

# SAS<sup>®</sup> Intelligent Decisioning: Command- Line Interfaces

2021.1.2 - 2021.1.5\*

\* 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. 2021. *SAS® Intelligent Decisioning: Command-Line Interfaces*. Cary, NC: SAS Institute Inc.

**SAS® Intelligent Decisioning: Command-Line Interfaces**

Copyright © 2021, 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

October 2022

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\_003-P1:edmccli

---

# Contents

<b>Chapter 1 / Using SAS Intelligent Decisioning Command-Line Interfaces</b> .....	<b>1</b>
About the SAS Intelligent Decisioning CLIs .....	1
Downloading and Installing the CLIs .....	2
Creating and Using Profiles .....	2
<b>Chapter 2 / SAS Intelligent Decisioning CLIs</b> .....	<b>3</b>
Global Commands .....	3
Global Options .....	4
Dictionary .....	6
<b>Chapter 3 / Transferring Version Comments</b> .....	<b>29</b>
Transfer Version Comments .....	29



# 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: 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.

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

# SAS Intelligent Decisioning CLIs

---

<b>Global Commands</b> .....	<b>3</b>
<b>Global Options</b> .....	<b>4</b>
<b>Dictionary</b> .....	<b>6</b>
dcmtransfer Plug-In .....	6
decisiongitdeploy Plug-In .....	16
rtdmobjectmigration Plug-In .....	19
scoreexecution Plug-In .....	22
sid-functions Plug-In .....	23

---

## 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 [“Global Command: Profile” in SAS Viya: Using the Command-Line Interface](#).

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

[“Command-Line Interface: Preliminary Instructions” in SAS Viya: 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**      [“Global Command: Profile” in SAS Viya: Using the Command-Line Interface](#)

[“Command-Line Interface: Preliminary Instructions” in SAS Viya: 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

### **--insecure**

allows connections to TLS sites without validating the server certificates.

**Alias**    -k



**--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](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: 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:** 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.

To transfer content between SAS Viya environments, use the transfer plug-in to the sas-viya CLI. All of the content is transferred except for notes that are associated with specific versions of the objects. For more information, see [“Command-Line Interface: Overview” in SAS Viya: Using the Command-Line Interface](#) and [“Transfer Version Comments” on page 29](#).

**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 7](#)

[“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 “Details” in [SAS Viya: 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 11](#)

**--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 “Details” in [SAS Viya: 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: Preliminary Instructions” in SAS Viya: 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:

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

- 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: 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 10](#), 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	<code>name="rule-set-name"</code>	Rule sets that match the specified name. This parameter is case sensitive.
	<code>vocabularyName="vocabulary-name"</code>	Rule sets that use the specified vocabulary
Rule flows	<code>name="rule-flow-name"</code>	Rule flows that match the specified name. This parameter is case sensitive.
	<code>simpleOnly="true"</code>	Beginning with SAS Intelligent Decisioning 5.1, SAS Intelligent

Object Type	Available Parameters	Objects Returned
		Decisioning does not support complex rule flows.
Decisions	<code>name="decision-name"</code>	Decisions that match the specified name. This parameter is case sensitive.
	<code>vocabularyName="vocabulary-name"</code>	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	<code>source.folderPath</code>	Folder path in the SAS 9.4 environment	No
	<code>target.folderPath</code>	Folder path in the SAS Viya environment. By default, this field is set to the same path as <code>source.folderPath</code> .	Yes
lookups	<code>source.id</code>	Lookup table ID in the SAS 9.4 environment	No
	<code>source.name</code>	Lookup table name	No
	<code>source.folderPath</code>	Lookup folder path in the SAS 9.4 environment	No



Object	Field	Description	Can Be Changed
ruleSets	target.id	Lookup table ID in the SAS Viya environment	Yes
	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
models	target.revisionId	Rule set version ID in the SAS Viya environment. If no version ID is specified, then the revision ID is set to <b>current</b> . When the rule set information is displayed in the user interface, the current version number is displayed.	Yes
	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 18](#).

**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 18](#).

**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: Using the Command-Line Interface](#) and [SAS Viya: 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
```

---

## 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 24.

**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;
```

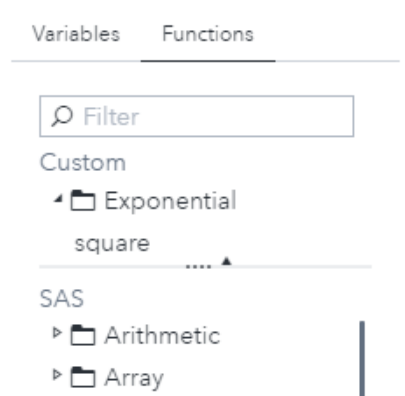
The method signature can include variables that are both input and output variables. Specify these variables with the `in_out` keyword. Custom functions support data grid packages and the DS2 data types that are listed in [“Data Types for SAS Data Sets” in SAS DS2 Language Reference](#).

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

<i>Transfer Version Comments</i> .....	29
--	----

## Transfer Version Comments

When you use the transfer plug-in to the sas-viya CLI to transfer objects between SAS Viya environments, all of the 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-viya transfer export
  --resource-uri /comments/comments?filter=startsWith(resourceUri,'object-uri')
```

The object URI is displayed on the **Properties** tab for the object in SAS Intelligent Decisioning, or 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-viya 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
```

For more information, see [“Transfer Content between SAS Viya 4 Environments” in \*SAS Intelligent Decisioning: Administrator’s Guide\*](#) and [“How To \(CLI\)” in \*SAS Viya: Content Migration from SAS Viya 4\*](#).