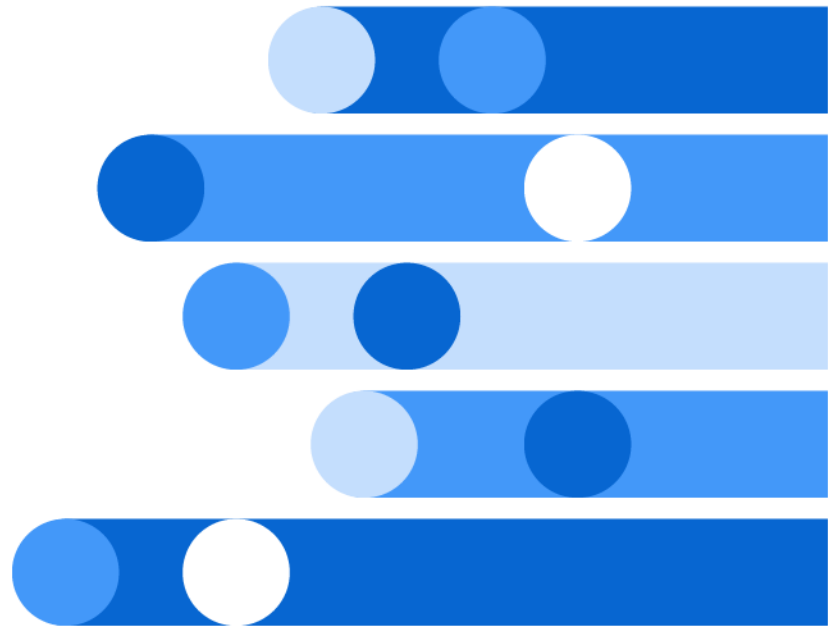




SAS[®] Intelligent Decisioning: Command- Line Interfaces

2024.12*



* 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.



The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2024. *SAS® Intelligent Decisioning: Command-Line Interfaces*. Cary, NC: SAS Institute Inc.

SAS® Intelligent Decisioning: Command-Line Interfaces

Copyright © 2024, SAS Institute Inc., Cary, NC, USA

All Rights Reserved. Produced in the United States of America.

For a hard copy book: No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

For a web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

U.S. Government License Rights; Restricted Rights: The Software and its documentation is commercial computer software developed at private expense and is provided with RESTRICTED RIGHTS to the United States Government. Use, duplication, or disclosure of the Software by the United States Government is subject to the license terms of this Agreement pursuant to, as applicable, FAR 12.212, DFAR 227.7202-1(a), DFAR 227.7202-3(a), and DFAR 227.7202-4, and, to the extent required under U.S. federal law, the minimum restricted rights as set out in FAR 52.227-19 (DEC 2007). If FAR 52.227-19 is applicable, this provision serves as notice under clause (c) thereof and no other notice is required to be affixed to the Software or documentation. The Government's rights in Software and documentation shall be only those set forth in this Agreement.

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

May 2025

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

v_006-P1:edmccli

Contents

Chapter 1 / Using SAS Intelligent Decisioning Command-Line Interfaces	1
About the SAS Intelligent Decisioning CLIs	1
Downloading and Installing the CLIs	2
Creating and Using Profiles	2
Chapter 2 / SAS Intelligent Decisioning CLIs	3
Global Commands	3
Global Options	4
Dictionary	6
Chapter 3 / Transferring Version Comments	59
Transfer Version Comments	59

Using SAS Intelligent Decisioning Command-Line Interfaces

<i>About the SAS Intelligent Decisioning CLIs</i>	1
<i>Downloading and Installing the CLIs</i>	2
<i>Creating and Using Profiles</i>	2

About the SAS Intelligent Decisioning CLIs

In SAS Viya, a command-line interface (CLI) is a user interface to the SAS Viya REST services. In this interface, you enter commands on a command line and receive a response back from the system. You can use a CLI to interact directly with SAS Viya programmatically without a GUI. For information about all of the CLIs that are provided with SAS Viya, see [“Command-Line Interface: Overview” in SAS Viya Platform: Using the Command-Line Interface](#).

The following SAS Intelligent Decisioning CLIs are available:

`dcmtransfer`

enables you to transfer rule sets, rule flows, lookup tables, and decisions from a SAS 9.4 environment to a SAS Viya environment.

`decisiongitdeploy`

enables you to deploy decisions and rule sets from a local Git repository to a SAS Micro Analytic Service environment or to a SAS Cloud Analytic Services (CAS) environment. You must publish the decisions and rule sets to a Git publishing destination and copy them into your local Git repository before you use the CLI.

`listdata`

enables you to create, read, update, and delete an advanced list.

rtdmobjectmigration

imports objects that were exported from SAS Real-Time Decision Manager on SAS 9.4. Currently, the only supported object type is treatments.

scoreexecution

lists or deletes resources such as log files, code files, jobs, and SAS Cloud Analytic Services (CAS) tables that were not deleted when the associated rule set, decision, or model test was deleted.

sid-functions

creates and manages custom functions and function categories.

Downloading and Installing the CLIs

The recommended approach for running the SAS Intelligent Decisioning CLIs is to run them as plug-ins to the SAS Viya Command-Line (sas-viya CLI). The most current plug-ins are included with the sas-viya CLI download file. You can download the sas-viya CLI directly from the SAS Support website and install the plug-ins that you need. The download file is available at <https://support.sas.com/downloads/browse.htm?fil=&cat=564>. For information about downloading the CLI file and installing plug-ins, see “Get the CLI and Its Plug-Ins” in *SAS Viya Platform: Using the Command-Line Interface*.

Creating and Using Profiles

To use these CLIs, you must create at least one profile and sign in to SAS Viya. The CLIs connect to the SAS Viya environment that is specified in your profile. If you do not specify a profile in the CLI command, the CLI uses the default profile. To connect to a different SAS Viya environment, use the `--profile` global option in the CLI command to specify a different profile. For more information, see the following topics:

- “Default Profile and Named Profiles” in *SAS Viya Platform: Using the Command-Line Interface*
- “Create at Least One Profile” in *SAS Viya Platform: Using the Command-Line Interface*
- “Use Your Profile to Sign In” in *SAS Viya Platform: Using the Command-Line Interface*
- “`--profile profile_name`” on page 5

SAS Intelligent Decisioning CLIs

<i>Global Commands</i>	3
<i>Global Options</i>	4
<i>Dictionary</i>	6
dcmtransfer Plug-In	6
decisiongitdeploy Plug-In	17
list data Plug-In	20
rtdmobjectmigration Plug-In	50
scoreexecution Plug-In	52
sid-functions Plug-In	54

Global Commands

The following commands are available for all CLIs:

authenticate

enables you to log on to and log off from the environment. When you log on, this command prompts you for your SAS Viya user ID and password.

Alias auth

Note When you run the AUTH LOGOUT global command, a bearer token is written to the credentials.json file. For more information, see [“Command and Global Option: Profile” in SAS Viya Platform: Using the Command-Line Interface](#).

See [“Global Command: Authenticate” in SAS Viya Platform: Using the Command-Line Interface](#)

[“Command-Line Interface: Instructions when Downloading the CLI” in SAS Viya Platform: Using the Command-Line Interface](#)

help <command>

displays the list of commands that are available for a CLI or, if you specify a command, displays help for that command.

Alias h

Example sas-viya scoreexecution help

plugins list

displays the list of command line plugins for a CLI.

Note No plugins are available for the SAS Intelligent Decisioning CLIs.

Example sas-viya plugins list

profile

creates or displays the connection profile that defines your SAS Viya deployment. This command asks you to enter the SAS Viya service endpoint, your preferred output type, and whether you want to enable colored output. Running the PROFILE global command creates a config.json file and a credentials.json file in the directory *home-directory/.sas*. The config.json file contains information about your SAS Viya deployment, and the credentials.json file contains the authentication tokens that are created when you log in.

Alias prof

See [“Command and Global Option: Profile” in SAS Viya Platform: Using the Command-Line Interface](#)

[“Command-Line Interface: Instructions when Downloading the CLI” in SAS Viya Platform: Using the Command-Line Interface](#)

Global Options

The following options apply to all CLIs.

--colors-enabled true|false

enables or disables ANSI-colored output.

Note This option does not work in all environments.

Tip You can set the environment variable `$SAS_CLI_COLOR` instead of specifying this option.

--help

displays the list of commands and options that are available for a CLI. If this option is specified after a command, it displays the Help for that command.

Alias -h

Example `sas-viya dcmtransfer authenticate9x -help`

--locale *locale*

specifies the locale to use, such as `en` (English) or `de` (German).

Tip You can set either the environment variable `$LC_ALL` or `$LANG` instead of specifying this option.

See https://www.w3schools.com/tags/ref_language_codes.asp

--log-file *filename*

specifies the name of the log file.

Tip You can set the environment variable `$SAS_LOG_FILE` instead of specifying this option.

--output *format*

specifies the format for the output from the CLI. Specify `text`, `json`, or `fulljson`.

Tip You can set the environment variable `$SAS_OUTPUT` instead of specifying this option.

See “Output Type” in *SAS Viya Platform: Using the Command-Line Interface*

--profile *profile_name*

specifies the name of the profile to use.

Alias `-p`

Default `Default`

Tips You can set the environment variable `$SAS_CLI_PROFILE` instead of specifying this option.

You can use the command `CLI-name profile list` to see the list of available profiles.

--quiet

suppresses all output from the CLI except the data.

Alias `-q`

--sas-endpoint *URL*

specifies the URL for the SAS Viya environment.

Tip You can set the environment variable `$SAS_SERVICES_ENDPOINT` instead of specifying this option.

--verbose

displays additional details about the commands that are processed, in addition to the output data.

--version

prints the version of the CLI.

Alias -v

--yes-to-all

suppresses all confirmation prompts by defaulting to yes.

Alias -y

Dictionary

dcmtransfer Plug-In

Enables you to transfer rule sets, rule flows, lookup tables, and decisions from a SAS 9.4 environment to a SAS Viya environment.

Requirement: You must create a profile and sign in before you use the CLI. See [“Creating and Using Profiles” on page 2](#) for more information.

Notes: To transfer content between SAS Viya environments, use the transfer plug-in to the sas-admin and sas-viya CLIs. When you export content from a SAS Viya 3.5 environment, all of the content is transferred except for notes that are associated with specific versions of the objects. For more information, see [“Transfer Version Comments” on page 59](#).

Data tables, tests, comments, attachments, and version information are not transferred from SAS 9.4 to SAS Viya. Vocabularies are not transferred, but the terms used in rule sets and decisions are transferred within those objects.

Lookup tables must be activated in the target environment. See [“Activating a Lookup Table” in SAS Intelligent Decisioning: User’s Guide](#) for more information.

See: [“Global Options” on page 4](#)
[“Global Commands” on page 3](#)

Syntax

sas-viya <*global-options*> **dcmtransfer** *command* <*command-options*>

Commands

In addition to the global commands in [“Global Commands” on page 3](#), you can specify the following commands:

authenticate9x subcommand<options>

logs you in to and out of the SAS 9.4 environment. You can specify the following subcommands:

login

logs you in to the SAS 9.4 environment. The login subcommand accepts these options:

--password *password*

specifies your password.

--service-endpoint *protocol://host_name:port*

specifies the URI for the SAS 9.4 environment.

Note Do not specify `localhost`. You must provide a fully qualified host name.

Example `http://mySAS94server:7980`

--user *user-id*

specifies your user ID.

Tip If you do not specify any options, the login subcommand prompts you for the SAS 9.4 service endpoint, user ID, and password.

logout

logs you out of the SAS 9.4 environment.

Alias `auth9x`

export9x *object-type* <export-options>

exports the specified object types from SAS 9.4. For the object type, specify one of the following: **decisions**, **lookups**, **ruleflows**, or **rulesets**. If you do not specify any options, the CLI prompts you for the required information.

Note: Beginning with SAS Intelligent Decisioning 5.1, SAS Intelligent Decisioning does not support rule flows. You cannot export complex rule flows. However, you can export simple rule flows from previous releases of SAS Intelligent Decisioning and import them as decisions into the current release. If you specify **ruleflows**, the rule flows are exported as decisions. If you specify **decisions**, information about the rule flows that are referenced in the decision is added to the mapping file for the decision.

See [“Options for the export9x Command” on page 8](#)

[“Transfer Content from a SAS 9.4 Environment to a SAS Viya Environment” on page 10](#)

import9x *object-type* <import-options>

imports SAS Intelligent Decisioning objects into SAS Viya. For the object type, specify one of the following: **decisions**, **lookups**, **ruleflows**, or **rulesets**.

See [“Options for the import9x Command” on page 9](#)

[“Transfer Content from a SAS 9.4 Environment to a SAS Viya Environment” on page 10](#)

Options for the export9x Command

TIP For information about using the at sign (@) to specify filenames, see “TRANSFER Command’s REQUEST Option” in *SAS Viya Platform: Content Migration from SAS Viya 4*.

--content @filename

specifies the name of the file to which you want to write the exported content.

Default content.json

--mappings @filename

specifies the name of the mapping file. This file describes the relationship between the exported content and any associated objects.

Default mappings.json

See [“Modifying the Mapping File” on page 12](#)

--report @filename

specifies the file to which you want the CLI to write messages that are generated while the content is exported.

Default export_report.json

--uri URI

specifies the URI for the objects that you want to export. This option is required.

You can specify the URI in any of the following forms.

--uri /SASBusinessRulesManagerWeb/rest/object-type?limit=214783647

--uri /SASDecisionManager/rest/decisions?limit=214783647

specifies all objects of the type *object-type* up to a maximum of 21478367. (This number is the maximum value for an integer.)

--uri /SASBusinessRulesManagerWeb/rest/object-type?filter_query

--uri /SASDecisionManager/rest/decisions?filter_query

specifies all objects that meet the filter criteria and that are of the type specified by the *object-type* argument.

Restriction You cannot use a filter query to specify lookup tables.

See [“Query Parameters for the --uri Option” on page 12](#)

--uri /SASBusinessRulesManagerWeb/rest/object-type/object-ID

--uri /SASDecisionManager/rest/decisions/object-ID

specifies the object with the ID *object-ID* that matches the type specified by the *object-type* argument.

--uri /SASWIPClientAccess/rest/navigation/814100/folder-ID

specifies all of the objects in the folder with the ID *folder-ID* that match the type specified by the *object-type* argument.

Restriction This form is not valid for exporting decisions.

Tip The number 814100 is the object type for a business rules folder.

--uri /SASWIPClientAccess/rest/navigation/814100/DCMFOLDER_ROOT_ID
specifies all objects in the business rules database that match the type specified by the *object-type* argument.

Restriction This form is not valid for exporting decisions.

Tip DCMFOLDER_ROOT_ID is the ID for the root folder.

--uri @filename.txt

specifies all of the objects that are identified by the URIs in the specified text file. The URIs in the text file must all be URIs for objects of the type specified by the *object-type* argument. Each URI must be on a separate line.

Options for the import9x Command

TIP For information about using the at sign (@) to specify filenames, see [“TRANSFER Command's REQUEST Option” in SAS Viya Platform: Content Migration from SAS Viya 4.](#)

--content @filename

specifies the file that contains the objects that you want to import.

Default content.json

--force <true | false>

specifies whether you want existing objects to be replaced if they already exist in the target SAS Viya environment. If you specify **true**, then existing objects with the same name are deleted and re-created from the imported content. If you specify **false**, error messages are generated for the duplicate objects.

The CLI searches the SAS Viya environment for existing objects before it imports new objects. The search criteria that the CLI uses to determine whether an object already exists depends on the information in the mapping file. If the target information for an object is available, the CLI uses it as the search criteria. If the target information is not available, the CLI uses the source information. For example, if you are importing a rule set, the CLI uses the **target.id** and **target.revisionId** as the search criteria. If that information is not available, it uses the **source.name** and **source.folderpath**.

Specifying **--force** without specify **true** or **false** is equivalent to specifying **--force true**.

Alias -f

Default false

See [“Modifying the Mapping File” on page 12](#)

--mappings @filename

specifies the name of the mapping file.

Default mappings.json

See [“Modifying the Mapping File” on page 12](#)

--report @filename

specifies the file to which you want the CLI to write messages that are generated while the content is imported.

Default import_report.json

--target-folder-path /pathname

Specifies the folder in the SAS Viya environment that you want to import the decisions into.

Default /Public

Restriction This option is valid only when the object type in the `import9x` subcommand is **decisions**.

Details

Transfer Content from a SAS 9.4 Environment to a SAS Viya Environment

Note: See [“Command-Line Interface: Instructions when Downloading the CLI” in SAS Viya Platform: Using the Command-Line Interface](#) for additional information.

To transfer business rules and decision content from a SAS 9.4 environment to a SAS Viya environment:

- 1 Log on to the machine where SAS Viya is running. For example, on Linux systems, you can use the `ssh` command:
- 2 Use the `cd` command to change to the directory where you downloaded the plug-in files.
- 3 Create a default profile if you have not already done so:

```
sas-viya profile init
```

The CLI prompts you for the URL (service endpoint) for SAS Viya, for the output type that you want, and whether you want to enable color output. For more information, see [“Create at Least One Profile” in SAS Viya Platform: Using the Command-Line Interface](#).

- 4 Log on to the SAS Viya environment:

```
sas-viya auth login
```

The CLI prompts you for the user ID and password for the SAS Viya environment.

5 Log on to the SAS 9.4 environment:

```
sas-viya dcmtransfer auth9x login
```

The CLI prompts you for the URL (service endpoint) for the SAS 9.4 environment and the user ID and password for that environment.

6 Export the SAS 9.4 content:

```
sas-viya dcmtransfer export9x object-type <export-options>
```

If you do not specify any options, the CLI prompts you for the required information.

By default, the `export9x` command creates three files:

`contents.json`

This file contains the exported content. Do not modify the contents of this file.

`mappings.json`

This file describes the relationships between the exported content and any associated objects. For example, if you export rule sets, the mapping file contains information about any lookup tables that are referenced in the rule sets and all folders where the rule sets and lookup tables reside. You can modify some of the data in this file before you import content into the SAS Viya environment. For more information, see [“Modifying the Mapping File” on page 12](#).

`export_report.json`

This file contains any messages that were generated when the content was exported.

You can use the `--content`, `--mappings`, and `--report` options to change these filenames.

7 Import the content into the SAS Viya environment:

```
sas-viya dcmtransfer import9x object-type <import-options>
```

By default, the `import9x` command looks for a file named `contents.json` that contains the content that was exported from SAS 9.4, and a file named `mappings.json` that describes the relationship between the exported content and any associated objects. If you specified the `--content` or `--mappings` options on the `export9x` command in [Step 6 on page 11](#), then specify the same options on the `import9x` command.

Note: When you view imported content in SAS Intelligent Decisioning, any errors in the content are marked with the error icon (⊗).

8 Log off from the SAS 9.4 environment:

```
sas-viya dcmtransfer auth9x logout
```

9 Log off from the SAS Viya environment:

```
sas-viya auth logout
```

Query Parameters for the --uri Option

The following table lists the query parameters that you can specify as part of the URI when you are exporting SAS 9.4 content. The parameters that are available depend on the object type that you are exporting.

Object Type	Available Parameters	Objects Returned
Rule Sets	name="rule-set-name"	Rule sets that match the specified name. This parameter is case sensitive.
	vocabularyName="vocabulary-name"	Rule sets that use the specified vocabulary
Rule flows	name="rule-flow-name"	Rule flows that match the specified name. This parameter is case sensitive.
	simpleOnly="true"	Beginning with SAS Intelligent Decisioning 5.1, SAS Intelligent Decisioning does not support complex rule flows.
Decisions	name="decision-name"	Decisions that match the specified name. This parameter is case sensitive.
	vocabularyName="vocabulary-name"	Decisions that use the specified vocabulary

Modifying the Mapping File

The mapping file is a JSON file that provides information about exported content and any associated objects. The content of the mapping file depends on the content that is exported. For example, if you export rule flows, then the mapping file contains information about the rule sets that are included in the rule flows and all folders where the rule sets and rule flows reside. If you export decisions, the mapping file contains information about the models and rule sets that are referenced in the decision. (Rule flows are exported as rule sets. Also, in SAS 9.4, decisions are not in folders, so the mapping file for decisions does not contain folder information.)

By default, rule flows that are imported as decisions, rule sets, and lookup tables are imported into a folder that has the same name as the folder in which it resided

in the SAS 9.4 environment. To import the content into a different folder, edit the mapping file.

You can change some of the information in the mapping file. You can change the **target.*** fields, but do not change the **source.*** fields. For example, you can change the folders into which the content is imported and the IDs that are assigned to the imported content. The following table lists the fields in the mapping file for each object type and specifies whether you can change the field before the content is imported.

Table 2.1 Fields in the sas-dcmtransfer CLI Mapping File

Object	Field	Description	Can Be Changed
folders	source.folderPath	Folder path in the SAS 9.4 environment	No
	target.folderPath	Folder path in the SAS Viya environment. By default, this field is set to the same path as source.folderPath.	Yes
lookups	source.id	Lookup table ID in the SAS 9.4 environment	No
	source.name	Lookup table name	No
	source.folderPath	Lookup folder path in the SAS 9.4 environment	No
	target.id	Lookup table ID in the SAS Viya environment	Yes
ruleSets	source.id	Rule set ID in the SAS 9.4 environment	No
	source.name	Rule set name	No
	source.folderPath	Rule set folder path	No
	target.id	Rule set ID in the SAS Viya environment	Yes
	target.revisionId	Rule set version ID in the SAS Viya environment. If no version ID is specified, then the revision ID is set to current . When the rule set information is displayed in the user interface, the current version number is displayed.	Yes
models	source.id	Model ID in the SAS 9.4 environment	No
	source.name	Model name	No
	target.id	Model ID in the SAS Viya environment	Yes

For example, if you export a rule set that references two lookup tables, the mappings file might look like this:

```

{
  "folders": [
    {
      "source": {
        "folderPath": "/A"
      },
      "target": {
        "folderPath": "/B"
      }
    },
  ],
  "lookups": [
    {
      "source": {
        "id": 10093,
        "name": "LookupXYZ",
        "folderPath": "/A"
      },
      "target": {
        "id": ""
      }
    },
    {
      "source": {
        "id": 10094,
        "name": "LookupABC",
        "folderPath": "/B"
      },
      "target": {
        "id": "2e95e765-1242-4c06-ad5a-e05fdc3932fb"
      }
    }
  ]
}

```

If you export rule flow that references two rule sets, the mapping file might look like this:

```

{
  "folders": [
    {
      "source": {
        "folderPath": "/A"
      },
      "target": {
        "folderPath": "/A"
      }
    },
  ],
  "ruleSets": [
    {
      "source": {
        "id": 10043,
        "name": "RuleSetXYZ",
        "folderPath": "/A"
      },
      "target": {

```

```

        "id": "",
        "revisionId": ""
    },
    {
        "source": {
            "id": 10044,
            "name": "RuleSetABC",
            "folderPath": "/A"
        },
        "target": {
            "id": "3619e4d5-3046-4a91-b940-df9dc9c8bd45",
            "revisionId": "b9195eb0-5a1c-4d0a-a833-254bc1ce5a33"
        }
    }
],
}

```

Examples

The following examples use the default profile. For information about defining and specifying a profile, see [“Creating and Using Profiles” on page 2](#).

Example 1

The following command exports the rule set with the ID 10093:

```

sas-viya dcmtransfer export9x rulesets
--uri /SASBusinessRulesManagerWeb/rest/ruleSets/10093

```

Example 2

The following command exports the decision with the ID d246d283-fd80-4d37-96ad-18fd9cd508c8:

```

sas-viya dcmtransfer export9x decisions
--uri /SASDecisionManager/rest/decisions/d246d283-
fd80-4d37-96ad-18fd9cd508c8

```

Example 3

The following command exports all rule flows:

```

sas-viya dcmtransfer export9x ruleflows
--uri /SASBusinessRulesManagerWeb/rest/ruleFlows?limit=2147483647

```

Example 4

The following command exports all lookup tables, writes the exported content to the file `lookupContent.json`, and writes the mapping information to the file `lookupMappings.json`:

```
sas-viya dcmtransfer export9x lookups --mappings "@lookupMappings.json"
--content "@lookupContent.json"
--uri /SASWIPClientAccess/rest/navigation/814100/DCMFOLDER_ROOT_ID
```

Example 5

The following command exports all rule sets that use the vocabulary named `LoanApplications`:

```
sas-viya dcmtransfer export9x rulesets
--uri /SASBusinessRulesManager/rest/ruleSets?
vocabularyName="LoanApplications"
```

Example 6

The following command imports all of the rule sets in the file `content.json`. It uses the mapping file `mappings.json`:

```
sas-viya dcmtransfer import9x rulesets
```

Example 7

The following command imports all lookup tables in the file `lookupContent.json`. It uses the mapping file `lookupMappings.json`.

```
sas-viya dcmtransfer import9x lookups --content "@lookupContent.json"
--mappings "@lookupMappings.json"
```

Example 8

The following command imports all rule flows in the file `content.json`. It uses the mapping file `mappings.json`. The rule flows are imported as decisions.

```
sas-viya dcmtransfer import9x ruleflows
```

Example 9

The following command imports the decisions in the contents.json file. The decisions are imported into the folder named /LoanApplications. Any decision in that folder that has the same name as the decision that is being imported is replaced.

```
sas-viya dcmtransfer import9x decisions --force
--target-folder-path /LoanApplications
```

decisiongitdeploy Plug-In

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.

Requirements: You must create a profile and sign in before you use the CLI. See [“Creating and Using Profiles” on page 2](#) for more information.

You must publish the decisions and rule sets to a Git publishing destination. Then, you must copy the published objects from the Git publishing destination into your local Git repository (by using either the Git clone command or the Git pull command) before you use this CLI to deploy the objects.

You must have Read access to the Git repository.

See: [“Global Options” on page 4](#)
[“Global Commands” on page 3](#)

Syntax

sas-viya *<global-options>* **decisiongitdeploy** deploy *<command-options>* *path-to-object-directory*

Command

In addition to the global commands in [“Global Commands” on page 3](#), you can specify the following command:

deploy *<command-options>* *path-to-object-directory*
 deploys the object from the specified directory into the target destination.

This command supports the following options:

--destinationtype **MAS | CAS**

Specifies the type of destination to which you want to deploy the published object. Specify **MAS** for the maslocal SAS Micro Analytic Service destination or **CAS** for SAS Cloud Analytic Services destinations. (The option values are case insensitive.)

Alias -d

Default **MAS**

--force false | true

Specifies whether to overwrite a previously deployed object that has the same name as the object that you are deploying.

Alias **-f**

Default **False**

--libname *caslib*

Specifies the name of the CAS library to which you want to deploy the object.

Alias **-l**

Default **Public**

Requirement The specified library must exist. See [“Specifying CAS Destination Properties” on page 19](#).

Interaction This option is used only if you specify `--destinationType=CAS`.

--server *server_name*

Specifies the server on which the library specified by the `--library` option is located.

Alias **-s**

Default **cas-shared-default**

Interaction This option is used only if you specify `--destinationType=CAS`.

--tablename *table_name*

Specifies the name of the global CAS model table into which you want to deploy the object.

Alias **-t**

Default **SAS_MODEL_TABLE**

Requirement The specified table must exist. See [“Specifying CAS Destination Properties” on page 19](#).

Interaction This option is used only if you specify `--destinationType=CAS`.

Details

Deploying Published Objects

You can use the decisiongitdeploy plug-in to the sas-viya CLI to deploy decisions and rule sets from a local Git repository to a SAS Micro Analytic Service destination or to a CAS destination.

You can deploy published objects only to destinations with which they are compatible. When you define a Git publishing destination, you can specify whether the code that is generated for and published to the destination is compatible with CAS destinations or with SAS Micro Analytic Service destinations. For more information, see [“Configuring Publishing Destinations” in SAS Intelligent Decisioning: Administrator’s Guide](#).

When you deploy an object to a CAS destination, the object is created as a row in the model table for that destination. When you deploy an object to a SAS Micro Analytic Service destination, the object is created as a module in the service.

TIP You can define a Jenkins job that listens for objects to be published to a Git repository. The job can use this CLI to automatically deploy the published module to the target destinations.

Specifying CAS Destination Properties

For CAS destinations, the CLI does not create the library or table if the library or table that you specify does not exist. The libraries and tables for CAS destinations are created when the publishing destination is configured. You can use the cas plug-in to the sas-viya CLI to verify that the library and table exist. For more information, see [“CLI Examples: CAS Administration” in SAS Viya Platform: Using the Command-Line Interface](#) and [SAS Viya Platform: Publishing Destinations](#).

You can view the properties of a publishing destination, including the library, server, and table name in SAS Environment Manager. To view the CAS destination properties:

- 1 Click  and select **Manage Environment** to navigate to SAS Environment Manager.
- 2 Click  in the navigation bar to view the list of publishing destinations.
- 3 Select the publishing destination and click .

Examples

The following examples use the default profile. For information about defining and specifying a profile, see [“Creating and Using Profiles” on page 2](#).

Example 1

The following command deploys the decision module named `loan_offer1_0` from the local Git repository named `gitMAS` into the SAS Micro Analytic Service destination on the target server that is defined in the default profile:

```
sas-viya decisiongitdeploy deploy ../gitMAS/loan_offer1_0
```

Example 2

The following command deploys the decision module named `loan_offer1_0` from the local Git repository named `gitCAS` into the CAS model table named `loanDecisions` in the Public library on the server `cas-shared-default`:

```
sas-viya decisiongitdeploy deploy --tablename loanDecisions
-d CAS ../gitCAS/loan_offer1_0
```

Example 3

The following command specifies the input directory as you would enter it at a Microsoft Windows shell prompt. This command deploys the published rule set named `masgit_rs_hmeq_value_11feb21` to the SAS Micro Analytic Service destination on the target server that is defined in the default profile. If a module of the same name already exists on the target server, the CLI overwrites the existing module.

```
sas-viya decisiongitdeploy deploy -f true
c:\Users\userid\gitMAS\masgit_rs_hmeq_value_11feb21
```

list data Plug-In

Enables you to create, read, update, and delete an advanced list.

Requirement: You must create a profile and sign in before you use the CLI. See [“Creating and Using Profiles” on page 2](#) for information.

See: [“Global-Level Help” in SAS Viya Platform: Using the Command-Line Interface](#)
[“Help for Plug-Ins” in SAS Viya Platform: Using the Command-Line Interface](#)

Syntax

```
sas-viya <global-options> listdata command <command-option> <subcommand>
<subcommand-option>
```

Commands

In addition to the [global commands](#), you can specify the following commands:

create --data | --file

creates a new list definition by specifying the list's attributes in either a JSON file or string input.

You initially create the list definition by defining the list's properties in a JSON file. For information about list properties, see [Create a List](#) in the *List Data* API documentation. After you create the list definition, you either manually define the list's columns or import list columns and data from a CSV file. For more information about how to define list columns, see the [import-rows on page 28](#) command.

Requirement The `--data` or `--file` option is required.

Examples [“Example 1: Create a list definition by specifying data in a JSON string” on page 35](#)

[“Example 2: Create a list definition by specifying data in a JSON file” on page 35](#)

--data *JSON-string*

defines the attributes for the new list in a JSON string.

--file *file-name.json*

specifies the JSON file that defines the attributes for the new list. You can include an absolute or relative path to the file name.

--parent-folder *folder-location*

specifies the folder path name for the location of the new list.

Note If no folder is specified, the default file location is used. For information about folders, see the **Details** tab of the [Folders](#) API.

delete *subcommand* <*subcommand-option*>

deletes specified contents of a particular list or deletes a specified list.

Note: This option enables you to delete only specified list contents, not all contents. To delete all contents of a list, use the [purge-rows on page 30](#) command.

Requirement All subcommands require the `--id` option. The `contents` subcommand also requires the `--file` option.

Examples [“Example 3: Delete contents from a list by specifying data in a JSON file” on page 35](#)

[“Example 4: Delete a list” on page 36](#)

[“Example 5: Delete a list that is associated with deployed jobs” on page 36](#)

contents --id --file

deletes the contents of a specified list as defined in a JSON file.

--file *file-name.json*

specifies the JSON file that defines the contents to be deleted. You can include an absolute or relative path to the file name.

--id *list-ID*

specifies the list that contains contents to delete.

list --id

deletes a specified list.

--force *false | true*

forces the deletion of a specified list when associated import or purge jobs are incorrectly marked as running.

Default false

--id *list-ID*

specifies the list to delete.

exists *subcommand*

checks whether a specified list has contents or exists.

Requirement All subcommands require the `--id` option.

Examples [“Example 8: Check whether a list has contents” on page 37](#)

[“Example 9: Check whether a list exists” on page 37](#)

contents --id

displays whether the specified list has contents (`true`) or not (`false`).

--id *list-ID*

specifies the list to check for the existence of contents.

list --id

displays whether the specified list exists (`true`) or not (`false`).

--id *list-ID*

specifies the list to check for its existence.

expiry *subcommand*

updates or deletes the expiry on rows in a specified list.

Requirement All subcommands require the `--id` option. The `at` and `in` subcommands also require the `--expiry` option.

Examples [“Example 6: Delete the expiry for a list’s contents by specifying data in a JSON file” on page 36](#)

[“Example 7: Delete the expiry for a list’s contents by specifying data in a JSON string” on page 37](#)

[“Example 67: Update the expiry for a list’s contents to be a number of seconds in the future by specifying data in a JSON file” on page 48](#)

[“Example 68: Update the expiry for a list’s contents to be an exact time by specifying data in a JSON file” on page 49](#)

[“Example 69: Update the expiry for a list’s contents to be an exact time by specifying data in a JSON string” on page 49](#)

at --id

specifies an expiry to be an exact time.

--data *JSON-string*

specifies the list content for which to update an expiry in a JSON string.

--expiry *time-format*

specifies the time at which an item expires by using either the UNIX Epoch time format or UTC time in the format YYYY-MM-DD HH:MM:SS.

--file *file-name.json*

specifies the JSON file that defines the list content for which to update an expiry. You can include an absolute or relative path to the file name.

--id *list-ID*

specifies the identifier for a list.

delete --id

deletes the expiry of contents for a particular list.

--data *JSON-string*

specifies the list content for which to delete an expiry in a JSON string.

--file *file-name.json*

specifies the JSON file that defines the list content for which to delete an expiry. You can include an absolute or relative path to the file name.

--id *list-ID*

specifies the identifier for a list.

in --id

specifies an expiry as the number of seconds in the future.

--data *JSON-string*

specifies the list content for which to update an expiry in a JSON string.

--expiry *number-of-seconds*

specifies the time at which an item expires as the number of seconds in the future.

--file *file-name.json*

specifies the JSON file that defines the list content for which to update an expiry. You can include an absolute or relative path to the file name.

--id *list-ID*

specifies the identifier for a list.

get subcommand <subcommand-option>

retrieves the contents, the consumers, or revisions of a particular list based on specified options, displays a list of one or more lists based on specified options, or displays the state of a specified list. Consumers are the SAS applications or workflows that use a list.

Requirement All subcommands, except `lists`, require the `--id` option.

Examples “Example 10: Get the state of and expiry for a specified number of data rows of a list” on page 37

“Example 11: Get the contents of a list based on specified filter criteria” on page 37

“Example 12: Get a list’s contents” on page 38

“Example 13: Get a specified number of data rows of a list” on page 38

“Example 14: Get the unmasked contents of a specified list (privileged)” on page 38

“Example 15: Get a list’s contents from a specified index point” on page 38

“Example 16: Get the consumers of a specified list” on page 38

“Example 17: Get all lists” on page 39

“Example 18: Get lists by specified filter criteria” on page 39

“Example 20: Get a specified number of lists” on page 39

“Example 19: Get a specified list” on page 39

“Example 21: Get all lists sorted by name alphabetically” on page 39

“Example 22: Get all lists from a specified index point” on page 39

“Example 23: Get the revisions for a specified list” on page 39

“Example 24: Get a particular revision for a specified list” on page 39

“Example 25: Get the state of a list” on page 40

contents --id <subcommand-option>

retrieves the contents and details about the contents, such as each row’s state and expiry, of a particular list based on specified options.

--details

gets the state of and expiries for the specified contents to retrieve.

--filter filter-criteria

specifies the filter criteria for the contents that are retrieved.

Note Filter criteria should be in the REST filtering format. For more information about filtering, see [Filtering](#) in *SAS Viya REST APIs*.

Important You only can apply filtering based on values in key columns. For more information about filtering content, see [Get a List's Contents](#) in the *List Data* API documentation.

--id *list-ID*

specifies the list for which contents are retrieved.

--limit *limit-value*

specifies the maximum number of items to retrieve.

Default 20

--privileged

applies the privileged permission to retrieve unmasked contents of a list.

Note: The List Data service supports the ability to redact (mask) sensitive data for any column in an advanced list. Only users that are granted the privileged permission by their administrator are able to view redacted data or modify the redaction policy for a list. For more information about data masking, see ["Data Masks" in SAS Intelligent Decisioning: User's Guide](#).

--start *number*

specifies the zero-based offset of the first item to retrieve.

Default 0

consumers --id

displays a list of the consumers of the specified list.

id

specifies the list for which consumers are requested.

lists <*sub-command option*>

displays a list of all lists or one or more lists based on specified criteria.

Alias list

--filter *filter-criteria*

specifies the filter criteria for the items that are returned.

Note Filter criteria should be in the REST filtering format. For more information about filtering, see [Filtering](#) in *SAS Viya REST APIs*.

--id *list-ID*

specifies the list to return.

--limit *limit-value*

indicates the maximum number of lists to return.

Default 20

--sort-by *sort-criteria*

sorts the lists that are returned based on a specified value. Valid values are `createdBy`; `createdTimeStamp`; `description`; `label`; `modifiedBy`; `modifiedTimeStamp`; `name`; or `state`.

--start *number*

specifies the zero-based offset of the first list to return.

Default 0

revisions --id *<subcommand-option>*

displays a collection of revisions or a specified revision for a specified list.

--id *list-ID*

specifies the list for which revisions are requested.

--revision-id *revision-ID*

specifies a revision to retrieve.

state --id

displays whether a specified list is deployed or developing.

--id *list-ID*

specifies the list of which to check for the state.

get-import-jobs --id *<command-option>*

displays a list of one or more import jobs that are associated with a particular list based on specified criteria. The list includes both jobs that are running and jobs that have completed.

Requirement The `--id` option is required.

Examples [“Example 26: Get import jobs for a list by specified filter criteria” on page 40](#)

[“Example 27: Get import jobs for a specified list” on page 40](#)

[“Example 28: Get a specified import job for a list” on page 40](#)

[“Example 29: Get a specified number of import jobs for a list” on page 40](#)

[“Example 30: Get all import jobs for a list sorted by name alphabetically” on page 40](#)

[“Example 31: Get all import jobs for a list from a specified index point” on page 41](#)

--filter *filter-criteria*

specifies the filter criteria for the items that are returned.

Note Filter criteria should be in the REST filtering format. For more information about filtering, see [Filtering](#) in *SAS Viya REST APIs*.

--id *list-ID*

specifies the list for which import jobs are returned.

--job-id *job-ID*

specifies the job to retrieve.

--limit *limit-value*

indicates the maximum number of jobs to return.

Default 20

--sort-by *sort-criteria*

sorts the jobs that are returned based on a specified value. Valid values are `createdBy`; `createdTimeStamp`; `description`; `label`; `modifiedBy`; `modifiedTimeStamp`; `name`; and `state`.

--start *number*

specifies the zero-based offset of the first job to return.

Default 0

get-purge-jobs --id <command-option>

displays a list of one or more purge jobs that are associated with a specified list. The list includes both jobs that are running and jobs that have completed.

Requirement The --id option is required.

Examples

[“Example 32: Get purge jobs for a list by specified filter criteria” on page 41](#)

[“Example 33: Get purge jobs for a specified list” on page 41](#)

[“Example 34: Get a specified purge job for a list” on page 41](#)

[“Example 35: Get a specified number of purge jobs for a list” on page 41](#)

[“Example 36: Get all purge jobs for a list sorted by name alphabetically” on page 41](#)

[“Example 37: Get all purge jobs for a list from a specified index point” on page 42](#)

--filter *filter-criteria*

specifies the filter criteria for the items that are returned.

Note Filter criteria should be in the REST filtering format. For more information about filtering, see [Filtering](#) in *SAS Viya REST APIs*.

--id *list-ID*

specifies the list for which purge jobs are returned.

--job-id *job-ID*

specifies the job to retrieve.

--limit *limit-value*

indicates the maximum number of jobs to return.

Default 20

--sort-by *sort-criteria*

sorts the jobs that are returned based on a specified value. Valid values are `createdBy`; `createdTimeStamp`; `description`; `label`; `modifiedBy`; `modifiedTimeStamp`; `name`; and `state`.

--start *index-number*

specifies the zero-based offset of the first job to return.

Default 0

import-rows --id --file

imports the contents from a particular CSV file to a specified list.

Note: The application performs an upsert operation. If keys from the CSV file match existing keys, the data is updated. If keys do not match, the data is appended to the list.

Alias import

Requirement The `--id` and `--file` options are required.

Example [“Example 38: Import contents from a file to a specified list” on page 42](#)

--file *file-name*

specifies the CSV file that defines the contents to be imported. You can include an absolute or relative path to the file name.

--id *list-ID*

specifies the list to which contents are imported.

move-to *subcommand sub-subcommand sub-subcommand-option*

changes the state of a specified list, specified rows of a particular list, or all rows of a specified list to deployed or developing. The subcommand specifies the desired state. The sub-subcommands specify the entity for the state change, which is defined in either a JSON file or string input. A deployed item is available for use by other SAS applications. A developing item is unavailable for use by other SAS applications. If you change the state of rows to deployed, you can also set an expiry for those rows using the `--expire-in` and `--expire-at` options.

Requirement The sub-subcommands require the `--id` option. When using the `rows` sub-subcommand, you must provide an array of row keys via string input or a JSON file with the `--data` and `--file` options, respectively. The `--expire-in` and `--expire-at` options can be used only with the `deployed` subcommand and require the `--id` option.

Examples [“Example 39: Move the state of a list to deployed” on page 42](#)
[“Example 40: Move the state of all rows in a list to deployed” on page 42](#)

“Example 41: Move the state of rows in a list to deployed by specifying data in a JSON string” on page 42

“Example 42: Move the state of rows in a list to deployed by specifying data in a JSON file” on page 42

“Example 43: Move the state of rows in a list to deployed by specifying data in a JSON string, and set an expiry for the rows to be a number of seconds in the future” on page 43

“Example 43: Move the state of rows in a list to deployed by specifying data in a JSON string, and set an expiry for the rows to be a number of seconds in the future” on page 43

“Example 45: Move the state of a list to developing” on page 43

“Example 46: Move the state of all rows in a list to developing” on page 44

“Example 47: Move the state of rows in a list to developing by specifying data in a JSON string” on page 44

“Example 48: Move the state of rows in a list to developing by specifying data in a JSON file” on page 44

deployed | developing

specifies to change the state of an item to deployed or developing.

list --id

specifies to change the state of a particular list.

--id *list-ID*

specifies the identifier for a list.

rows --id --data | --file

specifies to change the state of specific rows in a particular list, which is defined in either string input or a JSON file.

--data *JSON-string*

defines the state change for specified list content in a JSON string.

--expire-in *number-of-seconds*

specifies the time as the number of seconds at which an item expires in the future. You can use this option only with the `deployed` subcommand.

--expire-at *time-format*

specifies the time at which an item expires as either the UNIX Epoch time format or UTC time in the format `YYYY-MM-DD HH:MM:SS`. You can use this option only with the `deployed` subcommand.

--file *file-name.json*

specifies the JSON file that defines the state change for specified list content. You can include an absolute or relative path to the file name.

--id *list-ID*

specifies the identifier for a list.

all-rows --id

specifies to change the state of all rows in a particular list.

--id *list-ID*

specifies the identifier for a list.

purge-rows --id

deletes all content from a specified list.

Note: This option enables you to delete all list contents, not specific contents of a list. To delete specific contents of a list, use the [delete on page 21](#) command.

Alias **purge**

Requirement The `--id` option is required.

Example [“Example 49: Purge contents from a specified list” on page 44](#)

--id *list-ID*

retrieves a list with a specified ID.

revisions *subcommand subcommand-option*

creates a new list revision, deploys a specified list revision, gets the contents of a revision, or compares the contents of a list’s revisions.

Requirement The `--id` option is required for all subcommands. The `revision-id` option is required with the `contents` and `contents-compare` subcommands.

Examples [“Example 50: Get the contents of a list revision” on page 45](#)

[“Example 51: Get the unmasked contents of a list revision \(privileged\)” on page 45](#)

[“Example 52: Get a comparison of the contents of list revisions” on page 45](#)

[“Example 53: Get a comparison of the unmasked contents of list revisions \(privileged\)” on page 45](#)

[“Example 54: Create a new revision of a list” on page 45](#)

[“Example 55: Deploy a specific revision of a list” on page 45](#)

contents --id

gets the contents of a specified list revision.

--filter *filter-criteria*

specifies the filter criteria for the items that are returned.

Note Filter criteria should be in the REST filtering format. For more information about filtering, see [Filtering](#) in *SAS Viya REST APIs*.

--id *list-ID*

specifies the list for which revision contents are requested.

--limit *limit-value*

indicates the maximum number of items to return.

Default 20

--privileged

applies the privileged permission to get unmasked contents of a list revision.

--revision-id *revision-ID*

specifies the revision for which contents are requested.

--start

a zero-based offset of the first item to return.

Default 0

contents-compare --id

returns a collection of the contents of specified list revisions for comparison.

--diffs

an indicator for whether to highlight only the differences.

--id *list-ID*

specifies the list for which revision contents are requested.

--limit *limit-value*

indicates the maximum number of items to return.

Default 20

--privileged

applies the privileged permission to get unmasked contents of a list revision.

--revision *revision-ID* --revision *revision-ID*

specifies the numbers of the revisions for a list for comparison.

--start

a zero-based offset of the first item to return.

Default 0

create --id

creates a new revision of the specified list.

Note: Creating a new revision makes the previous revision immutable, meaning that its contents cannot be changed.

--id *list-ID*

specifies the identifier for a list.

deploy --id

changes the state of the specified list's revision to deployed.

Note: Deploying a revision means that the run-time consumers of list data can access the contents of the revision for the specified list.

--id *list-ID*

specifies the identifier for a list.

--revision-id *revision-ID*

specifies the identifier for a list's revision.

search *subcommand subcommand-option <arguments>*

returns a list of lists or data rows that match specified criteria, which are passed as additional arguments. You can search lists by these values: `createdBy`; `label`; and `modifiedBy`; `name`.

Note: When searching contents, you can search only columns that are marked as key columns. Searching multiple columns at one time produces a logical AND operation.

Requirement The `contents` subcommand requires the `--id` option. The `lists` subcommand requires the `--value` option.

Examples [“Example 56: Search a specified list for a value in a specified column” on page 46](#)

[“Example 57: Search a specified list for particular values in specified columns” on page 46](#)

[“Example 58: Search a specified list for a value in a specified column \(privileged\)” on page 46](#)

[“Example 59: Search the properties of all lists for a specified string” on page 46](#)

contents --id

retrieves the contents of a particular list based on specified options. Search criteria are passed as additional arguments.

--id *list-ID*

specifies the list from which to retrieve contents.

--privileged

applies the privileged permission to get unmasked contents of a list.

Note: The List Data service supports the ability to redact (mask) sensitive data for any column in an advanced list. Only users that are granted the privileged permission by their administrator are able to view redacted data or modify the redaction policy for a list. For more information about data masking, see [“Data Masks” in SAS Intelligent Decisioning: User's Guide](#).

lists --value

returns lists that contain a particular search value.

--value *search-value*

specifies the string value for which to search.

update *subcommand subcommand-option*

updates the contents or properties of a list.

Requirement All subcommands require the `--id` option. The `contents` subcommand also requires the `--file` option.

Examples

- [“Example 60: Update the contents of a specified list” on page 46](#)
- [“Example 61: Update the description property of a list” on page 47](#)
- [“Example 62: Update the properties of a list by specifying data in a JSON file” on page 47](#)
- [“Example 63: Update the immutable property of a list” on page 47](#)
- [“Example 64: Update the label property of a list” on page 47](#)
- [“Example 65: Update the name property of a list” on page 48](#)
- [“Example 66: Update the data mask for a list’s columns by specifying data in a JSON file \(privileged\)” on page 48](#)
- [“Example 70: Update the state property of a list” on page 50](#)

contents --id --file

updates the contents or properties of a specified list by using data that is defined in a JSON file.

--file *file-name.json*

specifies the JSON file that defines the contents to be updated. You can include an absolute or relative path to the file name.

--id *list-ID*

specifies the list for which contents are to be updated.

list --id

updates the properties of a particular list as specified through options or as defined in a JSON file.

--description *textual-string*

defines a new description for the specified list. This option updates the list’s `description` field.

--file *file-name.json*

specifies the JSON file that defines the properties to be updated for the specified list. You can include an absolute or relative path to the file name.

--id *list-ID*

specifies the list for which properties are to be updated.

--immutable *true | false*

specifies whether the list’s contents cannot be changed (`true`) or can be changed (`false`) after the list is created.

Default false

--label *label-value*

defines a new label for the specified list. This option updates the list's `label` field.

--name *name-value*

defines a new name for the specified list. This option updates the list's `name` field.

--privileged

applies the privileged permission to get unmasked contents of a list.

Note: The List Data service supports the ability to redact (mask) sensitive data for any column in an advanced list. Only users that are granted the privileged permission by their administrator are able to view redacted data or modify the redaction policy for a list. For more information about data masking, see [“Data Masks” in SAS Intelligent Decisioning: User's Guide](#).

--state *deployed | developing*

updates the value for the list's `state` field.

Default developing

Details

Using the listData CLI

An advanced list is a tabular data set that can have multiple keys and values and that can be shared among SAS applications and workflows. The `listData` CLI enables you to manage advanced lists in your environment. It also enables you to publish and query those lists in a low-latency, key and value data store.

When you create a list, initially you create the list definition by specifying values for the required properties. Then you define the structure of the list by creating a collection of column information objects that describe the list contents. You are required to identify one or more column as a unique key. By using the `listData` CLI commands and options, you can add data to your lists and manage lists across SAS products and solutions.

If you do not want to use the default location to store your list, you must specify a location. A list inherits the permissions of the folder to which it is a member. Ensure that the appropriate permissions are added for users and groups. For information about managing authorization and permissions, see [“General Authorization: How To \(CLI\)” in SAS Viya Platform: General Authorization](#).

Examples

The following examples use the default profile. For information about defining and specifying a profile, see [“Creating and Using Profiles” on page 2](#).

Example 1: Create a list definition by specifying data in a JSON string

The following command creates a new list definition based on attributes that are specified in a JSON string.

```
sas-viya listdata create --data '{"name":"my-list","columns":
[{"name":"column",
"dataType":"string","position":1,"isKey":true,"keyPosition":1}]}'
```

Example 2: Create a list definition by specifying data in a JSON file

The following command creates a new list definition based on attributes that are defined in a JSON file, and locates the list in a specified directory.

```
sas-viya listdata create --file file.json --parent-folder /home/
myfolder

## Contents of file.json
{
  "name": "my-new-list",
  "description": "This is an example list for the docs",
  "state": "developing",
  "label": "my-new-label",
  "columns": [
    {
      "name": "userId",
      "dataType": "string",
      "position": 1,
      "isKey": true,
      "keyPosition": 1,
      "dataMask": {
        "name": "Mask_All",
        "example": "*****",
        "regex": ".",
        "replace": "*"
      }
    }
  ]
}
```

Example 3: Delete contents from a list by specifying data in a JSON file

The following command deletes data from a list that matches the specified ID. The data to be deleted is defined in a specified JSON file.

Note: You can include an absolute or relative path to the file name.

```
sas-viya listdata delete contents --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --file file.json

## Contents of file.json
{
  "items": [
    {
      "userId": "123456789"
    },
    {
      "userId": "123456788"
    },
    {
      "userId": "123456787"
    },
    {
      "userId": "123456786"
    }
  ]
}
```

Example 4: Delete a list

The following command deletes a list that matches the specified ID.

```
sas-viya listdata delete list --id 8b36c2c4-10fa-4b52-8ffb-a1b6db474f10
```

Example 5: Delete a list that is associated with deployed jobs

The following command forces the specified list to be deleted, even if the list is associated with deployed import or purge jobs that are running on the list.

```
sas-viya listdata delete list --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --force true
```

Example 6: Delete the expiry for a list's contents by specifying data in a JSON file

The following command deletes the expiry for rows in a list that matches the specified ID. The rows that have an expiry to be deleted are defined in a JSON file.

```
sas-viya listdata expiry delete --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --file file.json

## Contents of file.json, where userId is the only key column in list
8b36.
{
  "items": [
    {
      "userId": "123456789"
    },
    {
      "userId": "123456788"
    }
  ],
}
```

```

    {
      "userId": "123456787"
    },
    {
      "userId": "123456786"
    },
  ]
}

```

Example 7: Delete the expiry for a list's contents by specifying data in a JSON string

The following command deletes the expiry for rows in a list that matches the specified ID. The rows that have an expiry to be deleted are defined in a JSON string.

```

sas-viya listdata expiry delete --id 8b36c2c4-10fa-4b52-8ffb-
alb6db474f10 --data '{"items":[{"userId": "123456789"}, {"userId":
"123456788"}]}'

```

Example 8: Check whether a list has contents

The following command checks whether the specified list has contents. If the specified list has contents, the value "true" is returned. If the specified list does not have contents, the value "false" is returned.

```

sas-viya listdata exists contents --id 8b36c2c4-10fa-4b52-8ffb-
alb6db474f10

```

Example 9: Check whether a list exists

The following command checks whether the specified list exists. If the specified list exists, the value "true" is returned. If the specified list does not exist, the value "false" is returned.

```

sas-viya listdata exists list --id 8b36c2c4-10fa-4b52-8ffb-alb6db474f10

```

Example 10: Get the state of and expiry for a specified number of data rows of a list

The following command gets the first three row items in a specified list and gets the state of and expiry for those row items.

```

sas-viya listdata get contents --id 8b36c2c4-10fa-4b52-8ffb-
alb6db474f10 --limit 3 --details

```

Example 11: Get the contents of a list based on specified filter criteria

The following command gets the contents in a specific list that match the specified filter criteria, up to the default maximum number of 20 rows of data.

IMPORTANT There are restrictions on content filtering. For more information about filtering content, see [Get a List's Contents](#) in the *List Data* API documentation.

Note: Filter criteria should be in the REST filtering format. For more information about filtering, see [Filtering](#) in *SAS Viya REST APIs*.

```
sas-viya listdata get contents --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --filter contains($primary,name,'searchval')
```

Example 12: Get a list's contents

The following command gets the contents of a list that matches the specified list ID.

```
sas-viya listdata get contents --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10
```

Example 13: Get a specified number of data rows of a list

The following command gets the first 10 rows of content of a list that matches the specified ID, in the default order.

```
sas-viya listdata get contents --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --limit 10
```

Example 14: Get the unmasked contents of a specified list (privileged)

The following command gets the unmasked contents of a list that matches the specified ID by using the privileged permission.

```
sas-viya listdata get contents --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --privileged
```

Example 15: Get a list's contents from a specified index point

The following command gets the contents of a list that matches the specified ID from index 5 onward, in the default order, up to the default maximum number of 20 rows of data.

```
sas-viya listdata get contents --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --start 5
```

Example 16: Get the consumers of a specified list

The following command gets the consumers of a list that matches the specified ID.

```
sas-list-data-cli get consumers --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10
```

Example 17: Get all lists

The following command gets all lists up to the default maximum number of 20 lists.

```
sas-list-data-cli get lists
```

Example 18: Get lists by specified filter criteria

The following command gets all lists that match the specified filter criteria, up to the default maximum number of 20 lists.

Note: Filter criteria should be in the REST filtering format. For more information about filtering, see [Filtering](#) in *SAS Viya REST APIs*.

```
sas-viya listdata get --filter "contains($primary,name,'searchval')"
```

Example 19: Get a specified list

The following command gets a list that matches the specified list ID.

```
sas-viya listdata get --id 8b36c2c4-10fa-4b52-8ffb-a1b6db474f10
```

Example 20: Get a specified number of lists

The following command gets the first 10 lists in the default order.

```
sas-viya listdata get --limit 10
```

Example 21: Get all lists sorted by name alphabetically

The following command gets all lists sorted by name alphabetically, up to the default maximum number of 20 lists.

```
sas-viya listdata get --sort-by name
```

Example 22: Get all lists from a specified index point

The following command gets all lists from index 5 onward, in the default order up to the default maximum number of 20 lists.

```
sas-viya listdata get --start 5
```

Example 23: Get the revisions for a specified list

The following command gets the revisions for a list that matches the specified ID.

```
sas-viya listdata get revisions --id 8345234b-  
acd2-4342-96db-6f1bf084d7ea
```

Example 24: Get a particular revision for a specified list

The following command gets a specified revision for a list that matches the specified ID.

```
sas-viya listdata get revisions --id 8345234b-
acd2-4342-96db-6f1bf084d7ea --revision-id
4a12e405-1acc-4ec5-812d-3e0e9a65d69d
```

Example 25: Get the state of a list

The following command gets the state (deployed or developing) of a list that matches the specified ID.

```
sas-viya listdata get state --id 8b36c2c4-10fa-4b52-8ffb-a1b6db474f10
```

Example 26: Get import jobs for a list by specified filter criteria

The following command gets the import jobs for a specified list that matches the specified filter criteria, up to the default maximum number of 20 jobs.

Note: Filter criteria should be in the REST filtering format. For more information about filtering, see [Filtering](#) in *SAS Viya REST APIs*.

```
sas-viya listdata get-import-jobs --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --filter "contains($primary,name,'searchval')"
```

Example 27: Get import jobs for a specified list

The following command gets the import jobs for a list that matches the specified list ID.

```
sas-viya listdata get-import-jobs --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10
```

Example 28: Get a specified import job for a list

The following command gets a specific import job for a list that matches the specified list ID.

```
sas-viya listdata get-import-jobs --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --job-id 77f1062f-a768-484c-942c-95cf3ea2a445
```

Example 29: Get a specified number of import jobs for a list

The following command gets the first 10 import jobs for a list that matches the specified list ID, in the default order.

```
sas-viya listdata get-import-jobs --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --limit 10
```

Example 30: Get all import jobs for a list sorted by name alphabetically

The following command gets all import jobs for a list that matches the specified list ID, sorted by name alphabetically, up to the default maximum number of 20 jobs.

```
sas-viya listdata get-import-jobs --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --sort-by name
```

Example 31: Get all import jobs for a list from a specified index point

The following command gets all import jobs for a list that matches the specified list ID, in the default order from index 5 onward, up to the default maximum number of 20 jobs.

```
sas-viya listdata get-import-jobs --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --start 5
```

Example 32: Get purge jobs for a list by specified filter criteria

The following command gets the purge jobs for a specified list that matches the specified filter criteria, up to the default maximum number of 20 jobs.

Note: Filter criteria should be in the REST filtering format. For more information about filtering, see [Filtering](#) in *SAS Viya REST APIs*.

```
sas-viya listdata get-purge-jobs --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --filter "contains($primary,name,'searchval')"
```

Example 33: Get purge jobs for a specified list

The following command gets the purge jobs for a list that matches the specified list ID, up to the default maximum number of 20 jobs.

```
sas-viya listdata get-purge-jobs --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10
```

Example 34: Get a specified purge job for a list

The following command gets a specific purge job for a list that matches the specified list ID.

```
sas-viya listdata get-purge-jobs --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --job-id 77f1062f-a768-484c-942c-95cf3ea2a445
```

Example 35: Get a specified number of purge jobs for a list

The following command gets the first 10 purge jobs for a list that matches the specified list ID, in the default order up to the maximum default number of 20 jobs.

```
sas-viya listdata get-purge-jobs --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --limit 10
```

Example 36: Get all purge jobs for a list sorted by name alphabetically

The following command gets all purge jobs for a list that matches the specified list ID, sorted by name alphabetically, up to the default maximum number of 20 jobs.

```
sas-viya listdata get-purge-jobs --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --sort-by name
```

Example 37: Get all purge jobs for a list from a specified index point

The following command gets all purge jobs for a list that matches the specified list ID, from index 5 onward in the default order, up to the default maximum number of 20 jobs.

```
sas-viya listdata get-purge-jobs --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --start 5
```

Example 38: Import contents from a file to a specified list

The following command imports contents from a specific file to a list that matches the specified list ID.

Note: You can include an absolute or relative path to the file name.

```
sas-viya listdata import-rows --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --file ./file.csv
```

Example 39: Move the state of a list to deployed

The following command sets the state of a list that matches the specified ID to "deployed".

```
sas-viya listdata move-to deployed list --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10
```

Example 40: Move the state of all rows in a list to deployed

The following command sets the state of every row to "deployed" in a list that matches the specified ID.

```
sas-viya listdata move-to deployed all-rows --id
8b36c2c4-10fa-4b52-8ffb-a1b6db474f10
```

Example 41: Move the state of rows in a list to deployed by specifying data in a JSON string

The following command sets the state of rows to "deployed" in a list that matches the specified ID. The rows to be deployed are defined in a JSON string.

```
sas-viya listdata move-to deployed rows --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --data '{"items":[{"userId": "123456789"}, {"userId":
"123456788"}]}'
```

Example 42: Move the state of rows in a list to deployed by specifying data in a JSON file

The following command sets the state of rows to "deployed" in a list that matches the specified ID. The rows to be deployed are defined in a specified JSON file.

```
sas-viya listdata move-to deployed rows --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --file file.json
```

```
## Contents of file.json, where userId is the only key column in list
id 8b36c2c4-10fa-4b52-8ffb-a1b6db474f10
{
  "items": [
    {
      "userId": "123456789"
    },
    {
      "userId": "123456788"
    },
    {
      "userId": "123456787"
    },
    {
      "userId": "123456786"
    }
  ]
}
```

Example 43: Move the state of rows in a list to deployed by specifying data in a JSON string, and set an expiry for the rows to be a number of seconds in the future

The following command sets the state of rows to "deployed" in a list that matches the specified ID. The rows to be deployed are defined in a JSON string. An expiry is set for the rows as the number of seconds in the future.

```
sas-viya listdata move-to deployed rows --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --data '{"items":[{"userId": "123456789"}, {"userId":
"123456788"}]}' --expire-in 2000
```

Example 44: Move the state of rows in a list to deployed by specifying data in a JSON string, and set an expiry for the rows to be an exact time

The following command sets the state of rows to "deployed" in a list that matches the specified ID. The rows to be deployed are defined in a JSON string. An expiry is set for the rows to be December 19, 2023 at 2:30 pm (UTC).

```
sas-viya listdata move-to deployed rows --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --data '{"items":[{"userId": "123456789"}, {"userId":
"123456788"}]}' --expire-at '2023-12-19 14:30:00
```

Example 45: Move the state of a list to developing

The following command sets the state of a list that matches the specified ID to "developing".

```
sas-viya listdata move-to developing --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10
```

Example 46: Move the state of all rows in a list to developing

The following command sets the state of every row to "developing" in a list that matches the specified ID.

```
sas-viya listdata move-to developing all-rows --id
8b36c2c4-10fa-4b52-8ffb-alb6db474f10
```

Example 47: Move the state of rows in a list to developing by specifying data in a JSON string

The following command sets the state of rows to "developing" in a list that matches the specified ID. The rows to be moved to developing are defined in a JSON string.

```
sas-viya listdata move-to developing rows --id 8b36c2c4-10fa-4b52-8ffb-
alb6db474f10 --data '{"items":[{"userId": "123456789"}, {"userId":
"123456788"}]}'
```

Example 48: Move the state of rows in a list to developing by specifying data in a JSON file

The following command sets the state of rows to "developing" in a list that matches the specified ID. The rows to be moved to developing are defined in a specified JSON file.

```
sas-viya listdata move-to developing rows --id 8b36c2c4-10fa-4b52-8ffb-
alb6db474f10 --file file.json
```

```
## Contents of file.json, where userId is the only key column in list
id 8b36c2c4-10fa-4b52-8ffb-alb6db474f10
```

```
{
  "items": [
    {
      "userId": "123456789"
    },
    {
      "userId": "123456788"
    },
    {
      "userId": "123456787"
    },
    {
      "userId": "123456786"
    }
  ]
}
```

Example 49: Purge contents from a specified list

The following command deletes all content from a list that matches the specified list ID.

```
sas-viya listdata purge-rows --id 8b36c2c4-10fa-4b52-8ffb-alb6db474f10
```

Example 50: Get the contents of a list revision

The following command gets the contents of a specified revision for a particular list.

```
sas-viya listdata revisions contents --id 9549912b-
efff-436e-99ea-160009712e22 --revision-id f6db268d-7fe7-4c09-
ae99-4e92d663daf9
```

Example 51: Get the unmasked contents of a list revision (privileged)

The following command uses the privileged permission to get the unmasked contents of a specified revision for a particular list.

```
sas-viya listdata revisions contents --id 9549912b-
efff-436e-99ea-160009712e22 --revision-id f6db268d-7fe7-4c09-
ae99-4e92d663daf9 --privileged
```

Example 52: Get a comparison of the contents of list revisions

The following command gets the contents of multiple revisions of a specified list for comparison.

```
sas-viya listdata revisions contents-compare --id 9549912b-
efff-436e-99ea-160009712e22 --revision 1 --revision 2
```

Example 53: Get a comparison of the unmasked contents of list revisions (privileged)

The following command uses the privileged permission to get the unmasked contents of specified revisions for a particular list.

```
sas-viya listdata revisions contents-compare --id 9549912b-
efff-436e-99ea-160009712e22 --revision 1 --revision 2 --privileged
```

Example 54: Create a new revision of a list

The following command creates a new revision of a list that matches the specified list ID.

```
sas-viya listdata revisions create --id 8345234b-
acd2-4342-96db-6f1bf084d7ea
```

Example 55: Deploy a specific revision of a list

The following command deploys a specified revision of a list that matches the specified ID.

```
sas-viya listdata revisions deploy --id 8345234b-
acd2-4342-96db-6f1bf084d7ea --revision-id
4a12e405-1acc-4ec5-812d-3e0e9a65d69d
```

Example 56: Search a specified list for a value in a specified column

The following command searches the contents of a specified list ID for rows with the value "1234" in the "UserId" column.

```
sas-viya listdata search contents --id 8b36c2c4-10fa-4b52-8ffb-
alb6db474f10 "UserId" "1234"
```

Example 57: Search a specified list for particular values in specified columns

The following command searches the content of a specified list ID for rows with the value "1234" in the "UserId" column AND the value "UK" in the "UserCountry" column.

```
sas-viya listdata search contents --id 8b36c2c4-10fa-4b52-8ffb-
alb6db474f10 "UserId" "1234" "UserCountry" "UK"
```

Example 58: Search a specified list for a value in a specified column (privileged)

The following command searches the content of a specified list ID for rows with the value "1234" in the "UserId" column by using the privileged permission to return the data unmasked.

```
sas-viya listdata search contents --id 8b36c2c4-10fa-4b52-8ffb-
alb6db474f10 --privileged "UserId" "1234"
```

Example 59: Search the properties of all lists for a specified string

The following command searches all lists for the string 'my-list' in any of the searchable columns and returns the lists that meet the search criteria.

```
sas-viya listdata search lists --value my-list
```

Example 60: Update the contents of a specified list

The following command adds four new data entries in the "user-ID" column to a list that matches the specified list ID by passing data from a specified JSON file.

Note: You can include an absolute or relative path to the file name.

```
sas-viya listdata update contents --id 8b36c2c4-10fa-4b52-8ffb-
alb6db474f10 --file file.json

## Contents of file.json
{
  "items": [
    {
      "userId": "123456789"
    },
    {

```

```

        "userId": "123456788"
      },
      {
        "userId": "123456787"
      },
      {
        "userId": "123456786"
      },
    ],
  }
}

```

Example 61: Update the description property of a list

The following command updates the value for the description property of a list that matches the specified list ID.

```

sas-viya listdata update list --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --description new-description

```

Example 62: Update the properties of a list by specifying data in a JSON file

The following command updates the values for the properties of a list that matches the specified list ID. The properties to be updated are defined in a specified JSON file.

Note: You can include an absolute or relative path to the file name.

```

sas-viya listdata update list --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --file file.json

## Contents of file.json
{
  "name": "new-name",
  "description": "new-description",
  "state": "deployed",
  "label": "new-label",
}

```

Example 63: Update the immutable property of a list

The following command updates the value for the immutable property of a list that matches the specified list ID.

```

sas-viya listdata update list --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --immutable true

```

Example 64: Update the label property of a list

The following command updates the value for the label property of a list that matches the specified list ID.

```

sas-viya listdata update list --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --label new-label

```

Example 65: Update the name property of a list

The following command updates the value for the name property of a list that matches the specified list ID.

```
sas-viya listdata update list --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --name new-name
```

Example 66: Update the data mask for a list's columns by specifying data in a JSON file (privileged)

The following command uses the privileged permission to update the data mask for specified columns of a list that matches the specified list ID. The data masks to be updated are defined in a specified JSON file.

Note: You can include an absolute or relative path to the file name.

```
sas-viya listdata update list --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --file file.json --privileged
```

```
## Contents of file.json
{
  "name": "updated-nonempty-list",
  "columns": [
    {
      "name": "userId",
      "dataType": "string",
      "position": 1,
      "isKey": true,
      "keyPosition": 1,
      "dataMask": {
        "name": "Mask_All",
        "example": "*****",
        "regex": ".",
        "replace": "*"
      }
    }
  ]
}
```

Example 67: Update the expiry for a list's contents to be a number of seconds in the future by specifying data in a JSON file

The following command updates the expiry for specified contents in a list that matches a particular ID. The expiry is updated to be one hour in the future as the number of seconds. The contents to be updated are defined in a specified JSON file.

```
sas-viya listdata expiry in --id 8b36c2c4-10fa-4b52-8ffb-a1b6db474f10
--file file.json --expiry 3600
```

```
## Contents of file.json, where userId is the only key column in list
8b36.
{
```

```

    "items": [
      {
        "userId": "123456789"
      },
      {
        "userId": "123456788"
      },
      {
        "userId": "123456787"
      },
      {
        "userId": "123456786"
      },
    ]
  }

```

Example 68: Update the expiry for a list's contents to be an exact time by specifying data in a JSON file

The following command updates the expiry for specified contents in a list that matches a particular ID. The expiry is updated to be on December 19, 2023 at 2:30 pm as UNIX Epoch time or as UTC time. The contents to be updated are defined in a specified JSON file.

```

sas-viya listdata expiry at --id 8b36c2c4-10fa-4b52-8ffb-a1b6db474f10
--file file.json --expiry "1702996200"

sas-viya listdata expiry at --id 8b36c2c4-10fa-4b52-8ffb-a1b6db474f10
--file file.json --expiry "2023-12-19 14:30:00"

## Contents of file.json, where userId is the only key column in list
8b36.
{
  "items": [
    {
      "userId": "123456789"
    },
    {
      "userId": "123456788"
    },
    {
      "userId": "123456787"
    },
    {
      "userId": "123456786"
    },
  ]
}

```

Example 69: Update the expiry for a list's contents to be an exact time by specifying data in a JSON string

The following command updates the expiry for specified contents in a list that matches a particular ID. The expiry is updated to be on December 19, 2023 at 2:30 pm UTC. The contents to be updated are defined in a JSON string.

```
sas-viya listdata expiry at --id 8b36c2c4-10fa-4b52-8ffb-a1b6db474f10
--data '{"items": [{"userId": "123456789"}, {"userId": "123456788"}]}'
--expiry "2023-12-19 14:30:00"
```

Example 70: Update the state property of a list

The following command updates the value for the state property of a list that matches the specified list ID.

```
sas-viya listdata update list --id 8b36c2c4-10fa-4b52-8ffb-
a1b6db474f10 --state deployed
```

rtdmobjectmigration Plug-In

Imports SAS Real-Time Decision Manager objects that have been extracted from metadata in a SAS 9.4 environment into a SAS Viya environment.

Requirements: You must use the SAS Customer Intelligence `sasmaextract` extraction utility to create the extraction file that contains the objects that you want to import. For instructions, see [“Transfer Treatments from SAS 9.4 to SAS Viya” in SAS Intelligent Decisioning: Administrator’s Guide](#).

You must create a profile and sign in before you use the CLI. See [“Creating and Using Profiles” on page 2](#) for more information.

See: [“Global Options” on page 4](#)
[“Global Commands” on page 3](#)

Syntax

```
sas-viya <global-options> rtdmobjectmigration start <command-options>
```

Commands

start

starts the SAS Real-Time Decision Manager migration command-line utility. The start command is required.

--folder *folder-GUID*

specifies the globally unique identifier (GUID) for the folder into which you want to import the objects. The folder option is optional.

Alias **-f**

Default **Public**

--input *filename.xml*

specifies the XML file that contains the objects that were extracted from the SAS Metadata Repository on SAS 9.4. This file is the output file that is

created by the `sasmaextract` utility. This option is required. For more information, see [“Transfer Treatments from SAS 9.4 to SAS Viya” in SAS Intelligent Decisioning: Administrator’s Guide](#).

Alias `-i`

--type *object-type*

specifies the object type to import. Currently, the only object type that is supported is **treatment**.

Alias `-t`

Details

Transfer Objects from SAS 9.4 to SAS Viya

- 1 Use the `sasmaextract` utility that is included with SAS Customer Intelligence to extract objects from the SAS Metadata Repository on SAS 9.4. Then, copy the output file from that utility to a location that is accessible from SAS Viya. The output file that this utility creates is the input file for this CLI. For instructions, see [“Transfer Treatments from SAS 9.4 to SAS Viya” in SAS Intelligent Decisioning: Administrator’s Guide](#).

- 2 Log on to the machine where SAS Viya is running. For example, on Linux systems, you can use the `ssh` command:

```
ssh -y machine-name
```

- 3 Use the `cd` command to change to the directory where you downloaded the plug-in files.

- 4 Create a default profile if you have not already done so:

```
sas-viya profile init
```

The CLI prompts you for the target URL (service endpoint) where SAS Viya is running, for the output type that you want, and whether you want to enable color output. The target URL is the destination to which you want to import the objects. For more information, see [“Creating and Using Profiles” on page 2](#).

- 5 Log on to the SAS Viya environment:

```
sas-viya auth login
```

The CLI prompts you for the user ID and password for the SAS Viya environment.

- 6 Import the objects into the SAS Viya environment:

```
sas-viya rtdmobjectmigration start
--input sasmaextract-output-file
--folder folder-GUID
--type treatment
```

If an object of the same name as the object in the extract file already exists in the destination folder, the CLI does not import the object.

The CLI displays the results of the import process. For example:

```
Total objects migrated: 6
Succeeded: 5
Failed: 0
Completed with errors: 0
Skipped due to name conflict: 1
```

Note: When you view imported content in SAS Intelligent Decisioning, any errors in the content are marked with the error icon .

7 Log off from the SAS Viya environment:

```
sas-viya auth logout
```

Example

The following examples use the default profile. For information about defining and specifying a profile, see [“Creating and Using Profiles” on page 2](#).

The following command imports the treatments that are in the file **treatments_extracted.xml** and writes the treatments to the folder with the ID **a750925f-2749-4c8c-94c6-dd93a471d427**:

```
sas-viya rtdmobjectmigration start
--input /extractFiles/treatments_extracted.xml
--folder a750925f-2749-4c8c-94c6-dd93a471d427
--type treatment
```

scoreexecution Plug-In

Lists or deletes resources such as log files, code files, jobs, and SAS Cloud Analytic Services (CAS) tables that were not deleted when the associated rule set, decision, or model test was deleted.

Requirement: You must create a profile and sign in before you use the CLI. See [“Creating and Using Profiles” on page 2](#) for more information.

See: [“Global Options” on page 4](#)
[“Global Commands” on page 3](#)

Syntax

```
sas-viya <global-options> scoreexecution command <command-options>
```

Commands

In addition to the global commands in [“Global Commands” on page 3](#), you can specify the following commands:

list-hanging-resources

lists the resources that are no longer used by the score execution service.

--file *filename*

specifies the file to write the list of resources to. You can use this file with the `remove-hanging-resources` command to delete the resources.

Alias `-f`

remove-hanging-resources

deletes resources that are no longer used by the score execution service.

--file *filename*

Specifies the file that contains the URIs of the resources that you want to remove.

Alias `-f`

Examples

The following examples use the default profile. For information about defining and specifying a profile, see [“Creating and Using Profiles” on page 2](#).

Example 1

The following command displays detailed information about the unused resources:

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

Example 2

The following command displays information about unused resources in a table format:

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

Example 3

The following command writes the URIs for unused resources to a file named `uris.txt`:

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

Example 4

The following command deletes the unused resources that are listed in the file `uris.txt`:

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

sid-functions Plug-In

Creates and manages custom functions and function categories in SAS Intelligent Decisioning.

Requirement: You must create a profile and sign in before you use the CLI. See [“Creating and Using Profiles” on page 2](#) for more information.

See: [“Global Options” on page 4](#)
[“Global Commands” on page 3](#)

Syntax

sas-viya *<global-options>* **sid-functions** *command* *<command-options>*

Commands

In addition to the global commands in [“Global Commands” on page 3](#), you can specify the following commands:

add *<- -replace>* **category-name** **path-to-DS2-input-file**

adds the function (method) that is defined in the input file to the specified category. If you specify the `- -replace` option, the function definition in the input file replaces the current definition of the function with the same name. Function names must be unique within an environment or tenant and are limited to 250 characters. Function names must be valid DS2 identifiers. For more information, see [“DS2 Identifiers” in SAS DS2 Programmer’s Guide](#).

categories **subcommand** *<arguments>*

creates and manages custom function categories. Category names must be unique within an environment or tenant and are limited to 250 characters. For information about each subcommand, see [“Subcommands for the categories Command” on page 55](#).

delete **function-name**

deletes the specified function. If the function does not exist, this command has no effect.

download-code

downloads the code for all of the custom functions in the environment and sends the output to stdout.

list

lists the names and parameters for all of the custom functions that are defined in the environment.

Subcommands for the categories Command

create <options> category-name**- -description "category-description"**

specifies a description for the category.

- -replace category-name <category-description>

enables you to add, delete, or modify the description of a category. To delete the existing description of a category, omit the - -description option:

```
sas-viya sid-functions --replace myCategory
```

delete category-name

deletes the specified category and all of the functions in that category.

help

displays help information for the categories command.

list

lists the custom function categories that are defined in the environment.

show category-name

displays the properties of the specified category.

Details

Using Custom Functions

You can use custom functions to perform actions that are not available with the standard functions that SAS provides. Custom functions also enable you to encapsulate and reuse business logic. You can define the method for the function in a DS2 file, add that file to SAS Intelligent Decisioning as a custom function, and use the function in rule sets and in DS2 code files.

A custom function DS2 file can contain only one method definition. The name of the method becomes the name of the custom function. The name cannot be the same as the name of an existing SAS function. For example, your custom function file might define a custom function named `square` that calculates the square of a number:

```
method square(double value) returns double;
    return value * value;
end;
```

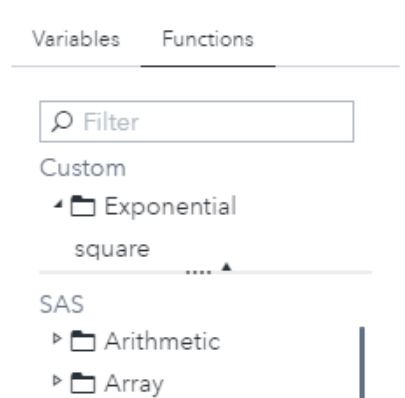
The method signature can include variables that are both input and output variables. Specify these variables with the `in_out` keyword. Custom functions support data grid packages and the DS2 data types that are listed in [“Data Types for SAS Data Sets” in SAS DS2 Language Reference](#).

Before you add a custom function to SAS Intelligent Decisioning, test the function code by including it in a DS2 custom code file, including the code file in a decision, and running a scenario test. For example, you can test the `square` function with the following custom code file:

```
package "${PACKAGE_NAME}" /inline;
  method square(double value) returns double;
    return value * value;
  end;
  method execute(double value, in_out double result);
    result = square(value);
  end;
endpackage;
```

For more information, see [“Ways to Test a Code File”](#) in *SAS Intelligent Decisioning: User’s Guide* and [“Test a Scenario”](#) in *SAS Intelligent Decisioning: User’s Guide*.

When you add a function category and custom function to SAS Intelligent Decisioning, this CLI adds the category and function to the list of functions in the expression editor. Custom categories appear as subcategories under the category Custom. For example, if you add a function category named Exponential and a function named `square`, these appear above the SAS functions in the expression editor:



IMPORTANT A delay of up to one minute can occur between the time when you define or modify (replace) a function and the time when the function is available to be executed.

When SAS Intelligent Decisioning generates code, it inserts the custom function definition ahead of the code that it generates for the rule set logic or code file logic. In the rule set or code file, you can reference custom functions in the same way that you reference functions that are provided by SAS.

Examples

The following examples use the default profile. For information about defining and specifying a profile, see [“Creating and Using Profiles”](#) on page 2.

Example 1

The following command creates the custom function category named Exponential:

```
sas-viya sid-functions categories create "Exponential"
```

Example 2

The following command adds a description to the existing category Exponential:

```
sas-viya sid-functions categories create --replace  
--description "Custom exponential functions" "Exponential"
```

Example 3

The following command adds the function defined in the squareFx.ds2 file to the category named Exponential:

```
sas-viya sid-functions add "Exponential"  
"myCustomFunctions/squareFx.ds2"
```

Example 4

To modify the code for an existing custom function, use the `--replace` option. The following command replaces the custom function that was defined in the previous example with the updated contents of the squareFx.ds2 file:

```
sas-viya sid-functions add --replace "Exponential"  
"myCustomFunctions/squareFx.ds2"
```

Example 5

The following command lists the names and parameters for all of the custom functions that are defined in the environment:

```
sas-viya sid-functions list
```

Example 6

The following command deletes the custom function named square:

```
sas-viya sid-functions delete "square"
```

Example 7

The following command deletes the custom function category Exponential and deletes all of the custom functions in that category:

```
sas-viya sid-functions categories delete "Exponential"
```

Example 8

The following command downloads the code for all of the custom functions that are defined in the environment and writes that code to the file name customFunctions:

```
sas-viya sid-functions download-code > customFunctions
```

Transferring Version Comments

Transfer Version Comments 59

Transfer Version Comments

To transfer content from SAS Decision Manager 5.x or SAS Intelligent Decisioning 5.x on Viya 3.5 to the latest release of SAS Intelligent Decisioning on SAS Viya 4, use the SAS Administrative CLI (sas-admin) on Viya 3.5 to export the content. Use the SAS Viya Command-Line (sas-viya) CLI to import the content on the target environment. For information about these CLIs, see [SAS Viya 3.5 Administration: Promotion \(Import and Export\)](#) and “How To (CLI)” in [SAS Viya Platform: Content Migration from SAS Viya 4](#).

All of an object’s content is transferred except for notes that are associated with specific versions of the objects. To transfer the version notes for an object, complete these steps:

- 1 Export a transfer package from the source environment that contains the version notes that are associated with the object:

```
sas-admin transfer export
--resource-uri /comments/comments?filter=startsWith(resourceUri,'object-uri')
```

The URI for the current object version is displayed on the **Properties** tab for the object in SAS Intelligent Decisioning. You can click  at the right of the URI to copy it. The object URI is the first part of this version URI, up to but not including /revisions/revision_ID. Alternatively, you can retrieve object URIs by using the SAS Intelligent Decisioning REST API. Object URIs have the following format:

```
/businessRules/ruleSets/rule-set-ID
/decisions/flows/decision-ID
/referenceData/domains/lookup-table-ID
/treatmentDefinitions/definitions/treatment-ID
/treatmentDefinitions/definitionGroups/treatment-group-ID
```

Record the transfer package ID.

- 2 Download the package to your local machine, and store it in a JSON file named `MyPackage.json`:

```
sas-admin transfer download --id transfer-package-ID --file /tmp/MyPackage.json
```

- 3 Upload the JSON file to the target environment:

```
sas-viya transfer upload --file /tmp/MyPackage.json
```

Record the upload package ID.

- 4 Import the version comments into the target environment:

```
sas-viya transfer import --id upload-package-ID
```