



SAS[®] Viya[®] Platform: Using the Command-Line Interface

2020.1 - 2023.02

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

Command-Line Interface: Overview	2
Introduction	2
Quick Start	2
Using the CLI	2
About the Examples	3
About This Document	4
Command-Line Interface: Preliminary Instructions	4
Provide the Path to Your Bundle of Trusted CA Certificates	4
Get the CLI and Its Plug-Ins	5
Update the CLI Plug-Ins	6
Create at Least One Profile	6
Use Your Profile to Sign In	7
Advanced: Create a Sign-In Script	8
Advanced: Set Up a Shared Location	9
Command-Line Interface: Syntax	9
Structure	9
Integrated Help	10
Command-Line Interface: Plug-Ins	11
Command-Line Interface: Concepts	14
Output Type	14
Global Command: Output	14
Global Command: Profile	15
Global Command: Authenticate	17

Source Files: JSON Templates	18
Source Files: Data	23
Command-Line Interface: Troubleshooting	25
Command-Line Interface: Examples and Details	25
Introduction	25
CLI Examples: CAS Administration	25
CLI Examples: Notifications	29
CLI Examples: Reports	29

Command-Line Interface: Overview

Introduction

The SAS Viya platform command-line interface (CLI) enables you to interact directly with the SAS Viya platform REST services. You enter commands on a command line and receive responses from the system. You can use the CLI to work with the SAS Viya platform programmatically, as an alternative to using a graphical user interface (GUI).

A unified CLI (`sas-viya-cli`) is new for the SAS Viya platform. Many of the earlier SAS Viya platform CLI plug-ins work with the unified CLI.

Quick Start

Here is a strategy for getting started:

- [Set up](#) your environment to run the CLI.
- Familiarize yourself with [CLI syntax](#) and the [integrated help](#) within the CLI.
- See [“Command-Line Interface: Plug-Ins”](#) for a list of plug-ins with links to examples.

Using the CLI

Here are key points:

- To prepare to use the plug-ins, see [“Command-Line Interface: Preliminary Instructions”](#).
- Use the Help from within each plug-in for information about the available commands, subcommands, and options. See [“Integrated Help”](#).

Note: The integrated Help supersedes this documentation and provides the most current information about any expanded or enhanced functionality.

- Except where otherwise noted, SAS recommends that users with administrative privileges run the CLI in order to ensure optimal results. Users without administrative privileges are able to run the CLI, but the results might not always be as expected.

Here are important details:

- Commands and subcommands are case sensitive.
- You must precede an option of a command with two hyphens (--) if the option is a word.

Note: 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.

- If a single parameter contains spaces (such as an object's name), 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.
- 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.
- Timestamps that are returned from the CLI are in Universal Time Coordinated (UTC) format rather than in local time. By contrast, 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 plug-in shows that the modified date for the same report is 2018-04-26T13:23:47.314Z. This is UTC time.

About the Examples

The following points apply to CLI examples in the SAS Viya platform documentation:

- The examples assume that you have signed in to the SAS Viya platform using the command line. See [“Command-Line Interface: Preliminary Instructions”](#).
- The examples include line breaks within commands to enhance readability. Do not include line breaks when you submit a command.
- 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 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. See [“Default Profile and Named Profiles”](#).
- The examples generally were run in a Linux environment.

About This Document

This document describes set up tasks that enable you to run the CLI, provides information about how to use the integrated Help, and includes examples of how to use selected plug-ins. This document includes global information about the CLI and references to documentation for plug-ins.

Command-Line Interface: Preliminary Instructions

Provide the Path to Your Bundle of Trusted CA Certificates

If your SAS Viya platform deployment is enabled for Transport Layer Security (TLS), the machine where you will run the CLI must trust the certificate.

If you used a trusted root certificate authority (CA) to sign your certificates or your company's signed certificate is already installed as a trusted root certificate on the machine where you will run the CLI, then the certificate should already be trusted and you do not need to proceed with the following steps.

Otherwise, make sure that the truststore has the root CA certificate and all the intermediate CA certificates. To retrieve the entire bundle of trusted root and intermediate CA certificates from your Kubernetes cluster:

- 1 Install the kubectl utility on the machine where you will run the CLI. See the [Kubernetes documentation](#) for more information.

- 2 Obtain a kubeconfig file from your Kubernetes cluster administrator.

- 3 Copy the trustedcerts.pem file from the pod to the local machine:

```
kubectl -n name-of-namespace cp $(kubectl get pod -n name-of-namespace | grep "sas-logon-app" | head -1 | awk -F" " '{print $1}'):security/trustedcerts.pem /tmp/trustedcerts.pem
```

- 4 On the machine where you will run the CLI, set the SSL_CERT_FILE environment variable.

Here is an example on Linux:

```
export SSL_CERT_FILE=/certificate-directory/trustedcerts.pem
```

For more information, see “How To” in [SAS Viya Platform Encryption: Data in Motion](#).

Get the CLI and Its Plug-Ins

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

- 1 Go to the [Support / Downloads and Hot Fixes page](#) and download the appropriate file to your machine.

Note: You must have a valid [SAS profile](#) on [support.sas.com](#) or on [sas.com](#) to download the CLI.

- 2 Prepare the downloaded file for your environment.

UNIX

Expand the file to a directory from which you plan to run the CLI:

```
tar -xvzf /CLI-directory/sas-viya-cli-version-download-linux.tgz
```

From the directory, make sure that the Execute permission is set:

```
chmod +x sas-viya
```

Windows or Macintosh

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

- 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 CLI available to all CLI users.
- 4 [Create a profile](#) and [sign in](#).
- 5 Install or display one or more plug-ins by running these commands:

```
1 sas-viya plugins
2 sas-viya plugins list
3 sas-viya plugins list-repo-plugins
4 sas-viya plugins install --repo SAS plugin-name
5 sas-viya plugins install --repo SAS all
```

- 1 Display the commands that are available to the CLI plug-ins command.
- 2 Display the 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.
- 5 Install all plug-ins from the SAS repository.

Note:

- If one of the following characters precedes a plug-in name from the SAS repository, it means the following:

Table 1 SAS Repository Plug-in Preceding Character

carat (^)	Means that the version of the plug-in in the SAS repository is more current than the version of the installed plug-in.
asterisk (*)	Means that the version of the plug-in in the SAS repository is the same as the version of the installed plug-in.
no preceding character	Means that the SAS repository plug-in is not installed.

- Use ALL to download all plug-ins to the sas-viya CLI at one time. ALL is available only with the REPO option. It is available for download as of January 12, 2022, and is available for all previous versions of the sas-viya CLI.
- The plug-ins are installed in the home directory of the user who ran the `plugins` command. Here is an example location: `home-directory/.sas/viya-plugins`.

Update the CLI Plug-Ins

Here are the steps to update one or all of the CLI plug-ins:

- 1 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 CLI available to all CLI users.
- 2 [Create a profile](#) and [sign in](#).
- 3 Update one of the CLI plug-ins by running this command:

```
sas-viya plugins install --repo SAS plugin-name
```

- 4 Update all of the CLI plug-ins by running this command:

```
sas-viya plugins install --repo SAS all
```

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 Navigate to the directory on the machine that contains the CLI.
- 2 At the command prompt, enter a command to initialize a new profile. Here are examples:

To create a default profile, enter: `sas-viya profile init`

To create a profile called `prod`, enter: `sas-viya --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 Profiles”](#) 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”](#).

3 Respond to the subsequent prompts as follows:

Service Endpoint	Specify the URL for the SAS Viya platform environment. Use the following format: <i>communications-protocol://web-server-host-name:web-server-port</i> Note: The port is required only if it is not the default port of 443. For example: <code>https://host.example.com</code>
------------------	---

Output type	Specify your preferred format for CLI output (<code>text</code> , <code>json</code> , or <code>fulljson</code>). Note: For more information about the output types, see “Output Type” .
-------------	---

Enable ANSI colored output	Specify whether to enable colored output (<code>y</code> or <code>n</code>).
----------------------------	--

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

Use Your Profile to Sign In

- 1 Navigate to the directory on the machine that contains the CLI.
- 2 At the command prompt, enter a command to initiate the sign-in process. Here are examples:

To use your default profile (assuming that the `SAS_CLI_PROFILE` environment variable is not set), enter: `sas-viya auth login`

To use a profile called `prod`, enter:

```
sas-viya --profile prod auth login
```

- 3 At the subsequent prompts, enter your user ID and password.

The authentication has a default timespan before it expires. See “[sas.logon.jwt](#)” in *SAS Viya Platform: Configuration Reference* for more information about the default expiry of an access token in SAS.

You can use the AUTH LOGOUT command to sign out.

Note: When you run the AUTH LOGOUT global command, a bearer token is written to the `credentials.json` file in this directory: `home-directory/.sas`. For more information, see “[Overview](#)”.

Note: If you log in using a named profile, you can specify the PROFILE option in the CLI command or you can set it in an environment variable:

- Specify the PROFILE option in the AUTH LOGIN command.

In the following example, `Target1` is the profile name:

```
sas-viya --profile Target1 auth login
```

- Set the SAS_CLI_PROFILE environment variable to the name of the profile. The environment variable remains in effect until you log off.

For example, if you set the SAS_CLI_PROFILE environment variable to `Target1`, when you sign in to the SAS Viya platform, the environment variable is applied automatically.

In the following example, `Target1` is not specified explicitly because it has been set via the environment variable:

```
sas-viya auth login
```

Advanced: Create a Sign-In Script

You can use a batch script to create a profile and sign in. The following example is from a Linux environment.

CAUTION

The following example script includes a password. If you include a password in the script, you must secure it so that it can be read by only the administrator and the execution tools that use it.

```
#!/bin/sh
```

```
clidir=/CLI-directory
```

```
$clidir/sas-viya auth login --user userID --password password
```

Note: If your password includes special characters, then you must enclose the password in single quotation marks.

Advanced: Set Up a Shared Location

IMPORTANT SAS recommends that each CLI user should download a unique instance of the CLI and install a unique instance of each plug-in. Use the following instructions only if your organization has a policy that enforces the download of only a single instance.

Here are the steps to make the plug-ins available from a shared location:

- 1 Create a profile and sign in.
- 2 Navigate to the config.json file according to the operating system being used.

Table 2 CLI Directory for config.json File

Linux	<code>home-directory/.sas/viya-plugins</code>
Macintosh	<code>home-directory/.sas/viya-plugins</code>
Windows	<code>home-directory\.sas\viya-plugins</code>

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

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

- 4 Propagate this change to the config.json file for all users.

Command-Line Interface: Syntax

Structure

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

```
sas-viya [global options] plug-in-name [command] [command options] [subcommand] [subcommand options] [arguments]
```

Note: Order matters. The global options are not recognized unless they appear after `sas-viya`.

`sas-viya`

specifies the CLI.

[*global options*]

specifies options that are applicable to all plug-ins.

plug-in name

specifies a plug-in.

[*command*]

specifies a command that is specific for the plug-in that you are using.

[*command options*]

specifies options for the plug-in specific command that you are using.

[*subcommand*]

specifies a subcommand for the plug-in specific command that you are using.

[*subcommand options*]

specifies options for the subcommand.

[*arguments*]

specifies arguments for options.

For a list of plug-ins and links to examples, see [“Command-Line Interface: Plug-Ins”](#).

Integrated Help

Global-Level Help

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 plug-in.

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

```
sas-viya help
```

Example: Show the version of the CLI.

```
sas-viya --version
```

Help for Plug-Ins

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

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

```
sas-viya devices --help
```

Example: List the subcommands of the `authorized-devices` command that is a part of the devices plug-in.

```
sas-viya devices help authorized-devices
```

Example: List the options of the `validate` subcommand of the `authorized-devices` command that is a part of the devices plug-in.

```
sas-viya devices authorized-devices help validate
```

Command-Line Interface: Plug-Ins

The `sas-viya` command is the top-level command of the `sas-viya` CLI. The `sas-viya` command is used to initialize, authenticate, and execute the following plug-ins:

Name	Scope and Examples	
audit	Gets SAS audit information. See “Audit: How to (CLI)” in <i>SAS Viya: Auditing</i> .	
authorization	Gets general authorization information and manages rules. See “General Authorization: How To (CLI)” in <i>SAS Viya Platform: General Authorization</i> .	
batch	Enables you to submit SAS programs or commands from a command line to a SAS Viya platform environment running in a Kubernetes cluster for batch processing. See Using the batch Plug-In for the SAS Viya Platform Command-Line Interface .	
cas	Manages CAS administration.	See “CLI Examples: CAS Administration” .
	Manages CAS authorization.	See “CAS Authorization: How To (CLI)” in <i>SAS Viya Platform: CAS Authorization</i> .
	Manages CAS users, sessions, and servers.	See “SAS Cloud Analytic Services: How To (CLI)” in <i>SAS Viya Platform: SAS Cloud Analytic Services</i> .
	Manages CAS formats.	See “CLI Examples: Formats” in <i>SAS Viya Platform: Data</i> .
	Manages guest access for predefined caslibs.	See SAS Viya Platform: Authentication .
compute	Manages the operations of the compute service. See “CLI Examples” in <i>SAS Viya Platform: Programming Run-Time Servers</i> .	
configuration	Manages the operations of the configuration service. See “How to (CLI)” in <i>SAS Viya Platform: Configuration Properties</i> .	

Name	Scope and Examples
credentials	<p>Manages the operations of the credentials service for domains.</p> <p>See “Credentials Domain: How to (CLI)” in <i>SAS Viya Platform: External Credentials</i>.</p>
dcmtransfer	<p>Enables you to transfer rule sets, rule flows, lookup tables, and decisions from a SAS 9.4 environment to a SAS Viya platform environment.</p> <p>See “dcmtransfer Plug-In” in <i>SAS Intelligent Decisioning: Command-Line Interfaces</i>.</p>
decisiongitdeploy	<p>Enables you to deploy decisions and rule sets from a local Git repository to the SAS Micro Analytic Service destination (maslocal) or to a SAS Cloud Analytic Services (CAS) destination.</p> <p>Note: The decisiongitdeploy plug-in to the sas-viya CLI is available as of the 2021.1.2 release.</p> <p>See “decisiongitdeploy Plug-In” in <i>SAS Intelligent Decisioning: Command-Line Interfaces</i>.</p>
devices	<p>Manages mobile devices.</p> <p>See “Mobile: How To (CLI)” in <i>SAS Viya Platform: Mobile Device Administration</i>.</p>
folders	<p>Manages SAS folders.</p> <p>See “Folders: CLI Examples” in <i>SAS Viya Platform: Folders</i>.</p>
fonts	<p>Manages fonts that are provided by SAS as well as custom fonts that are registered in SAS Visual Analytics 8.3 and later.</p> <p>See “Fonts: Reference” in <i>SAS Viya Platform: Fonts</i>.</p>
healthcheck	<p>Note: The healthcheck plug-in is no longer available as of the 2021.2.1 release.</p>
identities	<p>Retrieves identity information and manages custom groups.</p> <p>See “Identity Management: How to (CLI)” in <i>SAS Viya Platform: Identity Management</i>.</p>
job	<p>Manages the operations of the job flow scheduling service.</p> <p>See “Job Flow Scheduling Service: How To (CLI)” in <i>SAS Viya Platform: Jobs and Flows</i>.</p>
launcher	<p>Manages the operations of the launcher service.</p> <p>Note: The launcher plug-in to the sas-viya CLI is available as of the 2022.1.3 release.</p> <p>See “CLI Examples” in <i>SAS Viya Platform: Programming Run-Time Servers</i> for more information.</p>
licenses	<p>Manages SAS product license status and information.</p> <p>See “How To (CLI)” in <i>SAS Viya Platform: Licensing</i>.</p>

Name	Scope and Examples
mip-migration	<p>Transfers content from SAS Model Implementation Platform 3.2 to SAS Risk Engine on the SAS Viya platform.</p> <p>See “mip-migration Plug-In: IMPORT Command” in <i>SAS Risk Engine: Administration</i>.</p>
models	<p>Provides support for registering and managing models within a common repository, as well as for listing publishing destinations and published objects.</p> <p>See SAS Viya Platform: Models Command-Line Interface.</p>
notifications	<p>Manages notifications in SAS Environment Manager.</p> <p>Note: The notifications plug-in to the sas-viya CLI is available as of the 2021.1.3 release.</p> <p>See “CLI Examples: Notifications”.</p>
oauth	<p>Manages OAuth clients.</p> <p>See “CLI Examples: OAuth” in <i>SAS Viya Platform: Authentication</i>.</p>
qkbs	<p>Manages the SAS Quality Knowledge Bases (QKBs) on a CAS server.</p> <p>See “How To (CLI)” in <i>SAS Viya Platform: QKB Management</i>.</p>
reports	<p>Manages reports in SAS Visual Analytics 8.2 and later.</p> <p>See “CLI Examples: Reports”.</p>
rtdmobjectmigration	<p>Imports SAS Real-Time Decision Manager (RTDM) objects that have been exported from a SAS 9.4 environment.</p> <p>See “rtdmobjectmigration Plug-In” in <i>SAS Intelligent Decisioning: Command-Line Interfaces</i>.</p>
scoreexecution	<p>Manages resources that are no longer used by the Score Execution service.</p> <p>See “scoreexecution Plug-In” in <i>SAS Intelligent Decisioning: Command-Line Interfaces</i>.</p>
sid-functions	<p>Creates and manages custom functions and function categories in SAS Intelligent Decisioning.</p> <p>See “sid-functions Plug-In” in <i>SAS Intelligent Decisioning: Command-Line Interfaces</i>.</p>
transfer	<p>Promotes SAS content.</p> <p>See “How To (CLI)” in <i>SAS Viya Platform: Content Migration from SAS Viya 4</i>.</p>
visual-forecasting	<p>Imports projects from SAS Forecast Studio.</p> <p>Note: The visual-forecasting plug-in to the sas-viya CLI is available as of the 2022.1.3 release.</p> <p>See “Importing Projects from SAS Forecast Server” in <i>SAS Visual Forecasting: User’s Guide</i> for more information.</p>

Name	Scope and Examples
workload-orchestrator	<p>Manages and monitors a SAS Workload Orchestrator environment.</p> <p>Note: The workload-orchestrator plug-in to the sas-viya CLI is available as of the 2021.1.3 release.</p> <p>See “Using the SAS Workload Orchestrator CLI Plug-In” in <i>SAS Viya Platform: Workload Management</i>.</p>

TIP For syntax reference information for any plug-in, access the [integrated help](#).

Command-Line Interface: Concepts

Output Type

You must specify an output type when you create your profile. The output types 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 identifies your SAS Viya platform deployment. This process creates the following two files in the directory `home-directory/.sas`:

- `config.json`

Contains information about your SAS Viya platform deployment, including the name of the connection profile, the service endpoint, and the output type (TEXT, JSON, FULLJSON).

The following table documents the location in which the `config.json` file is created:

Table 3 Location of the config.json File

Linux	<code>home-directory/.sas/viya-plugins</code>
Macintosh	<code>home-directory/.sas/viya-plugins</code>
Windows	<code>home-directory\.sas\viya-plugins</code>

- `credentials.json`

Contains the authentication tokens for your session that are created after you issue the AUTH LOGIN global command.

Default Profile and Named Profiles

You can create a default profile or a named profile. If you do not specify a named profile in the 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 deployments from the same machine using the same set of plug-ins. When you sign in to a deployment with a named profile, the associated token is stored for that specific profile. You can work in different deployments 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 use a named profile to eliminate the need 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 named profile.

Note: If you log in using a named profile, you can specify the PROFILE option in the CLI command or you can set it in an environment variable:

- Specify the PROFILE option in the AUTH LOGIN command.

In the following example, `Target1` is the profile name:

```
sas-viya --profile Target1 auth login
```

- Set the `SAS_CLI_PROFILE` environment variable to the name of the profile. The environment variable remains in effect until you log off.

For example, if you set the `SAS_CLI_PROFILE` environment variable to `Target1`, when you sign in to the SAS Viya platform, the environment variable is applied automatically.

In the following example, `Target1` is not specified explicitly because it has been set via the environment variable:

```
sas-viya auth login
```

Examples

Here are typical examples of creating default and named profiles:

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

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

```
sas-viya profile init
```

- At the prompt for service endpoint, enter the URL for the SAS Viya platform environment as follows: `https://endpoint URL`
- At the prompt for output type, enter `text`.
- At the prompt for "whether to 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 platform deployment as the service endpoint.

```
sas-viya --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"
  }
}
```

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

See [“Command-Line Interface: Preliminary Instructions”](#) 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-viya --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 authentication has a default timespan before it expires, so you might need to re-execute the command at a later time to reconnect to the environment. See [“sas.logon.jwt” in SAS Viya Platform: Configuration Reference](#) for more information about the default expiry of an access token in SAS.

To log on, run the AUTH LOGIN command. For syntax information, see [“Structure”](#).

Here are typical examples from a Linux environment:

- Use this command to log on with a default profile: `sas-viya 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-viya --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-viya auth login`

To log off, issue the AUTH LOGOUT command. For syntax information, see [“Structure”](#). Here are typical examples:

- Use this command to log off from the SAS Viya platform environment that is specified in your default profile: `sas-viya 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 platform environment that is specified in the profile named Target1: `sas-viya --profile Target1 auth logout`.

If the `SAS_CLI_PROFILE` environment variable is set to `Target1`, then you can log off from the Target environment as follows: `sas-viya 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-viya cas generate-cas-samples --output-location sample-template-path
2 sas-viya cas caslibs create --help
3 sas-viya cas caslibs create path --server serverA --path path --name caslibA
  --source-file /sample-template-path/path.json
4 sas-viya --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 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 template that you just created. In the next step, you 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-viya cas caslibs create --help
2 sas-viya cas caslibs create path --server serverB --path pathB --name caslibB
  --source-file pathA.json
3 sas-viya --output json cas caslibs show-info --name caslibB --server serverB
```

- 1 List the types of caslibs that can be specified. You use this information in 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 JSON templates for creating user-defined formats with the cas plug-in's `generate-cas-samples` command.

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

```
1 sas-viya cas generate-cas-samples --output-location sample-template-path
2 sas-viya cas format-libraries add-format json --format-library format_libraryA --
  server serverA
  --source-file /sample-template-path/gender.json
3 sas-viya cas --output json format-libraries show-format-ranges --format 'en_us-$gender'
  --format-library format_libraryA --server serverA > formatA_template.json
4 sas-viya 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 templates in the `output-location` option.
- 2 Add a new format to a format library. In the `source-file` option, specify the `gender.json` 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 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. Because 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.

Global Caslib Policies

Example: Create a policy from a sample JSON template.

```
1 sas-viya cas generate-cas-samples --output-location sample-template-path
2 sas-viya cas servers policies define global-caslibs
  --server serverA --source-file /sample-template-path/policies-examples/
  globalCaslibs.json
```

- 1 Generate templates. Specify the folder location for the 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` 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-viya cas generate-cas-samples --output-location sample-template-path
2 sas-viya cas servers policies define global-caslibs
  --server serverA --source-file /sample-template-path/policies-examples/
  globalCaslibs.json
3 sas-viya --output json cas servers policies show-info --policy globalCaslibs
  --server serverA > /path/global-caslib-template.json
4 sas-viya cas servers policies define global-caslibs
  --attributes sessionTables:1000000000 --server serverA
  --source-file /path/global-caslib-template.json
5 sas-viya cas servers policies show-info --policy globalCaslibs --server serverA
```

- 1 Generate templates. Specify the folder location for the 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 `sessionTables` value was specified. The value that you assign to the `sessionTables` 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 `sessionTables` 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-viya cas generate-cas-samples --output-location sample-template-path
2 sas-viya cas servers policies define priority-assignments
  --server serverA --source-file /sample-template-path/policies-examples/
  priorityAssignments.json
```

- 1 Generate templates. Specify the folder location for the 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` 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-viya cas generate-cas-samples --output-location sample-template-path
2 sas-viya cas servers policies define priority-assignments
  --server serverA --source-file /sample-template-path/policies-examples/
  priorityAssignments.json
3 sas-viya --output json cas servers policies show-info --policy priorityAssignments
  --server serverA > /path/priority-assignments-template.json
4 sas-viya cas servers policies define priority-assignments
  --attributes userM:1 --server serverA --source-file /path/priority-assignments-
  template.json
5 sas-viya cas servers policies show-info --policy priorityAssignments --server serverA
```

- 1 Generate templates. Specify the folder location for the 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 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 assignments 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-viya cas generate-cas-samples --output-location sample-template-path
2 sas-viya 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 templates. Specify the folder location for the 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-viya cas generate-cas-samples --output-location sample-template-path
2 sas-viya 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-viya --output json cas servers policies show-info --policy serverA-priority-3
--server serverA > /path/priority-levels-template.json
4 sas-viya cas servers policies define priority-levels --priority 4
--session-tables 2000000000 --server serverA
--source-file /path/priority-levels-template.json
```

- 1 Generate templates. Specify the folder location for the 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 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 shows a policy for priority level 4.

Specify the appropriate values for priority level 4. This example specifies a value for `sessionTables` attribute with the `session-tables` option. The value that you assign to the `session-tables` option 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 value for `sessionTables`.

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 Platform: 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 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 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

[“CLI Examples: Formats” in SAS Viya Platform: Data](#)

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

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

Here is an example of a mobile devices source file:

```
deviceID4
deviceID5
deviceID6
```

Example: Add multiple devices to the denylist 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-viya devices denylist add --source-file /path_to_template/templateA.txt
```

Command-Line Interface: Troubleshooting

Message: token expired and refresh token is not set

Explanation: You are not currently authenticated to the SAS Viya platform at the command line. See [“Command-Line Interface: Preliminary Instructions”](#).

Message: flag provided but not defined

Explanation: You might have specified a global option in the wrong location. See [“Command-Line Interface: Syntax”](#).

Command-Line Interface: Examples and Details

Introduction

The following topics contain examples and details for selected administrative plug-ins. For examples and details about other plug-ins, see [“Command-Line Interface: Plug-Ins”](#).

For background information, see [“About the Examples”](#).

CLI Examples: CAS Administration

Note:

- When using the cas plug-in to the sas-viya CLI, make sure that the URL that you specify for the SAS Viya platform environment in your profile (`Service Endpoint`) does not end with a forward slash (/).
- 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 Platform: External Credentials*.

The following examples assume that you have already signed in to the SAS Viya platform at the command line. See [“Command-Line Interface: Preliminary Instructions”](#).

Generate CAS Samples

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

```
sas-viya 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”](#)

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”](#).

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-viya cas help tables
2 sas-viya cas tables help list
3 sas-viya 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-viya cas tables load --table airlines --server serverA --caslib caslibA
```

Example: Load 3 copies of the given table for the specified caslib and CAS server.

```
sas-viya cas tables load --table airlines --server serverA --caslib caslibA --copies 3
```

Note: As of the 2020.1.3 release, the `--copies` option is available to specify how many copies of the table to load. The `--copies` option is honored only in massively parallel processing (MPP) environments. The option is ignored if used in a symmetric multi-processing (SMP) environment. Also, if the number specified for the `--copies` option exceeds the number of available nodes, the number of available nodes is used.

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

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

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

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

Example: Import and load the specified shapefiles into CAS as a table.

```
sas-viya cas tables import shape --server serverA --caslib caslibA --file-format shape
--source-file "path-to-shapefile/shapefile.dbf, path-to-shapefile/shapefile.prj,
path-to-shapefile/shapefile.shp,
path-to-shapefile/shapefile.shx"
```

Note: As of the 2021.1.2 release, the `shape` subcommand is available to load shape files into CAS as a table. In the `--source-file` option, you must specify a file in these formats: SHP, DBF, or SHX. Optionally, you can provide a PRJ file. Specify these files as a quoted string, separated by commas.

Example: Import and load a Microsoft Excel Open XML Spreadsheet as a table into the specified caslib and CAS server. The first row of the spreadsheet is a header row.

```
1 sas-viya cas tables import xlsx --caslib caslibA --server serverA --contains-header-row
--source-file /path-to-file --table airlines
2 sas-viya cas tables list --caslib caslibA --server serverA
```

- 1 Import and load the spreadsheet as a table into the specified caslib and CAS server. Use the `source-file` option to specify the path to the spreadsheet file. Use the `contains-header-row` option to indicate that the first row of the spreadsheet is a header row.
- 2 Verify that the new table was added to the specified caslib and CAS server.

Note: The CAS plug-in does not verify that the type of table to import (specified on the command line following the `import` command) matches the file type of the file that is specified with the `source-file` option. The CAS plug-in attempts to process the file and later produces an unknown encoding error.

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](#). 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”](#).

Example: Generate sample template files for creating caslibs.

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

Example: Delete the given caslib from the specified server.

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

See Also

[“Create Caslibs from JSON Templates”](#)

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-viya 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”](#)
- [SAS Viya Platform: Data](#)

CLI Examples: Notifications

Note: The notifications plug-in to the sas-viya CLI is available as of the 2021.1.3 release.

The following examples assume that you have already signed in to the SAS Viya platform at the command line. See [“Command-Line Interface: Preliminary Instructions”](#).

Example: List notifications in SAS Environment Manager.

```
sas-viya notifications list
```

Example: Show details about the notification with ID 48c02a16-e348-4627-8000-70253c47cfee.

```
sas-viya notifications show --id 48c02a16-e348-4627-8000-70253c47cfee
```

Note: Use the `notifications list` command to determine the ID of a notification.

Example: Update the notification with ID 48c02a16-e348-4627-8000-70253c47cfee as read.

```
sas-viya notifications mark-as-read --id 48c02a16-e348-4627-8000-70253c47cfee
```

Example: Send a notification in SAS Environment Manager.

```
sas-viya notifications send --message "The system will be going down in 30 minutes."  
--subject "System outage" --level alert
```

CLI Examples: Reports

The following examples assume that you have already signed in to the SAS Viya platform at the command line. See [“Command-Line Interface: Preliminary Instructions”](#).

Single Command Examples

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

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

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

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

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

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

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

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

Example: Delete the report that has the ID a85235e7-fad1-4f8a-9ad9-ea0d576619e1 without prompting the user for confirmation.

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

Example: List all of the reports in which the name contains the specified string.

```
sas-viya reports list --name-contains Daily // Name contains the word "Daily"
```

Example: Export an entire report to a SAS report package by using the report name and the current directory.

```
sas-viya reports build-package --id report_ID
```

Example: Export an entire report to a SAS report package called `report` to the directory named `/home/myname`.

```
sas-viya reports build-package --id report_ID --output-file report --output-location /home/myname
```

Example: Export a report package that has English text translations, but uses the French locale for data formatting and sorting.

```
sas-viya reports build-package --id report_ID --locale en_US --format-locale fr_FR
```

Note: If you do not specify the `format-locale` option, then the locale is used for both text translations and data formatting and sorting. If neither is specified, then the default CLI locale is used.

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-viya reports themes list
2 sas-viya reports themes show-info --theme-id theme_ID
3 sas-viya reports list --name reportA
4 sas-viya 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 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" for additional information about translation worksheets.

```

1 sas-viya reports list --created-after 2018-05-01 --limit "100" --name-contains VAN
2 sas-viya reports translations export --report-id report_ID
  --output-location /output-path-for-worksheet --report-locale fr-FR
3 sas-viya reports translations export --report-id report_ID
  --output-location /output-path-for-worksheet --report-locale es-ES
4 sas-viya 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-viya reports translations import
  --source-file /output-path-for-worksheet/report-name_fr-FR.reports
7 sas-viya reports translations import
  --source-file /output-path-for-worksheet/report-name_es-ES.reports
8 sas-viya 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*).

Example: Build a SAS report package by using a comma-separated list of pages, where *vi* is the identifier for a page.

```
1 sas-viya reports list-elements --id report_ID // Determines the page IDs
2 sas-viya reports build-package --id report_ID --report-elements "vi1" // One page
```

- 1 Use the `list-elements` command to determine the page IDs in the report.
- 2 Use the `report-element` option to export the information for a single page in the report. You use the page ID that was identified in the previous step.

Example: Build a SAS report package by using a comma-separated list of objects, where *ve* is the identifier for an object.

```
1 sas-viya reports list-elements --id report_ID // Determines the object IDs
2 sas-viya reports build-package --id report_ID --report-elements "ve1,ve2,ve3" //
Three objects
```

- 1 Use the `list-elements` command to determine the object IDs in the report.
- 2 Use the `report-element` option to export the information for three objects in the report. You use the objects IDs that were identified in the previous step.

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 is 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` command clears the results cache of report data for SAS Visual Analytics reports in the SAS Viya platform. In multi-tenancy environment releases prior to 2022.10, the command can be run only on the provider tenant, and clears the results cache for all tenants. In multi-tenancy environment releases 2022.10 and later, this command is tenant-specific. This means that it can now be run on any tenant (including the provider), and clears the results cache for only that tenant (or provider).
- 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”](#)

