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

2026.02 - 2026.05

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 3
- About the Examples 4
- About This Document 4

Command-Line Interface: Instructions when Downloading the CLI **4**
- Provide the Path to Your Bundle of Trusted CA Certificates 5
- Get the CLI and Its Plug-Ins 5
- Update the CLI Plug-Ins 7
- Create at Least One Profile 7
- Use Your Profile to Sign In 8
- Advanced: Create a Sign-In Script 9
- Advanced: Set Up a Shared Location 10

Command-Line Interface: Instructions When Running the CLI from a Container **11**
- Overview 11
- Prerequisites 11
- Pull the sas-viya CLI Container Image 11
- Set Environment Variables 12
- Copy Certificates to the Local Machine 13
- Authenticate to the SAS Viya Platform Using the sas-viya CLI Container 14
- Run the CLI Commands 15
- Make Artifacts Accessible on the Local Machine 16

Command-Line Interface: Syntax ... **16**
Auto-Completion ... 16
Did You Mean This? ... 17
Structure ... 18
Integrated Help ... 18

Command-Line Interface: Plug-Ins ... **19**

Command-Line Interface: Concepts ... **23**
Output Type ... 23
Global Option: Output ... 23
Command and Global Option: Profile ... 24
Global Option: Verbose ... 26
Global Command: Authenticate ... 27
Source Files: JSON Templates ... 28
Source Files: Data ... 34

Command-Line Interface : Troubleshooting ... **35**
General Troubleshooting Strategies ... 35
Diagnose a Problem with Any sas-viya CLI Command ... 35
User Is Unable to Log In ... 35
Command with Option or Global Option Fails ... 36

Command-Line Interface: Examples and Details ... **36**
Introduction ... 36
CLI Examples: CAS Administration ... 36
CLI Examples: Notifications ... 40
CLI Examples: Reports ... 41

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.

- See "Command-Line Interface: Instructions when Downloading the CLI" to download and run the CLI.
- See "Command-Line Interface: Instructions When Running the CLI from a Container" on page 11 to run the CLI from a container.

- 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: Instructions when Downloading the CLI" or "Command-Line Interface: Instructions When Running the CLI from a Container" on page 11.
- 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.

- All users can use the sas-viya CLI, even non-administrative users. However, there are situations where running the sas-viya CLI with administrative permissions might give more optimal results. This is due to the permissions that are required by the underlying functional area, rather than the permissions that are required by the sas-viya CLI itself. For example, suppose that a non-administrator cannot use all of the commands in the identities plug-in to the sas-viya CLI. This might be because they do not have the underlying permissions that are required by the identities service to create identity groups or change group memberships.

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: Instructions when Downloading the CLI".
- 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: Instructions when Downloading the CLI

Note: See "Command-Line Interface: Instructions When Running the CLI from a Container" on page 11 to run the CLI from a container.

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 can skip the following steps.

Otherwise, you must ensure that the truststore has the root CA certificate and all the intermediate CA certificates. The bundle of trusted root and intermediate CA certificates was loaded during deployment, see "Incorporate Additional CA Certificates" in *SAS Viya Platform Encryption: Data in Motion* for more details. To retrieve this bundle 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. It is important to always use the latest CLI.

   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: *communications-protocol://web-server-host-name:web-server-port* **Note:** The port is required only if it is not the default port of 443. 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".
`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 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.

Note: If you want to automate the login process without the requirement to store your username and password on disk, use the `loginCode` option. WIth this method, the user requests an authorization code from SAS Logon Manager, which they can then use to obtain an access token. No username and password are required. Run the `auth loginCode` command, follow the instructions to obtain a login code, and use the code to log in. Here is the command:

```
sas-viya auth loginCode
```

The authentication has a default timespan before it expires. See sas.logon.jwt for more information about the default expiry of an access token in SAS.

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".

You can use the AUTH LOGOUT command to sign out.

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	***home-directory*/.sas/viya-plugins**
Macintosh	***home-directory*/.sas/viya-plugins**
Windows	***home-directory*\.sas\viya-plugins**

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"
```

Note: Make sure that the sas-viya executable file is in a different directory than the shared location that you use for the plug-ins. You installed the sas-viya executable file following the process here on page 5.

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

Command-Line Interface: Instructions When Running the CLI from a Container

Overview

The sas-viya command-line interface (CLI) container image contains the sas-viya CLI and related scripts to support the batch operation of the sas-viya CLI in SAS Viya platform deployments or to function as a standalone container. All of the sas-viya CLI plug-ins here on page 19are installed in the container.

Prerequisites

In order to run the sas-viya CLI from the container image, the following software is required on the machine where you run the sas-viya CLI commands:

- Docker

  See the Docker documentation for installation instructions.

- SAS Mirror Manager

  If SAS Mirror Manager is not included in your SAS Viya deployment, install it. See "Create a Mirror Registry" in *SAS Viya Platform: Deployment Guide* for installation instructions.

- When running the sas-viya CLI from the container image you must run the **`docker run`** command with the option **`--user root`**.

Pull the sas-viya CLI Container Image

Note: SAS Mirror Manager is required in order to connect to the container that is running the sas-viya command-line interface (CLI). If SAS Mirror Manager is not included in your SAS Viya deployment, install it. See "Create a Mirror Registry" in *SAS Viya Platform: Deployment Guide*.

1 Download the certificates.

   a Log in to my.sas.com.

   b Select **My Orders**, and then click any SAS Viya platform software order to display the order details.

c Click **Downloads**. Next, click the **Certificates** check box for the version that you want to deploy, and then click the **Download** button. A ZIP file that includes the license and certificates is downloaded.

d Copy the ZIP file to a directory on the machine where you installed SAS Mirror Manager and Docker.

2 On the machine where you installed SAS Mirror Manager and Docker, run the following command to use SAS Mirror Manager to retrieve login information:

```
mirrormgr list remote docker login --deployment-data path-to-certificate-zip-file
```

Here is an example of the results:

```
docker login -u order-number -p password cr.sas.com
```

3 On the machine where you installed SAS Mirror Manager and Docker, run the command that was returned in the last step in order to log in to the SAS Container Registry:

```
docker login -u order-number -p password cr.sas.com
```

4 Use SAS Mirror Manager to obtain the version tag of the sas-viya CLI container image:

```
mirrormgr list remote docker tags --deployment-data certs.zip --latest | grep sas-viya-cli
```

Here is an example of the results:

```
cr.sas.com/viya-4-x64_oci_linux_2-docker-prod/sas-viya-cli:1.0.13-20240219.1708375629586
```

Substitute the results from this command for *image-name* in the other command examples throughout this topic.

5 Pull the sas-viya CLI container image.

```
docker pull image-name
```

6 Verify that the sas-viya CLI container image has been downloaded from the SAS Container Registry:

```
docker images
```

Here is an example of the results:

```
sas-viya-cli:1.0.13-20240219.1708375629586
```

Set Environment Variables

To run commands from the sas-viya CLI container, you use the `docker run` container command. If you want to use the default profile and do not plan to use multiple named profiles, you must specify the following connection environment variables on the `docker run` container command:

- Set the SAS_SERVICES_ENDPOINT environment variable to the URL for the SAS Viya platform environment. Use the following format for the URL:

  communications-protocol://*web-server-host-name*:*web-server-port*

- Set the VIYA_USER and VIYA_PASSWORD environment variables to the user ID and password that you use to connect to the SAS Viya platform if you are authenticating with a user ID and password.

- Set the JWT_TOKEN environment variable with the JSON Web Token (JWT) bearer token that you use to connect to the SAS Viya platform if you are authenticating with a JWT bearer token.

Here is an example of the `docker run` command with the required environment variables when using the default profile inside the container and authenticating with a user ID and password.

```
docker run -v /tmp:/security --user root -e SAS_SERVICES_ENDPOINT -e VIYA_USER -e VIYA_PASSWORD image-name
```

In some cases, if the syntax in the above example does not work you might need to pass the environment variables as follows:

```
docker run -v /tmp:/security --user root -e SAS_SERVICES_ENDPOINT=$SAS_SERVICES_ENDPOINT -e VIYA_USER=$VIYA_USER -e VIYA_PASSWORD=$VIYA_PASSWORD image-name
```

Note: See "Copy Certificates to the Local Machine" on page 13 for information about `-v /tmp:/security`.

Copy Certificates to the Local Machine

If your SAS Viya platform deployment is enabled for Transport Layer Security (TLS), the machine where you run the sas-viya CLI container commands from must trust the certificate. The sas-viya CLI container connects to the deployment only through the ingress controller and has the same TLS configuration requirements as any browser or other client connecting to the ingress controller. No special configuration is required if the ingress controller TLS certificate is chained to a public CA. If the TLS certificate is self-signed or the CA is not chained to a known public authority, then a custom truststore must be made accessible to the container.

Copy the trustedcerts.pem file from the SAS Viya platform 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
```

The above command copies the trustedcerts.pem file to the **/tmp** directory on the local machine. This example is from a Linux environment.

Note: The sas-viya-cli container uses a custom truststore named trustedcerts.pem if it is mounted into the container at /security. Here is an example of a Docker command where the trustedcerts.pem file stored in the **/tmp** directory on the local machine is mounted into the container at **/security** using the -v option.

```
docker run -v /tmp:/security --user root -e SAS_SERVICES_ENDPOINT -e VIYA_USER -e VIYA_PASSWORD image-name
```

You must mount the certificate into the container's **/security** location on each Docker command.

Authenticate to the SAS Viya Platform Using the sas-viya CLI Container

IMPORTANT When running the sas-viya CLI from the download site, the profile and credential tokens are stored in the ***/userhome/.sas*** directory on the user's local machine. However, when running the sas-viya CLI from the container image, the profile and credential tokens are stored in the **/cli-home/.sas** directory in the image itself. When running the sas-viya CLI from the container image, it is not recommended to store the profile and credential tokens on your local machine (<userhome>.sas).

The process of creating a sas-viya CLI profile and signing in (authenticating) is different when you run the CLI from a container from when you download and run the CLI.

- If you want to use the default profile and do not need multiple named profiles, create the profile and log in by including the following environment variables with the `-e` option on the `docker run` command:
  - The SAS_SERVICES_ENDPOINT environment variable that specifies the location of the SAS Viya platform endpoint.
  - The VIYA_USER and VIYA_PASSWORD environment variables if authenticating with a user name and password.
  - The JWT_TOKEN environment variable if authenticating with the JWT bearer token.

  For example, the following Docker command creates a sas-viya CLI default profile and authenticates with a user name and password. In this scenario, there is no need to specify an endpoint with the **`sas-viya profile`** command or authenticate with the **`sas-viya auth`** command.

  ```
  docker run -v /tmp:/security --user root -e SAS_SERVICES_ENDPOINT -e VIYA_USER -e
  VIYA_PASSWORD image-name
  ```

  Note: See "Copy Certificates to the Local Machine" on page 13 for information about `-v /tmp:/security`.

- If you want to use multiple named profiles, create the profile and log in with the `profile` command and the `--profile` global option on the individual sas-viya CLI commands while running in interactive mode. Running the sas-viya CLI commands in interactive mode is discussed here on page 15. Do not set the connection environment variables if you are using multiple named profiles See "Command and Global Option: Profile" on page 24.

  For example, use the following commands to open a shell to run the sas-viya CLI container in interactive mode and then create a named profile and log in.

  ```
  docker run -v /tmp:/security --user root -it image-name
  ```

  ```
  ./sas-viya --profile profile-name profile list
  ```

  ```
  ./sas-viya --profile profile-name auth login
  ```

Run the CLI Commands

The sas-viya CLI container image includes the sas-viya executable and the SAS Viya platform sas-viya CLI plug-ins. You do not need to create a SAS repository and install the plug-ins before you run commands. Once you are connected to the sas-viya CLI container, you can run commands as follows:

- Run a single sas-viya CLI command from outside of the container.
- Run sas-viya CLI commands in interactive mode from inside of the container.

Run a Single Command from outside of the Container

Running the base image with the `docker run` command with arguments causes the sas-viya CLI command to be run with the specified arguments. If the connection environment variables (SAS_SERVICES_ENDPOINT, VIYA_USER, VIYA_PASSWORD, JWT_TOKEN) are set, a login is performed before running the sas-viya CLI command.

For example, in this command the `sas-viya profile` and `sas-viya login` are performed before the `--version` global command is run.

```
docker run -v /tmp:/security --user root -e SAS_SERVICES_ENDPOINT -e VIYA_USER -e VIYA_PASSWORD image-name --version
```

Likewise, in this command the `sas-viya profile` and `sas-viya login` are performed before the `plugins list` command is run.

```
docker run -v /tmp:/security --user root -e SAS_SERVICES_ENDPOINT -e VIYA_USER -e VIYA_PASSWORD image-name plugins list
```

Run Commands in Interactive Mode from inside of the Container

Running the base image with the `docker run` command with no arguments causes the shell in which the sas-viya CLI commands execute to run inside of the container. If the connection environment variables (SAS_SERVICES_ENDPOINT, VIYA_USER, VIYA_PASSWORD, JWT_TOKEN) are set, a login is performed by a separate `docker run` command that is run before running the sas-viya CLI commands.

In the following example, the `docker run` command with no arguments performs the sas-viya CLI login and then opens a Linux shell. The next two sas-viya CLI commands run interactively from inside of the shell. There is no requirement for `docker run` to precede each command, because you are running from inside of the shell where the login has already been performed.

```
docker run -v /tmp:/security --user root -it -e SAS_SERVICES_ENDPOINT -e VIYA_USER -e
VIYA_PASSWORD image-name
./sas-viya --version
./sas-viya plugins list
```

Make Artifacts Accessible on the Local Machine

If a sas-viya CLI plug-in running from the container image creates any artifacts such as output files, use the `-v` option on the `docker run` command to mount a local directory to the container so that you can access the artifacts.

For example, use the following single line command to run the LIST command of the AUDIT plug-in and redirect the CSV file that is created to a directory on the local machine.

```
docker run -v /tmp:/security --user root -e SAS_SERVICES_ENDPOINT -e VIYA_USER -e VIYA_PASSWORD
image-name audit list --limit "1" --csv /container_directory
```

Command-Line Interface: Syntax

Auto-Completion

Auto-completion is an optional feature that suggests SAS Viya CLI commands based on what the user has typed. When enabled, press the tab key to see possible matching commands. This should expedite coding and reduce errors. At present the suggestions are limited to **commands**, not to **flags** or **options**.

Auto-completion is supported for:

- Bash (version 4 or later)
- Zsh
- PowerShell (version 5 or later)

Enabling

The auto-completion module must be installed for your chosen shell.

For Bash, enter:

```
type complete
```

to check if the module is installed.

If the module is missing install it:

For Ubuntu/Debian enter: `sudo apt install bash-completion`

For RHEL/CentOS enter: `sudo yum install bash-completion`

For Zsh auto-completion is included by default.

For PowerShell, if auto-completion shows errors, these are the steps to try:

- If `$PROFILE` is not found:

  ```
  mkdir -p (Split-Path $PROFILE)
  ```

- If script execution is blocked:

  ```
  Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
  ```

- If PSReadline module is not found:

  ```
  Install-Module PSReadLine -Scope CurrentUser -Force
  ```

Issue the following command to enable auto-completion:
`./sas-viya enable-completion --true`

Note: Your shell configuration file is updated to source the auto-completion script. The auto-completion script files will be added to the **`~/.sas/scripts`** directory.

Restart the terminal to apply the changes.

Usage

Type a partial CLI command and press Tab. For example, type: `./sas-viya au` followed by Tab. The result should complete the syntax to say: `./sas-viya auth`.

Double click the Tab key to receive subcommand suggestions for the current command. For example, `./sas-viya <tab><tab>`

generates a list of potential next commands. In this case:

profile prof authenicate auth authn curl... and so on.

Disabling

Disable auto-completion by typing:
`./sas-viya enable-completion -f`

Restart or reopen the shell to complete the process.

Did You Mean This?

The SAS Viya platform command-line interface (CLI) includes a feature that helps you when a command is entered incorrectly. This functionality is known as: **Did you mean this?**

If you enter a command that does not exist, the SAS Viya CLI analyzes your input and suggests similar valid commands. The intent is to help you quickly find the correct syntax and update your command.

The feature only activates when you enter an invalid command. The system compares your input with available commands and suggests those which are closest to the invalid entry.

For example, if you type: `./sas-viya ath`, **ath** is not a valid argument. The system returns the following:

```
Did you mean this?
auth, authn, authenticate
sas-viya: 'ath' is not a sas-viya comamnd.
See 'sas-viya --help' for more information.
```

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 models plug-in.

```
sas-viya models --help
```

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

```
sas-viya models help destination
```

Example: List the options of the **`create`** subcommand of the **`destination`** command that is a part of the models plug-in.

```
sas-viya models destination help create
```

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 *SAS Viya: Auditing*.
authorization	Gets general authorization information and manages rules. See "General Authorization: How To (CLI)" in *SAS Viya Platform: General Authorization*.
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. **Note:** Programs and commands include SAS programs, Python, R, other open source languages, and shell commands. See Using the batch Plug-In for the SAS Viya Platform Command-Line Interface.
cas	Manages CAS administration. See "CLI Examples: CAS Administration".

Name	Scope and Examples	
	Manages CAS authorization.	See "CAS Authorization: How To (CLI)" in *SAS Viya Platform: CAS Authorization*.
	Manages CAS users, sessions, and servers.	See "SAS Cloud Analytic Services: How To (CLI)" in *SAS Viya Platform: SAS Cloud Analytic Services*.
	Manages CAS formats.	See "CLI Examples: Formats" in *SAS Viya Platform: Data*.
	Manages guest access for predefined caslibs.	See SAS Viya Platform: Authentication.
compute	Manages the operations of the compute service. See "SAS Compute: Server, Service, and Contexts" in *SAS Viya Platform: Programming Run-Time Servers*.	
common-analytics	Provides support for Common Analytics operations. See "Changing Ownership of a Project in SAS Model Studio" in *SAS Visual Forecasting: User's Guide* for an example of how to use the common-analytics plugin.	
configuration	Manages the operations of the configuration service. See "How to (CLI)" in *SAS Viya Platform: Configuration Properties*.	
credentials	Manages the operations of the credentials service for domains. See "Credentials Domain: How to (CLI)" in *SAS Viya Platform: External Credentials*.	
dagentsrv	Enables you to create, list, and delete data services. It also enables you to view definitions for data sources, catalogs, and schemas. See "Command Line Interface" in *Cloud Data Exchange for the SAS Viya Platform: Administrator's Guide*.	
dcmtransfer	Enables you to transfer rule sets, rule flows, lookup tables, and decisions from a SAS 9.4 environment to a SAS Viya platform environment. See "dcmtransfer Plug-In" in *SAS Intelligent Decisioning: Command-Line Interfaces*.	
decisiongitdeploy	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. See "decisiongitdeploy Plug-In" in *SAS Intelligent Decisioning: Command-Line Interfaces*.	

Name	Scope and Examples
detection	Provides access to the SAS Detection Engine REST interface in a SAS Viya 4 platform. See "detection" in *SAS Detection Command Line Interfaces*.
detection-definition	**Note:** With 2026.05, this plug-in is made a sub-command of the **detection** plug-in. Creates and retrieves information about organizational hierarchies and organizations. See "detection-definition" in *SAS Detection Command Line Interfaces*.
detection-message-schema	**Note:** With 2026.05, this plug-in is made a sub-command of the **detection** plug-in. Enables you to create, apply, and retrieve available message schema templates. See "detection-message-schema" in *SAS Detection Command Line Interfaces*.
folders	Manages SAS folders. See "Folders: How to (CLI)" in *SAS Viya Platform: Folders*.
fonts	Manages fonts that are provided by SAS as well as custom fonts that are registered in SAS Visual Analytics 8.3 and later. See "Fonts: Reference" in *SAS Viya Platform: Fonts*.
identities	Retrieves identity information and manages custom groups. See "Identity Management: How to (CLI)" in *SAS Viya Platform: Identity Management*.
job	Manages the operations of the job flow scheduling service. See "Jobs and Flows: How To (CLI)" in *SAS Viya Platform: Jobs and Flows*.
launcher	Manages SAS Launcher information. See "How To (CLI)" in *SAS Viya Platform: Programming Run-Time Servers* for more details.
licenses	Manages SAS product license status and information. See "How To (CLI)" in *SAS Viya Platform: Licensing*.
listdata	Enables you to create, read, update, and delete an advanced list. See "list data Plug-In" in *SAS Intelligent Decisioning: Command-Line Interfaces*.
migrationmanagement	Manages your data from a run of the SAS Content Assessment inventory application. See "Using the migrationmanagement Plug-in" in *SAS Viya Platform: Content Migration From SAS 9.4*.

Name	Scope and Examples
mip-migration	Transfers content from SAS Model Implementation Platform 3.2 to SAS Risk Engine on the SAS Viya platform. See "mip-migration Plug-In: IMPORT Command" in *SAS Risk Engine: Administration*.
models	Provides support for registering and managing models within a common repository. It also enables you to manage publishing destinations and view a listing of published objects. See SAS Viya Platform: Models Command-Line Interface.
notifications	Manages notifications in SAS Environment Manager. See "CLI Examples: Notifications".
oauth	Manages OAuth clients. See "CLI Examples: OAuth" in *SAS Viya Platform: Authentication*.
reports	Manages reports in SAS Visual Analytics 8.2 and later. See "CLI Examples: Reports".
rtdmobjectmigration	Imports SAS Real-Time Decision Manager (RTDM) objects that have been exported from a SAS 9.4 environment. See "rtdmobjectmigration Plug-In" in *SAS Intelligent Decisioning: Command-Line Interfaces*.
rfc-solution-config	Manages SAS RFC Solution Config internal tooling. See "rfc-solution-config" in *SAS® Fraud Decisioning Command Line Interfaces* for more details.
scoreexecution	Manages resources that are no longer used by the Score Execution service. See "scoreexecution Plug-In" in *SAS Intelligent Decisioning: Command-Line Interfaces*.
sid-functions	Creates and manages custom functions and function categories in SAS Intelligent Decisioning. See "sid-functions Plug-In" in *SAS Intelligent Decisioning: Command-Line Interfaces*.
transfer	Promotes SAS content. See "How To (CLI)" in *SAS Viya Platform: Content Migration from SAS Viya 4*.
visual-forecasting	Imports projects from SAS Forecast Studio. See "Importing Projects from SAS Forecast Server" in *SAS Visual Forecasting: User's Guide* for more information.

Name	Scope and Examples
workload-orchestrator	Manages and monitors a SAS Workload Orchestrator environment. See "SAS Workload Orchestrator: Reference (CLI)" in *SAS Viya Platform: Workload Management*.

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 Option: Output

Use the OUTPUT global option 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.

Command and Global Option: Profile

Overview

Use the PROFILE 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	***home-directory*/.sas/viya-plugins**
Macintosh	***home-directory*/.sas/viya-plugins**
Windows	***home-directory*\.sas\viya-plugins**

- credentials.json

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

Use the PROFILE global option (`--profile` *`profile_name`*) to use a named profile as opposed to the default profile.

Default Profile and Named Profiles

IF you are using the downloaded version of the sas-viya CLI, you can create a default profile or use multiple named profiles. If you do not specify a named profile with the --PROFILE global option in the AUTH LOGIN command, and the SAS_CLI_PROFILE environment variable is not set, then the default profile is used.

If you are using the sas-viya CLI from the container image, you can create a default profile when running the sas-viya CLI in single command mode or interactive mode. You can create a named profile with the --PROFILE global option when running the CLI in interactive mode and the connection environment variables are not set on the `docker run` command.

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
  ```

Note: See "Authenticate to the SAS Viya Platform Using the sas-viya CLI Container" on page 14 for additional information about profiles when using the sas-viya CLI from a container.

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:

- If running the sas-viya CLI from the container image, this example applies only when running in interactive mode.
- 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.

Note: If running the sas-viya CLI from the container image, this example applies only when running in interactive mode.

```
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
```

See "Command-Line Interface: Instructions when Downloading the CLI" for more information about creating profiles.

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

Note: If running the sas-viya CLI from the container image, this example applies only when running in interactive mode.

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

Example: In a sas-viya CLI container image running in interactive mode, create a connection profile named Target1 and log in using the Target1 profile.

```
docker run -it image-name
./sas-viya --profile Target1 profile init
./sas-viya --profile Target1 auth login
```

Global Option: Verbose

Use the VERBOSE global option to run a sas-viya CLI command in debug mode. Doing so causes the sas-viya CLI command to show additional detailed information and output to help you diagnose problems.

Suppose that you got an error when you attempted to use the sas-viya CLI cas plug-in command to create a new path-based caslib. You can use the VERBOSE global option on the command to help determine what the problem is. Here is an example:

```
sas-viya --verbose cas caslibs create path --name salescaslib --path /tmp --server cas-shared-default
```

Here is an excerpt of the output that you might receive from the above command using the VERBOSE global option:

```
HTTP/1.1 409 Conflict
Content-Length: 369
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Connection: keep-alive
Content-Security-Policy: default-src 'self'; object-src 'none'; frame-ancestors 'self'; form-action 'self';
Content-Type: application/vnd.sas.error+json;charset=utf-8;version=2
Date: Thu, 30 May 2024 21:26:42 GMT
Expires: 0
Pragma: no-cache
Sas-Service-Response-Flag: true
Set-Cookie: sas-ingress-nginx=6ce56785c7a2a20e4d612ee33a70199b|7849693f0639d06de3aac95788399f19;
            Path=/casManagement/; Secure; HttpOnly; SameSite=Lax
Strict-Transport-Security: max-age=31536000; includeSubDomains
Vary: Origin
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block

{"version":2,"httpStatusCode":409,"message":"The action was not successful.",
"details":["2-0-2640595: 0x887fc253:PERM_ALLOWLISTED","ERROR: Cannot add caslib salescaslib.
It uses a path that is not in the allowlist.","ERROR: The action stopped due to errors.",
"path: /casManagement/servers/cas-shared-default/caslibs","correlator: 32d25447-9799-44d1-a728-795243ab3fee"]}
The action was not successful.
```

From the output produced with the VERBOSE global option, you can see that the problem is that the specified path for the caslib is not in the allowlist.

Global Command: Authenticate

Note: See "Authenticate to the SAS Viya Platform Using the sas-viya CLI Container" on page 14 for information about authenticating when using the sas-viya CLI from a container.

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 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.

Command-Line Interface : Troubleshooting

General Troubleshooting Strategies

See *Troubleshooting Strategies* for general information about how to troubleshoot issues.

Diagnose a Problem with Any sas-viya CLI Command

To diagnose problems with any sas-viya CLI command, use the VERBOSE global option to run the sas-viya CLI in debug mode. See "Global Option: Verbose" on page 26.

User Is Unable to Log In

Problem

An error such as the following results when logging in to the sas-viya CLI:

```
The user token is expired. Login again before attempting any commands.
```

Explanation
The access token has expired.

Resolution
Re-execute the command to log in to the sas-viya CLI. See "Use Your Profile to Sign In " on page 8.

Command with Option or Global Option Fails

Problem
An error such as the following results after running a sas-viya CLI command that contains an option:

```
flag provided but not defined: option
```

Explanation
An invalid option is used with a command or subcommand or a global option is in the wrong location.

Resolution
If using a global option, make sure to place the global option immediately following the `sas-viya plug-in` command. Then re-execute the command. See "[*global options*] " on page 18.

For other options, re-execute the sas-viya CLI command replacing the option with the `help` command. Review the available options for the command that result. See "Structure" on page 18.

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: Instructions when Downloading the CLI".

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: The **--copies** option specifies 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: The **shape** subcommand loads shapefiles 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

The following examples assume that you have already signed in to the SAS Viya platform at the command line. See "Command-Line Interface: Instructions when Downloading the CLI".

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: Instructions when Downloading the CLI".

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
```

Example: Export an entire report to a SAS report package by using the report ID 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 from one page of a report, where **vi** is the identifier for the page.

```
1 sas-viya reports list-elements --id report_ID --pages // 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 --pages // 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.

- 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"