name	string	The name of the export job that is used to export objects from a source system to a transfer package. This is also the name of the transfer package that is being created.
description	string	A short description of the export job.
items	list	The list of URIs to include in the transfer package.

  Here is a sample JSON file of type ExportRequest. The name of the file is export.json.

```
{
  "version": 1,
  "name": "My reports",
  "description": "Export of all my reports",
  "items": [
    "/reports/reports/4d083692-3c9a-4f2c-945f-e96fad972036",
    "/folders/folders/d4f1533a-229d-4d45-a5c9-b8a21fbc1e39"
  ]
}
```

  You can use either of the following syntax options of the command to export the information:

  - `sas-admin transfer export --request @/path-to-file/export.json`

Note: When you include the information in a file, the first character following the request option must be the at sign (@). Therefore, if a pathname is used, it must start with the at sign (@).

- UNIX: `sas-admin transfer export --request '{ "version": 1, "name": "My reports", "description": "Export of all my reports", "items": [ "/reports/reports/4d083692-3c9a-4f2c-945f-e96fad972036", "/folders/folders/d4f1533a-229d-4d45-a5c9-b8a21fbc1e39" ] }'`

  Windows: `sas-admin transfer export --request "{ \"version\": 1, \"name\": \"My reports\", \"description\": \"Export of all my reports\", \"items\": [ \"/reports/reports/4d083692-3c9a-4f2c-945f-e96fad972036\", \"/folders/folders/d4f1533a-229d-4d45-a5c9-b8a21fbc1e39\" ] }"`

  Note: When running the transfer CLI in a Windows environment, single quotation marks are not allowed. You must enclose the **`request`** option data in double quotation marks, and then escape the embedded double quotation marks with a backslash (\).

- import

You can specify information about the import operation that you want to perform using the request option of the transfer import command. The option accepts JSON input of type ImportRequest. The content can be contained in a quoted string or in a file. The filename must begin with an at sign (@). The filename can be specified in either of two forms:

`@filename.txt`

`@/path/.filename.txt`

The content of this option makes up the POST request through which the import package is sent.

Table 7 *HTTP POST Import Request Members*

Name	Type	Description
version	integer	The schema version number of the JSON media type. This is version 1.
name	string	The name of the import job that is used to import objects from a transfer package to a target system.
description	string	A short description of the import job.
packageUri	string	The package to import.

Here is an example of a JSON file of type ImportRequest that is named import.json:

```
{
      "version": 1,
      "name": "My reports",
      "description" : "import all my reports ",
      "packageUri": "/transfer/packages/a2ef940e-14ac-4960-9a9a-7689702b06f0"
}
```

You can use either of the following syntax options of the command to import the information:

- `sas-admin transfer import --request @/path-to-file/import.json`

  Note: When you include the information in a file, the first character following the request option must be the at sign (@). Therefore, if a pathname is used, it must start with the at sign (@).

- UNIX:`sas-admin transfer import --request '{ "version": 1, "name": "My reports", "description" : "import all my reports ", "packageUri": "/transfer/packages/a2ef940e-14ac-4960-9a9a-7689702b06f0" }'`

  Windows:`sas-admin transfer import --request "{ \"version\": 1, \"name\": \"My reports\", \"description\" : \"import all my reports \", \"packageUri\": \"/transfer/packages/a2ef940e-14ac-4960-9a9a-7689702b06f0\" }"`

  Note: When running the transfer CLI in a Windows environment, single quotation marks are not allowed. You must enclose the **`request`** option data in double quotation marks, and then escape the embedded double quotation marks with a backslash (\).

See Also

- "How To" in *SAS Viya Administration: Promotion (Import and Export)*
- "Command-Line Interface: Overview" on page 2

Command-Line Interface: Troubleshooting

Message: `token expired and refresh token is not set`

Explanation: You are not currently authenticated to SAS Viya at the command line. See "Command-Line Interface: Preliminary Instructions" on page 5.

Message: `flag provided but not defined`

Explanation: You might have specified a global option in the wrong location. See "Command-Line Interface: Syntax" on page 10.