SAS® Intelligent Decisioning: Administrator’s Guide

2023.04

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.

Performing Post-installation Tasks

General post-installation tasks for SAS Viya are in the deployment guide for your environment. Deployment guides for SAS Viya are available at https://support.sas.com//en/software/sas-viya.html.

After you install SAS Intelligent Decisioning, complete the following steps:

1 Verify configuration properties.

2 Configure publishing destinations.

3 (Optional) Configure support for SQL query files.

4 (Optional) Configure support for Python code files.

5 (Optional) Set the creation_dttm values in the subject contacts database.

6 (Optional) Set up third-party databases for subject contacts or treatments or both:

   a Configure the subject contacts database or the treatment definition database or both databases.

   b (Optional) Partition the subject contacts database.

7 (Optional) Enable the check-out and commit feature.

8 (Optional) Enable performance logging.

9 (Optional) Enable variable assignment logging.

10 (Optional) Enable the asset approval workflow.

11 (Optional) Transfer content to the new environment.

12 (Optional) Enable lookup tables to be reloaded automatically.

13 Verify that permissions are correct for your site.

Verify Configuration Properties

View or Set Configuration Properties for SAS Intelligent Decisioning Services

After you install SAS Intelligent Decisioning, review the configuration properties to ensure that the values are appropriate for your environment.

1 Sign in to SAS Environment Manager as an administrator.

   Note: If you are already logged in to SAS Intelligent Decisioning as an administrator, access SAS Environment Manager by clicking ☰ and selecting **Manage Environment**.

2 Click 🔧 to navigate to the Configuration category view.

3 In the **View** menu, select **All Services**.

4 Select a service to view its properties. Click ✎ to edit the properties. Verify the property settings for each of the following services:

   - Business Rules

- Decisions
- Micro Analytic Score
- Reference Data
- Score Execution
- Subject Contact
- Treatment Definitions

For more information, see “Configuration Page” in *SAS Environment Manager: User’s Guide* and *SAS Viya Platform: Configuration Properties*.

Business Rules Service Properties

Verify the settings for the following configuration instances:

- “jvm”
- “sas.businessrules”
- “sas.businessrules.checkout”
- “sas.businessrules.publish”

jvm

Click **New Configuration** to define this configuration instance.

jvm.java_option_xmx
Specifies the JVM heap size of the Business Rules service.

Tip You might need to increase the heap size if you are importing very large lookup tables.

sas.businessrules

sas.businessrules.additionalDS2Options
Enables you to specify additional DS2 options that are not specified by other configuration options. Separate multiple options with a space. This option affects basic tests, scenario tests, and publishing validation tests for rule sets.

Default MISSING_NOTE. This setting writes a note to the SAS log when an invalid function argument generates a missing value.

See “DS2_OPTIONS Statement” in *SAS DS2 Language Reference*

Example MISSING_NOTE DIVBYZERO=IGNORE MISSING TYPEWARN

sas.businessrules.bootstrap.refreshPackages
Specifies whether DS2 packages in the Micro Analytic Score service should be refreshed if they already exist.

Default Off

sas.businessrules.dataGridMetadataGenerating

Specifies whether DS2 packages that are published to SAS Micro Analytic Service destinations have two methods (step IDs), `datagrid_metadata` and `datagrid_metadata_skel`, that you can use to determine the data grid metadata that is applicable to the rule flow. When this property is set to On, you can post a request to these steps to get data grid metadata. See Execute a step in the SAS Viya REST API documentation.

Default	Off

sas.businessrules.deleteVersions

Enables users who have permission to delete rule sets to delete specific versions of rule sets.

Default	On

sas.businessrules.execution.threadCount

Specifies the maximum number of threads that can be allocated for executing DS2 code packages.

Default	4

sas.businessrules.inlineGlobalVariableValues

specifies whether the current values of global variables are included inline in generated code when you publish a rule set to CAS or to SAS Micro Analytic Service destinations. When this property is set to On, the current value of each global variable is included in the code that is generated for these destinations. Changes to the values of the variables do not affect published rule sets. When this property is set to Off, the generated code uses a SAS format or a SAS Micro Analytic Service module to retrieve the current value of the global variable when the rule set is run.

Default	On
See	“Activating a Global Variable” in *SAS Intelligent Decisioning: User’s Guide*

sas.businessrules.inputVariableReadOnly

Determines whether the values of character variables that are designated as input-only variables are available to all objects in a decision. When inputVariableReadOnly is set to Off and the value of an input-only character variable is modified in a rule set, the modified value is passed back to the parent decision and is available to the remaining objects in the decision. When this property is set to On, modified values are not passed back to the parent decision and are available only within the rule set in which the value was modified.

Default	Off

sas.businessrules.logicalExpression

Specifies whether SAS Intelligent Decisioning adds the DS2 option LOGICALEXPR= to the code that it generates for rule sets. The LOGICALEXPR= option specifies how logical AND and OR expressions are evaluated. Valid option values are STANDARD and OPTIMIZED.

STANDARD	specifies that DS2 can evaluate both operands of a logical AND or OR expression before returning a result.
OPTIMIZED	specifies that DS2 optimizes the evaluation of the operands of logical AND and OR expressions by returning a result that is based on the first operand when possible. Specifying OPTIMIZED can enable the code to run faster and can simplify the processing of complex rule conditions.

Default	STANDARD

Note	This option applies to both published code and test code.
See	"LOGICALEXPR=STANDARD | OPTIMIZED" in *SAS DS2 Language Reference*

sas.businessrules.lookupStaticBinding

Specifies whether published rule sets and decisions use new versions of lookup tables that were activated after the rule set or decision was published. When this property is set to On, SAS Intelligent Decisioning generates a static version of the active version of the lookup table and embeds it in the code that is generated for the published object. Newly activated versions of lookup tables are ignored by the published objects.

> **IMPORTANT** If you publish an object that uses a lookup table to a Teradata or Hadoop destination, set this option to On. Code that is generated for Teradata and Hadoop destinations work correctly only when lookup tables are included in the generated code.
>
> Lookup tables are automatically embedded in objects published to container destinations. This configuration option is ignored for container destinations.

For more information, see "Controlling Where Lookup Tables Are Activated And How They Are Used" in *SAS Intelligent Decisioning: User's Guide*.

Default	Off

sas.businessrules.messageLimit

Specifies whether SAS Intelligent Decisioning adds the DS2 option MSGLIMIT= to the code that it generates for rule sets. The MSGLIMIT= option specifies the maximum number of error, warning, and note messages that can be written to the SAS log while the code executes.

Default	1024
Restriction	This configuration option is valid only when sas.businessrules.messageOrder is set to TEMPORAL.
See	"MSGLIMIT=n | MIN | MAX" in *SAS DS2 Language Reference*

sas.businessrules.messageOrder

Specifies whether SAS Intelligent Decisioning adds the DS2 option MSGORDER= to the code that it generates for rule sets. The MSGORDER= option specifies whether DS2 writes error, warning, and note messages to the SAS log as they are produced or after the code executes. Valid option values are STANDARD and TEMPORAL.

STANDARD	specifies that DS2 writes the messages after the code executes. PUT statement messages precede the diagnostic messages in the log.
TEMPORAL	specifies that DS2 writes the messages as they are produced. PUT statement messages are interspersed with the diagnostic messages in the log.

Default	STANDARD
Note	This option applies to both published code and test code.
See	"MSGORDER=STANDARD | TEMPORAL" in *SAS DS2 Language Reference*

sas.businessrules.scond

Specifies whether SAS Intelligent Decisioning adds the DS2 option SCOND= to the code that it generates for rule sets. The SCOND= option specifies the level of the messages that are displayed in the SAS log for the DS2 variable declaration strict mode. This mode requires that every variable must be declared in the DS2 program. This option affects basic tests, scenario tests, and publishing validation tests for rule sets. You can specify the following settings:

WARNING	writes warning messages to the SAS log.
NONE	no messages are written to the SAS log.
NOTE	writes notes to the SAS log.
ERROR	writes error messages to the SAS log.

Default WARNING

See “SCOND” in *SAS DS2 Language Reference*

sas.businessrules.variableLengthOverridden

Specifies whether you can override the default length for input-only and input-output character variables in rule sets. When this property is set to Off, the lengths for these variables are based on the length of the input columns to which the variables are mapped. When this property is set to On, SAS Intelligent Decisioning uses the lengths that you specify when you create the variables. For more information, see “Managing the Variables in a Decision” in *SAS Intelligent Decisioning: User’s Guide*.

Default Off

sas.businessrules.checkout

sas.businessrules.checkout.allowConcurrentCheckout

Controls whether the same rule set can be checked-out by multiple users at the same time.

Default ON

See “Concurrently Checking Out and Committing Rule Set Versions” in *SAS Intelligent Decisioning: User’s Guide*

sas.businessrules.checkout.checkoutEnabledFolderPaths

Specifies the list of folder paths whose rule sets can be or must be checked out before they can be edited. Separate path names with commas.

Default **`/Decision Repository`**

Tip Permissions are set on the default folder, `/Decision Repository`, so that users are required to check out the objects in this folder before they can be edited. Similar permissions are not automatically set on any additional folders that you specify with this configuration option. You must explicitly set the permissions in order to require that the objects in the additional folders are checked out before they are modified. See “Set Permissions for Check-Out Folders” on page 59 for more information.

Example `/Public/checkout`

sas.businessrules.publish

sas.businessrules.publish.hideRuleSetPublish

Specifies whether to hide the **Publish** button for rule sets. When this property is turned on, rule sets cannot be published independently from the user interface. They must be included in a decision, and the decision must be published. Rule sets can still be published by using the REST API when this property is turned on.

If you are using the SAS Intelligent Decisioning approval workflow, it is recommended that you turn this option on so that all published content is developed in the workflow.

Default	Off

Decisions Service Properties

Verify the settings for the following configuration instances:

- "jvm"
- "sas.decisions"
- "sas.decisions.checkout"
- "sas.decisions.codefiles"
- "sas.decisions.masnode"
- "sas.decisions.nodetraces"
- "sas.decisions.variable.length"
- "sas.decisions.workflow"
- "sas.subjectcontacts.datagrid"
- "sas.treatmentdefinitions.datagrid"

jvm

Click **New Configuration** to define this configuration instance.

jvm.java_option_xmx
Specifies the JVM heap size of the Decisions service.

sas.decisions

sas.decisions.additionalDS2Options
Enables you to specify additional DS2 options that are not specified by other configuration options. Separate multiple options with a space. This option affects basic tests, scenario tests, and publishing validation tests for decisions.

Default	MISSING_NOTE. This setting writes a note to the SAS log when an invalid function argument generates a missing value.
See	"DS2_OPTIONS Statement" in *SAS DS2 Language Reference*
Example	MISSING_NOTE DIVBYZERO=IGNORE MISSING TYPEWARN

sas.decisions.clearOutputOnlyDataGridSubDecision

Specifies whether existing data is cleared in output-only data grids that are used in subdecisions before the subdecision executes. When this property is set to On, the data in the data grids is cleared. When this property is set to Off, data from previous runs is not cleared in the data grids.

Default Off

sas.decisions.dataGridMetadataGenerating

Specifies whether DS2 packages that are published to SAS Micro Analytic Service destinations have two methods (step IDs), `datagrid_metadata` and `datagrid_metadata_skel`, that you can use to determine the data grid metadata that is applicable to the decision flow. When this property is set to On, you can post a request to these steps to get data grid metadata. See Execute a step in the SAS Viya REST API documentation.

Note: If you enable this property, SAS Intelligent Decisioning generates more code for published objects and might increase the time required to publish an object to SAS Micro Analytic Service destinations. Enabling this option does not significantly affect the time required to execute published objects.

Default Off

sas.decisions.deleteVersions

Enables users who have permission to delete decisions to delete specific versions of decisions.

Default On

sas.decisions.disableLineReporting

Specifies whether line numbers are included in DS2 error messages. When this option is turned off, lines numbers are included in error messages. When this option is turned on, lines numbers are not included in error messages.

Default Off

sas.decisions.execution.threadCount

Specifies the maximum number of threads that can be allocated for executing DS2 code packages.

TIP When the **Enable variable assignment logging** option is selected for a scenario test, this configuration option is ignored. The default thread count for the test is set to 1 in order to avoid duplicate and interleaved log entries. For more information, see "Enable Variable Assignment Logging" on page 39.

Default 4

sas.decisions.includeLoggingInGeneratedCode

Enables all logging, including performance logging, for decisions that are running in the Micro Analytic Score service.

Default On

See "Enabling Performance Logging" on page 36

sas.decisions.includeRuleFiredInformationForSubdecisions

specifies whether rule-fired information is recorded for subdecisions when a decision is tested. This option must be turned on in order to specify the INCLUDESUBDECISIONRULEFIRE=YES option on the %DCM_RULEFIRE_DETAIL macro.

Default	Off
Tip	If you turn on this option, it is recommended that you re-run any existing tests or scenario tests for decisions that contain subdecisions.
See	"%DCM_RULEFIRE_DETAIL Macro" in *SAS Intelligent Decisioning: Macro Guide*

sas.decisions.inlineGlobalVariableValues

specifies whether the current values of global variables are included inline in generated code when you publish a decision to CAS or to SAS Micro Analytic Service destinations. When this property is set to On, the current value of each global variable is included in the code that is generated for these destinations. Changes to the values of the variables do not affect published decisions. When this property is set to Off, the generated code uses a SAS format or SAS Micro Analytic Service module to retrieve the current value of the global variable when the decision is run.

Default	On
See	"Activating a Global Variable" in *SAS Intelligent Decisioning: User's Guide*

sas.decisions.logicalExpression

Specifies whether SAS Intelligent Decisioning adds the DS2 option LOGICALEXPR= to the code that it generates for decisions. The LOGICALEXPR= option specifies how logical AND and OR expressions are evaluated. Valid option values are STANDARD and OPTIMIZED.

STANDARD	specifies that DS2 can evaluate both operands of a logical AND or OR expression before returning a result.
OPTIMIZED	specifies that DS2 optimizes the evaluation of the operands of logical AND and OR expressions by returning a result that is based on the first operand when possible. Specifying OPTIMIZED can enable the code to run faster and can simplify the processing of complex rule conditions.

Default	STANDARD
Note	This option applies to both published code and test code.
See	"LOGICALEXPR=STANDARD | OPTIMIZED" in *SAS DS2 Language Reference*

sas.decisions.masInlineTreatmentGroup

Determines whether the packages for treatment groups that are published to SAS Micro Analytic Service destinations are included inline in the published content.

Default	Off
See	"Content Executed by Published Decisions" in *SAS Intelligent Decisioning: User's Guide*

sas.decisions.messageLimit

Specifies whether SAS Intelligent Decisioning adds the DS2 option MSGLIMIT= to the code that it generates for decisions. The MSGLIMIT= option specifies the maximum number of error, warning, and note messages that can be written to the SAS log while the code executes.

Default	1024

Restriction	This configuration option is valid only when sas.decisions.messageOrder is set to TEMPORAL.
See	“MSGLIMIT=n | MIN | MAX” in *SAS DS2 Language Reference*

sas.decisions.messageOrder

Specifies whether SAS Intelligent Decisioning adds the DS2 option MSGORDER= to the code that it generates for decisions. The MSGORDER= option specifies whether DS2 writes error, warning, and note messages to the SAS log as they are produced or after the code executes. Valid option values are STANDARD and TEMPORAL.

STANDARD	specifies that DS2 writes the messages after the code executes. PUT statement messages precede the diagnostic messages in the log.
TEMPORAL	specifies that DS2 writes the messages as they are produced. PUT statement messages are interspersed with the diagnostic messages in the log.

Default	STANDARD
Note	This option applies to both published code and test code.
See	“MSGORDER=STANDARD | TEMPORAL” in *SAS DS2 Language Reference*

sas.decisions.scond

Specifies whether SAS Intelligent Decisioning adds the DS2 option SCOND= to the code that it generates for decisions. The SCOND= option specifies the level of the messages that are displayed in the SAS log for the DS2 variable declaration strict mode. This mode requires that every variable must be declared in the DS2 program. This option affects basic tests, scenario tests, and publishing validation tests for decisions. You can specify the following values:

WARNING	writes warning messages to the SAS log.
NONE	no messages are written to the SAS log.
NOTE	writes notes to the SAS log.
ERROR	writes error messages to the SAS log.

Default	WARNING
Note	When the decision contains a model, SAS Intelligent Decisioning sets SCOND=NONE before the code for the model package is generated. After the model code is generated, SAS Intelligent Decisioning resets the SCOND= option to the value that is specified by this option.
See	“SCOND” in *SAS DS2 Language Reference*

sas.decisions.taskExecutor.maxThreadsPerRequest

Specifies the maximum number of threads that can be used by SAS Intelligent Decisioning to generate the DS2 code for a decision.

Default	4

sas.decisions.taskExecutor.minItemsPerThread

Specifies the minimum number of items that can be processed inside the same thread when SAS Intelligent Decisioning is generating the DS2 code for a decision.

Default	5

sas.decisions.validation.validationCoreThreadPoolSize

Specifies the maximum number of threads that SAS Intelligent Decisioning can use to validate decisions. If you routinely validate very large decisions, increasing the number of threads available to the validation process can improve performance.

Default	8

sas.decisions.variableLengthOverridden

Specifies whether you can override the default length for input-only and input-output character variables in decisions and code files. When this property is set to Off, the lengths for these variables are based on the length of the input columns to which the variables are mapped. When this property is set to On, SAS Intelligent Decisioning uses the lengths that you specify when you create the variables.

Default	Off
See	"Managing the Variables in a Decision" in *SAS Intelligent Decisioning: User's Guide*

sas.decisions.checkout

sas.decisions.checkout.allowConcurrentCheckout

Controls whether the same decision or code file can be checked-out by multiple users at the same time.

Default	ON
See	"Concurrently Checking Out and Committing Code File Versions" in *SAS Intelligent Decisioning: User's Guide* "Concurrently Checking Out and Committing Decision Versions" in *SAS Intelligent Decisioning: User's Guide*

sas.decisions.checkout.checkoutEnabledFolderPaths

Specifies the list of folder paths whose decisions and code files can be or must be checked out before they can be edited. Separate path names with commas.

Default	**`/Decision Repository`**
Tip	Permissions are set on the default folder, `/Decision Repository`, so that users are required to check out the objects in this folder before they can be edited. Similar permissions are not automatically set on any additional folders that you specify with this configuration option. You must explicitly set the permissions in order to require that the objects in the additional folders are checked out before they are modified. See "Set Permissions for Check-Out Folders" on page 59 for more information.
Example	`/Public/checkout`

sas.decisions.codefiles

Click **New Configuration** to define this configuration instance.

sas.decisions.codefiles.deleteVersions

Enables users who have permission to delete custom code files to delete specific versions of custom code files.

Default	On

sas.decisions.codefiles.quoteStudioQueryIdentifiers

Specifies whether SAS Intelligent Decisioning encloses table and column identifiers in quotation marks when it generates DS2 code for custom data query files.

It is recommended that you turn this option on if you do not have existing query files that rely on table or column names that are not enclosed in quotation marks. For more information, see "Handling Table and Column Names in Data Query Files" in *SAS Intelligent Decisioning: User's Guide*.

Default	Off

sas.decisions.masnode

Click **New Configuration** to define this configuration instance.

sas.decisions.masnode.removeTrailingUnderscoresFromInput

Determines whether underscores are appended to input-output and input-only variable names that are passed to the SAS Micro Analytic Service when a Micro Analytic Module node in a decision is executed. If this property is set to Off, SAS Intelligent Decisioning adds an underscore to the names of the variables. If this property is set to On, SAS Intelligent Decisioning does not add underscores.

Default	Off
Requirement	The setting of this configuration property must match the setting of the service.removetrailingunderscoresfrominputs property for the SAS Micro Analytic Service.
Note	When both this property and the service.removetrailingunderscoresfrominputs property are set to On, publishing validation tests will fail for content that has been published to a SAS Micro Analytic Service destination.
See	"sas.microanalyticservice.system: supplementalProperties" in *SAS Micro Analytic Service: Programming and Administration Guide*
	"Executing Content Published to SAS Micro Analytic Service Destinations" in *SAS Intelligent Decisioning: User's Guide*

sas.decisions.nodetraces

Click **New Configuration** to define this configuration instance.

sas.decisions.nodetraces.includeRuleFiredPathTrackInfoInVariableAssignmentLogging

Determines whether rule-fired information and path-tracking information is included in the log when variable assignment logging is enabled.

Default	On
Tip	Turning this option off reduces the size of the log and the space that is required to store it.
See	"Enabling Variable Assignment Logging" on page 38

"Using Variable Assignment Logging" in *SAS Intelligent Decisioning: User's Guide*

sas.decisions.variable.length

Click **New Configuration** to define this configuration instance.

honorOutputLengthInMAS
Specifies whether output-only decision variables of type Character honor the length that is specified in the user interface. This option affects only the code that is generated for container destinations and for SAS Micro Analytic Service destinations.

> **IMPORTANT** When this option is turned on and the value of an output variable is longer than the length specified by this option, the output value is truncated. When this option is turned off or is not defined, the length specified in the user interface is ignored in container destinations and in SAS Micro Analytic Service destinations.

sas.decisions.workflow

sas.decisions.workflow.authorMayApprove
Specifies that a user who is a member of both the SIDWFAuthor and the SIDWFReviewer custom user groups can approve work that they authored. If a user is not a member of the SIDWFReviewer group, then that user cannot approve decisions. This property is ignored if the sas.decisions.workflow.enabled property is turned off.

Default Off

See "Define Asset Approval Workflow User Groups" on page 57

sas.decisions.workflow.enabled
Specifies whether decisions are developed in the SAS Intelligent Decisioning approval workflow. When this property is turned on, a new instance of the workflow is started when a new decision or a new version of an existing decision is created. Decision versions that existed before the workflow was enabled are not automatically put into the workflow.

Default Off

See "Enable the Asset Approval Workflow" on page 41

"Using SAS Workflow with SAS Intelligent Decisioning" in *SAS Intelligent Decisioning: User's Guide*

sas.subjectcontacts.datagrid

sas.subjectcontacts.datagrid.length
Specifies the default length of the serialized data grid, including the JSON syntax, that is generated for subject contacts. Specifying a shorter length can reduce memory usage and might improve performance.

Default 10,485,760

See	"Introduction to Data Grids" in *SAS Intelligent Decisioning: Using Data Grids*

sas.treatmentdefinitions.datagrid

sas.treatmentdefinitions.datagrid.length
Specifies the default length of the serialized data grid, including the JSON syntax, that is generated for treatment groups. Specifying a shorter length can reduce memory usage and might improve performance.

Default	10,485,760
See	"Introduction to Data Grids" in *SAS Intelligent Decisioning: Using Data Grids*

Micro Analytic Score Service Properties

sas.microanalyticservice.system.asychronousexecution
Controls asynchronous communication between the SAS Micro Analytic Service and the subject contact service. For more information, see "sas.microanalyticservice.system: asynchronousexecution " in *SAS Micro Analytic Service: Programming and Administration Guide*.

sas.microanalyticservice.system.historyscheduler
Controls the extraction and publishing of subject contact history records. For information, see "sas.microanalyticservice.system: historyscheduler" in *SAS Micro Analytic Service: Programming and Administration Guide*.

Reference Data Service Properties

Verify the settings for the following configuration instances:

- "sas.referencedata"
- "sas.referencedata.casformats"
- "sas.referencedata.checkout" on page 17
- "sas.referencedata.publish"

sas.referencedata

sas.referencedata.activation.activateGlobalVariableOnImport
Specifies whether a global variable that was active in the source environment is automatically activated in the target environment when it is imported by using the SAS Viya CLI (sas-viya)SAS administrative CLI (sas-admin).

Default	Off
Note	Turning this option on might increase the time required to import content.

sas.referencedata.activation.activateLookupOnImport

Specifies whether a lookup table that was active in the source environment is automatically activated in the target environment when it is imported by using the SAS Viya Command-Line (sas-viya)SAS administrative CLI (sas-admin).

Default	Off
Note	Turning this option on might increase the time required to import content.

sas.referencedata.activation.destinations

Specifies the SAS Micro Analytic Service publishing destinations in which lookup tables are activated. Only one SAS Micro Analytic Service destination is supported on `localhost`. By default, that destination is named maslocal.

This property is ignored if no SAS Micro Analytic Service publishing destinations exist.

Default	maslocal
See	"Activating a Lookup Table" in *SAS Intelligent Decisioning: User's Guide*
	"Controlling Where Lookup Tables Are Activated And How They Are Used" in *SAS Intelligent Decisioning: User's Guide*

sas.referencedata.activation.timeout

Specifies the time limit in seconds after which an attempt to activate a lookup table or a global variable is considered to have failed.

Default	3600

sas.referencedata.deleteVersions

Enables users who have permission to delete lookup tables to delete specific versions of lookup tables.

Default	On

sas.referencedata.casformats

sas.referencedata.casformats.backupLibrary

Specifies the CAS library in which to store a backup of the formats library that is identified in the formatsLibrary property.

Default	Formats

sas.referencedata.casformats.formatsLibrary

Specifies the name of the formats library that contains the CAS formats that are associated with production lookup tables. If you change this property, and you have previously modified the **cas-shared-default: startup** configuration instance to reload lookup tables, you must update the configuration instance to match this property.

Default	userformats3
See	"Reloading Lookup Tables When CAS Is Restarted" on page 63

sas.referencedata.checkout

sas.referencedata.checkout.allowConcurrentCheckout

Controls whether the same lookup table can be checked-out by multiple users at the same time.

Default	ON
See	"Concurrently Checking Out and Committing Lookup Table Versions" in *SAS Intelligent Decisioning: User's Guide*

sas.referencedata.checkout.checkoutEnabledFolderPaths

Specifies the list of folder paths whose lookup tables can be or must be checked out before they can be edited. Separate path names with commas. This option is available starting with 2021.1.6.

Default	**`/Decision Repository`**
Tip	Permissions are set on the default folder, `/Decision Repository`, so that users are required to check out the objects in this folder before they can be edited. Similar permissions are not automatically set on any additional folders that you specify with this configuration option. You must explicitly set the permissions in order to require that the objects in the additional folders are checked out before they are modified. See "Set Permissions for Check-Out Folders" on page 59 for more information.
Example	`/Public/checkout`

sas.referencedata.publish

sas.referencedata.publish.lookupDisableMasPublish

Controls whether lookup tables are activated in SAS Micro Analytic Service destinations. When this option is set to Off and you activate a lookup table, it is activated in all of the SAS Micro Analytic Service destinations that are defined at your site. If this option is set to On, then lookup tables are not activated in these destinations, and your administrator must set the sas.businessrules.lookupStaticBinding option to On in order to include static copies of lookup tables in generated code and to ensure that published content executes correctly.

Default	Off
See	"Controlling Where Lookup Tables Are Activated And How They Are Used" in *SAS Intelligent Decisioning: User's Guide*

Score Execution Service Properties

sas.scoreexecution.deleteExecutions

Specifies whether existing test results for a rule set, model, or decision test are deleted before the test is re-run. By default, existing results are not deleted when a test is re-run.

Default	Off
See	"Managing Test Data" on page 64

Subject Contact Service Properties

Verify the settings for the following configuration instances:

- "jvm"
- "sas.subjectcontacts"

jvm

Click **New Configuration** to define this configuration instance.

jvm.java_option_hibernate_extra_table
Specifies the table type for partitioned PostgreSQL tables. Specify`-Dhibernate.hbm2dll.extra_physical_table_types=PARTITIONED TABLE`. If you partition the subject contacts table, you must specify this option in order for the SubjectContacts service to restart correctly.

Note: This option has no effect if your database is not partitioned or if you are using Oracle for your database.

jvm.java_option_xmx
Specifies the JVM heap size of the Subject Contact service.

sas.subjectcontacts

Click **New Configuration** to define this configuration instance.

sas.subjectcontacts.nodetraces.maxLifeTime
Specifies whether to automatically delete the node trace information in the subject contacts database that is older than the maximum lifetime specified by the nodetraces.maxLifeTimeDays setting.

Default Off

sas.subjectcontacts.nodetraces.maxLifeTimeDays
Specifies the maximum number of days that you want to retain the node trace information in the subject contacts database. If the nodetraces.maxLifeTime configuration setting is set to On, then node trace information that is older than this number of days is automatically deleted.

Default 7

Treatment Definition Service Properties

Verify the settings for the following configuration instances:

- "sas.treatmentdefinitions"

- "sas.treatmentdefinitions.checkout" on page 20
- "sas.treatmentdefinitions.datagrid"
- "sas.treatmentdefinitions.import"

sas.treatmentdefinitions

sas.treatmentdefinitions.activation.activateTreatmentGroupOnImport

Specifies whether the version of a treatment group that was active in the source environment is automatically activated in the target environment when it is imported by using the SAS Viya Command-Line (sas-viya)SAS administrative CLI (sas-admin).

> **IMPORTANT** When this option is set to On, you must transfer the treatments that are used in the treatment group before you transfer the group. If you do not transfer the individual treatments first, attempts to transfer the treatment group will fail when the group tries to activate itself. Because you must transfer objects in a specific order, the recommended setting is Off.

Default	Off
Note	Turning this option on might increase the time required to import content.
See	"sas.treatmentdefinitions.import.resetActiveVersionOfTreatmentGroupOnImport" on page 20
	"Activate a Treatment Group" in *SAS Intelligent Decisioning: User's Guide*

sas.treatmentdefinitions.activation.destinations

Specifies the SAS Micro Analytic Service publishing destinations in which treatment groups are activated. Only one SAS Micro Analytic Service destination is supported on `localhost`. By default, that destination is named maslocal.

This property is ignored if no SAS Micro Analytic Service publishing destinations exist.

Default	maslocal
See	"Activate a Treatment Group" in *SAS Intelligent Decisioning: User's Guide*

sas.treatmentdefinitions.activation.timeout

Specifies the time limit in seconds after which an attempt to activate a treatment group is considered to have failed.

Default	3600

sas.treatmentdefinitions.deleteVersions

Enables users who have permission to delete treatments or treatment groups to delete specific versions of treatments or treatment groups.

Default	On

sas.treatmentdefinitions.checkout

sas.treatmentdefinitions.checkout.allowConcurrentCheckout

Controls whether the same treatment or treatment group can be checked-out by multiple users at the same time.

Default	ON
See	"Concurrently Checking Out and Committing a Treatment or Treatment Group Versions" in *SAS Intelligent Decisioning: User's Guide*

sas.treatmentdefinitions.checkout.checkoutEnabledFolderPaths

Specifies the list of folder paths whose treatments and treatment groups can be or must be checked out before they can be edited. Separate path names with commas.

Default	**`/Decision Repository`**
Tip	Permissions are set on the default folder, `/Decision Repository`, so that users are required to check out the objects in this folder before they can be edited. Similar permissions are not automatically set on any additional folders that you specify with this configuration option. You must explicitly set the permissions in order to require that the objects in the additional folders are checked out before they are modified. See "Set Permissions for Check-Out Folders" on page 59 for more information.
Example	`/Public/checkout`

sas.treatmentdefinitions.datagrid

sas.treatmentdefinitions.datagrid.length

Specifies the default length of the serialized data grid, including the JSON syntax, that is generated for treatment groups. Specifying a shorter length can reduce memory usage and might improve performance. For more information, see "Introduction to Data Grids" in *SAS Intelligent Decisioning: Using Data Grids*.

Default	10,485,760

sas.treatmentdefinitions.import

sas.treatmentdefinitions.import.resetActiveVersionOfTreatmentGroupOnImport

Specifies whether the status of a treatment group is set to Active when it is imported by using the SAS Viya Command-Line (sas-viya)SAS administrative CLI (sas-admin). This option is useful when you are importing a treatment group that was already active when it was exported. The treatment group is not reactivated. Only the status is changed.

Default	Off
See	"Activate a Treatment Group" in *SAS Intelligent Decisioning: User's Guide*
	"sas.treatmentdefinitions.activation.activateTreatmentGroupOnImport" on page 19

Properties for Third-Party Subject Contacts and Treatment Definition Databases

About the Databases and Schemas

By default, the subject contacts and treatment definitions are stored in the PostgreSQL instance that was installed with SAS Viya. You can use Oracle or a different instance of PostgreSQL for these databases.

The best practice for configuring third-party databases for subject contacts and treatment definitions is to use different schemas (owners and passwords) for each database. SAS Intelligent Decisioning uses Liquibase to manage a database changelog. Liquibase uses this changelog to determine how to update the database on your behalf during installation and updates. If you use a different schema for each database, then the database changelog tables that are used by Liquibase are not shared. Using different schemas makes it easier to destroy all of the tables that are associated with a service in case you need to re-create a database.

An Oracle user has only one schema, and it has the same name as the user. To use different schemas in Oracle, different users must own the databases.

Third-Party Database Properties

Note: SAS Intelligent Decisioning provides database drivers for PostgreSQL and Oracle within the containers for each database. You cannot customize the database drivers.

To use a third-party database, you must add the configuration properties for the appropriate service in SAS Environment Manager.

Note: Before you configure a third-party database for subject contacts or treatment definitions, verify that the SharedServices database and the schemas for your third-party databases (subjectcontacts or treatmentdefinitions) are already configured with the correct ID and privileges.

1 Sign in to SAS Environment Manager as an administrator.

   Note: If you are already logged in to SAS Intelligent Decisioning, access SAS Environment Manager by clicking ☰ and selecting **Manage Environment**.

2 Click 🔧 to navigate to the Configuration category view.

3 In the **View** menu, select **Definitions**.

4 Complete these steps for each definition listed in Table 1 on page 22:

   a Select the definition. Definitions are listed in Table 1 on page 22.

   b Click **New Configuration**. The New Configuration window appears. By default, the new configurations apply to both the Subject Contacts service and the Treatment Definition service.

c (Optional) Click ✎ and remove a service from the list of services to which this new configuration applies.

d For the remaining properties, enter the appropriate values from Table 1 on page 22.

e Click **Save** to save the definition. SAS Environment Manager displays the ID of the configuration instance.

f Record the ID of the configuration instance. If the third-party configuration is not specified correctly, the service will not start. In this case, you might need to remove the third-party configuration instances and re-configure the database. For more information, see “Delete Third-Party Database Configurations” on page 33.

Table 1 *Third-Party Database Properties*

Definition	Property	Valid Values	Description
application	schema	For PostgreSQL, specify `subjectcontacts` or `treatmentdefinitions`. For Oracle, specify the user ID that will be used to log in to the database.	Specifies the database schema. **Note:** For PostgreSQL, you cannot use a custom schema.
sas.datasource.initializer	enabled	true false	Specifies whether to initialize the third-party database that is specified by the spring.datasource definition. The default setting for the internal PostgreSQL database is True. For Oracle and third-party PostgreSQL databases, specify False.
sas.verify.resource	database	true false	Enables the third-party database that is specified by spring.datasource to be verified. When this resource is set to True, a test connection pool is created. The default setting for the internal PostgreSQL database is True. This pool is created automatically for the default PostgreSQL database. For Oracle and third-party PostgreSQL databases, specify False.
spring.datasource	platform	postgresql oracle	Specifies the database type.
	driver-class-name	org.postgresql.Driver oracle.jdbc.OracleDriver	Specifies the data source JDBC driver for the database.

Definition	Property	Valid Values	Description
	username		Specifies the user ID that will be used to log in to the database.
	password		Specifies the password for the user ID specified by the username property.
	url	jdbc:postgresql://*host*:*port*/*database* jdbc:oracle:*driver*:@*server*:*portOracle-service-name*	Specifies the JDBC URI to the database. For example: jdbc:postgresql://mydbhost.com:5432/SharedServices?ApplicationName=subjectContacts&currentSchema=subjectcontacts jdbc:postgresql://mydbhost.com:5432/SharedServices?ApplicationName=treatmentDefinitions&currentSchema=treatmentdefinitions jdbc:oracle:oci:@edmt2.mycomp.com:1521/EDMT2

Properties for Multi-tenancy Environments

After you onboard a tenant, you can set tenant-specific values for the following configuration options for the Decisions service:

- sas.decisions.checkout
- sas.decisions.codefiles
- sas.decisions.masnode
- sas.decisions.nodetraces
- sas.decisions.workflow
- sas.subjectcontacts.datagrid

To set these options, sign in to the tenant SAS Environment Manager as an administrator for the tenant environment, and follow the instructions in "View or Set Configuration Properties for SAS Intelligent Decisioning Services" on page 3.

Note: For the provider tenant, these configuration options are populated in the configuration menu. For all other tenants, these options must be added as a new configuration. For more information about multi-tenancy in SAS Viya, see "Multi-tenancy: Overview" in *SAS Viya: Multi-tenancy*.

Content Security Policy Properties

A Content Security Policy (CSP) provides protection against cross-site scripting, clickjacking, and cross-site leak vulnerabilities. SAS Intelligent Decisioning provides a default CSP that uses the latest secure CSP recommendations.

To override the default CSP:

1. Sign in to SAS Environment Manager as an administrator.

   Note: If you are already logged in to SAS Intelligent Decisioning, access SAS Environment Manager by clicking ☰ and selecting **Manage Environment**.

2. Click 🔧 to navigate to the Configuration category view.
3. In the **View** menu, select **Definitions**.
4. Enter **`commons`** in the filter field, and select `sas.commons.web.security` from the list.
5. Click ✎ beside the SAS Intelligent Decisioning configuration instance in order to edit its properties.
6. Modify the configuration instance properties as needed. See "SAS Intelligent Decisioning" in *SAS Viya Platform: Configuration Properties* for a description of the properties.
7. Click **Save** to save your changes.

Configuring Publishing Destinations

About Configuring Publishing Destinations

You can publish content to destinations on SAS Cloud Analytic Services (CAS), Apache Hadoop, SAS Micro Analytic Service, Teradata, and Git. You can also publish content to container destinations on Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP) and private Docker repositories. By default, a SAS Micro Analytic Service destination named **maslocal** is defined for you. You must configure all other publishing destinations.

You can define and manage publishing destinations by using the following application or interfaces:

- *Publishing Destinations page* in SAS Environment Manager.
- *SAS Viya: Models Command-Line Interface*.
- Model Publish API. For additional information about using the API, see the following resources:
  - Model Publish API examples on GitHub
  - Python examples for defining CAS and Git publishing destinations

- "Configuring Git Publishing Destinations by Using the REST API" on page 25

For information about configuring a publishing destination to support querying a database, see "Configuring Support for Querying Databases" on page 25.

Configuring Git Publishing Destinations by Using the REST API

When you define a Git publishing destination by using the REST API, you can specify whether the code that is generated for the destination is compatible with CAS environments (`CAS`) or with SAS Micro Analytic Service (`MAS`) environments. The code that is generated for an object when it is published can be run only in a destination that matches the destination type for which the code generated. If you deploy a published object to a destination with which it is not compatible, an error occurs when you run the code.

To specify the destination type for which you want SAS Intelligent Decisioning to generate the code that is published to the Git destination, include the `codeGenerationMode` property in the specification for the target destination. Specify `CAS` or `MAS` as the value of the property. For example:

```
targetDestination= {
    "properties":[
        {"name": "codeGenerationMode",
        "value": "CAS"}
    ]
}
```

The default setting is `MAS`. If you do not include this property, the code that is generated for the destination will be compatible with SAS Micro Analytic Service destinations.

Verifying Publishing Destinations

You can use the SAS Intelligent Decisioning tutorials to verify that your publishing destinations are configured properly. For more information, see *SAS Intelligent Decisioning: Tutorials*.

Configuring Support for Querying Databases

You can publish decisions that use data query files, DS2 code files that contain SQL queries, or custom code files that contain data queries to SAS Micro Analytic Service destinations and to container destinations.

For SAS Micro Analytic Service destinations, you must configure SAS Micro Analytic Service support for your database. You can supply the database connection string in code, but SAS recommends that you supply it by using the sas.microanalyticservice.service.connectionstring property for the Micro

Analytic Score service in SAS Environment Manager. For information, see "Database Access with DS2" in *SAS Micro Analytic Service: Programming and Administration Guide*.

In container destinations, only certain database types can be queried by using data query files, DS2 code files that contain SQL queries, or custom context files that contain SQL queries. If a decision requires database access, SAS Container Runtime includes the required database-supported driver when it creates the image. To identify the required database, specify the --databaseDriver property when you create the publishing destination. If you do not specify the --databaseDriver property, the driver for Oracle is included in the image. For information about creating a publishing destination and specifying the --databaseDriver property, see "models Plug-In" in *SAS Viya: Models Command-Line Interface*. For information about configuring database access, see "Configuring a Database Connection" in *SAS Container Runtime: Programming and Administration Guide*.

Configuring Support for Python Code Files

To support decisions that contain custom Python code files, you must enable PyMAS package support. For more information, see "Configuring Python for SAS Micro Analytic Service" in *SAS Micro Analytic Service: Programming and Administration Guide*.

To use Python code files in SAS Intelligent Decisioning, user authentication must be enabled in one of the following ways:

- Users who are working with Python code files must be added to the CASHostAccountRequired custom group. For more information, see the following topics:
  - "The CASHostAccountRequired Custom Group" in *SAS Viya: Identity Management*
  - "Manage Custom Groups" in *SAS Environment Manager: User's Guide*
- The CASALLHOSTACCOUNTS environment variable must be set. For information, see "env.CASALLHOSTACCOUNTS" in *SAS Viya: SAS Cloud Analytic Services*.

The MAS_M2PATH, MAS_PYPATH, and MAS_PYWAIT environment variables must be set. They are usually set during the deployment process. See "Environment Variables" in *SAS Micro Analytic Service: Programming and Administration Guide* for more information.

Additional information is available in the file **`$deploy/sas-bases/docs/configure_python_for_sas_viya.html`**.

Set creation_dttm Values

Note: You do not need to set the creation_dttm values for new installations of SAS Intelligent Decisioning on SAS Viya. This task is required only for databases that were created with SAS Intelligent Decisioning 5.4 and earlier on SAS Viya.

Beginning with SAS Intelligent Decisioning 5.5, the schema for the subject contacts database is converted to a schema that enables you to partition the database when SAS Intelligent Decisioning is installed. The new schema adds a creation_dttm column to the following tables:

- dcm_consideration_treatment
- dcm_custom_treatment
- dcm_object_values
- dcm_object_variables
- dcm_subject_context

When you restart the subject contacts service during deployment, a placeholder value is assigned to the creation_dttm cell in all of the existing rows in these tables. After the service has restarted, you must set the values in these cells to match the values in the creation_dttm column in either the dcm_contact_communication table or the dcm_consideration_treatment table.

> **IMPORTANT** Set the creation_dttm values in the database tables only after you migrate data from SAS Intelligent Decisioning 5.4 and earlier to the latest release of SAS Intelligent Decisioning. You do not need to repeat this task each time the subject contacts service is restarted.

As a Kubernetes cluster administrator, complete the following steps:

1 On the primary node of the Kubernetes cluster, export the KUBECONFIG environment variable, and set the value of the Kubernetes namespace (such as `d9997`):

```
export KUBECONFIG=/etc/kubernetes/admin.conf
namespace=k8s_namespace_value
```

2 Obtain the password to the PostgresSQL instance that was installed with SAS Viya:

```
kubectl -n $namespace exec -it $(kubectl get pods -n $namespace |
    grep sas-consul-server | cut -f 1 -d ' ') -c sas-consul-server
    -- /opt/sas/viya/home/bin/sas-bootstrap-config kv read --recurse config  |
    grep '^config/application/sas-crunchy-data-postgres/password'  | cut -f 2 -d '='
```

3 Export the PostgreSQL password to the PGPASSWORD environment variable. Specify the password that you obtained in Step 2 in the following command:

```
export PGPASSWORD='password'
```

4 Use psql to connect to the database:

```
kubectl -n $namespace exec -it $(kubectl get pods -n $namespace |
    grep sas-crunchy-data-postgres |grep Running|grep -v operator | grep -v stanza |
    grep -v backrest | head  -1| cut -d " " -f 1) -c database
    -- psql -U dbmsowner SharedServices
```

5 Execute the following SQL statements:

```
update subjectcontacts.dcm_consideration_treatment
set creation_dttm = contact.creation_dttm
from subjectcontacts.dcm_contact_communication contact
where dcm_consideration_treatment.contact_id=contact.contact_id;

update subjectcontacts.dcm_custom_treatment
set creation_dttm = consider.creation_dttm
from subjectcontacts.dcm_consideration_treatment consider
where dcm_custom_treatment.consider_id=consider.consider_id;
```

```
update subjectcontacts.dcm_object_variables
set creation_dttm = contact.creation_dttm
from subjectcontacts.dcm_contact_communication contact
where dcm_object_variables.contact_id=contact.contact_id;

update subjectcontacts.dcm_object_values
set creation_dttm = contact.creation_dttm
from subjectcontacts.dcm_contact_communication contact
where dcm_object_values.contact_id=contact.contact_id;

update subjectcontacts.dcm_subject_context
set creation_dttm = contact.creation_dttm
from subjectcontacts.dcm_contact_communication contact
where dcm_subject_context.contact_id=contact.contact_id;
```

Managing Third-Party Database Configurations

About the Subject Contacts and Treatment Definition Databases

By default, the subject contacts and treatment definitions are stored in the PostgreSQL instance that was installed with SAS Viya. You can use Oracle or a different instance of PostgreSQL for these databases. For more information, see "Configure Third-Party Databases" on page 28.

IMPORTANT SAS Intelligent Decisioning does not support third-party configurations in environments that are configured for multi-tenancy.

Configure Third-Party Databases

By default, the subject contacts database and the treatment definitions database are stored in the PostgresSQL instance that was installed with SAS Viya. If you want to use that instance of PostgreSQL for your subject contacts and treatment definitions databases, no additional configuration is required.

Alternatively, you can use Oracle or a different instance of PostgresSQL for either or both of these databases. Complete these steps to configure a third-party database:

1 (Optional) To preserve treatment groups that are stored in the PostgresSQL instance that was installed with SAS Viya, use SAS Environment Manager or the transfer plug-in to the sas-viya CLI to export the treatment groups. Select the **Include dependencies** check box in SAS Environment

Manager, or specify the `--include-dependencies` option in the sas-viya CLI. If you include dependencies in the transfer package, the package includes the eligibility rule sets for the treatments and any version notes that are associated with different versions of the treatments. For more information about exporting and importing content, see *SAS Viya Platform: Content Migration from SAS Viya 4*.

Note: You cannot transfer subject contacts records between databases.

2 Delete any existing third-party configuration settings for the database. For more information, see "Delete Third-Party Database Configurations" on page 33.

3 In SAS Environment Manager, edit the configuration properties for the database. For more information, see "Properties for Third-Party Subject Contacts and Treatment Definition Databases" on page 21.

4 On the primary node of the Kubernetes cluster, export the KUBECONFIG environment variable, and set the value of the Kubernetes namespace (such as `d9997`):

```
export KUBECONFIG=/etc/kubernetes/admin.conf
# List all the available namespaces. Find the namespace of the SAS Viya deployment.
kubectl get namespace
# Set the namespace of the SAS Viya deployment.
namespace=k8s_namespace_value
```

5 Shut down the subject contacts service, the treatment definitions service, or both services, depending on which databases you are reconfiguring:

```
kubectl scale deployment sas-subject-contacts -n $namespace --replicas=0
kubectl scale deployment sas-decisions-definitions -n $namespace --replicas=0
```

6 Start new instances of the services that you shut down in Step 5:

```
kubectl scale deployment sas-subject-contacts -n $namespace --replicas=1
kubectl scale deployment sas-decisions-definitions -n $namespace --replicas=1
```

7 (Optional) Import the treatment groups that you exported in Step 1 into the third-party database.

IMPORTANT After a treatment group is imported into a new database, you must activate the group in order to use the group in a decision. For more information, see "Activate a Treatment Group" in *SAS Intelligent Decisioning: User's Guide* or "Activate Treatment Groups" in *SAS Intelligent Decisioning: Decision Management REST API Examples*.

IMPORTANT The backup and restore processes described in *SAS Viya Platform: Backup and Restore* apply only to services that do not use a third-party configuration. They do not apply to third-party configuration settings or to content stored in a third-party database. You must implement your own backup and restore processes for any services that use a third-party configuration.

Partition the Subject Contacts Database

IMPORTANT Partitioning the subject contacts database requires that PostgreSQL databases are based on PostgreSQL 12. Oracle databases must be based on Oracle 12c Release 2 or higher.

Partitioning and Trimming the Database

The number of records stored in the subject contacts database can grow at a fast pace. However, as time passes, older records typically become less relevant. In order to avoid storing large amounts of stale data, consider periodically trimming the database. How often you trim your database depends on how fast the database grows and how quickly the data becomes stale.

It is easier to remove older, less relevant data if the database is partitioned. With a partitioned database, you can drop the partitions that contain the older data instead of searching for and deleting older records.

IMPORTANT If you are using PostgreSQL and you partition the subject contacts database, you must specify the jvm.java_option_hibernate_extra_table property for the Subject Contact service. For more information, see "Subject Contact Service Properties" on page 18.

The amount of disk space used by the subject contacts database can grow rapidly. Closely monitor the disk space that is used by the database. It is recommended that you partition the subject contacts database and periodically drop older partitions. If the number of partitions is very high, you might experience degraded performance, and calls to the subject contacts database might time out.

SAS provides scripts that generate SQL statements that partition the subject contacts database. There is one script for PostgreSQL databases and one script for Oracle databases. You can use the scripts to generate the SQL statements, and then use a database client application to execute the generated SQL. For more information, see "About the Partitioning Scripts" on page 30.

Note: You must be familiar with how partitioning works for your database. Refer to the documentation for your version of PostgreSQL at https://www.postgresql.org/docs/current/ddl-partitioning.html or for your version of Oracle at https://www.oracle.com/database/technologies/partitioning.html.

About the Partitioning Scripts

The partitionScripts ZIP file contains two scripts: generatePostgreSQLPartitions.sh and generateOracleSQLPartitions.sh. This ZIP file is available at https://support.sas.com/downloads/browse.htm?fil=&cat=612. You can download the ZIP file and extract the contents to your local machine. These scripts generate sample SQL statements to partition your subject contacts database based on datetime ranges.

PostgreSQL does not support the partitioning of existing tables, so you must re-create the subject contacts tables as partitioned tables. In the SQL code that is generated by generatePostgreSQLPartitions.sh, IF statements are enclosed by the begin transaction and commit transaction directives. If an error occurs, changes to the database are rolled back by PostgreSQL.

Syntax

To execute the generatePostgreSQLPartitions script, enter the following command:

```
generatePostgreSQLPartitions.sh dbschema
     "first_partition_start_timestamp"
     partition_granularity
     partition_size
     partition_number
     > output_file_name.sql
```

To execute the generateOraclePartitions script, enter the following command:

```
generateOraclePartitions.sh dbschema
     "first_partition_start_date"
     partition_granularity
     partition_size
     > output_file_name.sql
```

Arguments

dbschema
specify **subjectcontacts**.

first_partition_start_date
specifies the start date of the first partition. Enclose the date in single quotation marks. Specify the date in the following format:

```
YYYY-MM-DD
```

Applies to	generateOraclePartitions script only
Example	'2020–06–02'

first_partition_start_timestamp
specifies the starting timestamp of the first partition. Enclose the timestamp in single quotation marks. Specify the timestamp in the following format:

```
YYYY-MM-DD hh:mm:ss [AM|PM]
```

Applies to	generatePostgreSQLPartitions script only
Example	'2020-04-15 12:00:00 AM'

partition_granularity
specifies the granularity of the partitions. Specify **hour**, **day**, **month**, or **year**.

partition_size
specifies the number of hours, days, months, or years that you want to include in each partition. For example, if you specify **month** for the partition granularity and 3 for the partition size, then each partition includes the data for three months. Specify a whole number greater than zero.

partition_number
specifies how many partitions you want to create. Specify a whole number greater than zero.

Applies to generatePostgreSQLPartitions script only

***output_file_name*.sql**
specifies the name of the file where you want to write the SQL statements.

Examples

The following command writes SQL statements that partition a PostgreSQL subject contacts database. It writes the statements to the file **`/tmp/partition.sql`**. These statements partition the database into five partitions. The first partitions starts on June 1st, 2020, and each partition contains the data for a single three-month period (quarter). You can execute the generated SQL statements in **`/tmp/partition.sql`** by using a database client program such as psql.

```
generatePostgreSQLPartitions.sh subjectcontacts "2020-6-01 12:00:00 AM" month 3 5
     > /tmp/partition.sql
```

The following command writes SQL statements that partition an Oracle subject contacts database. It writes the statements to the file **`partitionSubjectContacts.sql`** in the directory where the script is run. The first partitions starts on July 1st, 2020, and each partition contains data for 15 days. You can execute the generated SQL statements in **`partitionSubjectContacts.sql`** by using a database client that is compatible with your database.

```
generateOraclePartitions.sh subjectcontacts "2020-07-01" day 15 > partitionSubjectContacts.sql
```

Revert to the Default PostgreSQL Database

By default, the subject contacts and treatment definitions databases are stored in the PostgresSQL instance that was installed with SAS Viya. If you have previously chosen to use Oracle or a different instance of PostgresSQL for either or both of these databases, you can reconfigure these databases to use the default instance of PostgreSQL.

1 On the primary node of the Kubernetes cluster, export the KUBECONFIG environment variable, and set the value of the Kubernetes namespace (such as **`d9997`**):

```
export KUBECONFIG=/etc/kubernetes/admin.conf
# List all the available namespaces. Find the namespace of the SAS Viya deployment.
kubectl get namespace
# Set the namespace of the SAS Viya deployment.
namespace=k8s_namespace_value
```

2 Run the following commands to print the PostgreSQL user name and password to the terminal:

```
kubectl -n $namespace exec -i  $(kubectl -n $namespace get pods | grep  sas-consul-
server |grep Running|cut -d " " -f 1) -c sas-consul-server -- bash  -s <<< "export
CONSUL_HTTP_ADDR=https://localhost:8500; export SSL_CERT_FILE=/opt/sas/viya/
config/etc/SASSecurityCertificateFramework/cacerts/trustedcerts.pem; export
CONSUL_TOKEN=\$(cat /opt/sas/viya/config/etc/SASSecurityCertificateFramework/tokens/
consul/default/client.token); /opt/sas/viya/home/bin/sas-bootstrap-config kv read --
recurse config/application/sas-crunchy-data-postgres"
```

3 On the primary node of the Kubernetes cluster, obtain the cluster ID:

```
kubectl -n $namespace get service sas-crunchy-data-postgres -o wide
```

Specify this cluster ID as the host ID in the JDBC URI when you define the spring.datasource.uri property in Step 4.

4 In SAS Environment Manager, edit the configuration properties for the database. For more information, see "Properties for Third-Party Subject Contacts and Treatment Definition Databases" on page 21.

5 Shut down the subject contacts service, the treatment definitions service, or both services, depending on which databases you are reconfiguring:

```
kubectl scale deployment sas-subject-contacts -n $namespace --replicas=0
kubectl scale deployment sas-decisions-definitions -n $namespace --replicas=0
```

6 Start new instances of the services that you shut down in Step 5:

```
kubectl scale deployment sas-subject-contacts -n $namespace --replicas=1
kubectl scale deployment sas-decisions-definitions -n $namespace --replicas=1
```

For more information, see "SAS Infrastructure Data Server" in *SAS Viya: Infrastructure Servers* and Crunchy PostgreSQL Operator.

Delete Third-Party Database Configurations

If a configuration instance is not specified correctly, and the associated services do not start, remove all the third-party database configurations.

1 Determine the configuration instance IDs for your third-party database.

You might have recorded the configuration instance IDs when you completed Step 4f in "Third-Party Database Properties" on page 21.

If you did not record the IDs when you configured the database, you can retrieve the instance IDs by completing these steps:

a On the primary node of the Kubernetes cluster, export the KUBECONFIG environment variable, and set the value of the Kubernetes namespace (such as **d9997**):

```
export KUBECONFIG=/etc/kubernetes/admin.conf
# List all the available namespaces. Find the namespace of the SAS Viya deployment.
kubectl get namespace
# Set the namespace of the SAS Viya deployment.
namespace=k8s_namespace_value
```

b Run the following commands to export the configuration information to a file such as **/tmp/consul.txt**:

```
kubectl -n $namespace exec -i  $(kubectl -n $namespace get pods | grep  sas-consul-
server |grep Running|cut -d " " -f 1) -c sas-consul-server -- bash  -s <<< "export
CONSUL_HTTP_ADDR=https://localhost:8500; export SSL_CERT_FILE=/opt/sas/viya/
config/etc/SASSecurityCertificateFramework/cacerts/trustedcerts.pem; export
CONSUL_TOKEN=\$(cat /opt/sas/viya/config/etc/SASSecurityCertificateFramework/
tokens/consul/default/client.token); /opt/sas/viya/home/bin/sas-bootstrap-config
kv read --recurse config" > /tmp/consul.txt
```

c Delete the static keys from Consul. Open a UNIX shell on the server where Consul is running, and enter the appropriate commands for the database whose configuration you are removing.

To delete the keys for the subject contacts database, enter the following commands:

```
kubectl -n $namespace exec -i  $(kubectl -n $namespace get pods | grep  sas-consul-
server | grep Running|cut -d " " -f 1) -c sas-consul-server -- bash  -s <<<
"export CONSUL_HTTP_ADDR=https://localhost:8501
export SSL_CERT_FILE=/opt/sas/viya/config/etc/SASSecurityCertificateFramework/
cacerts/trustedcerts.pem
export CONSUL_TOKEN=$(sudo cat /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/tokens/consul/default/client.token)
/opt/sas/viya/home/bin/sas-bootstrap-config kv delete config/subjectContacts/
application/schema
/opt/sas/viya/home/bin/sas-bootstrap-config kv delete config/subjectContacts/
sas.datasource.initializer/enabled
/opt/sas/viya/home/bin/sas-bootstrap-config kv delete config/subjectContacts/
sas.verify.resource/database
/opt/sas/viya/home/bin/sas-bootstrap-config kv delete config/subjectContacts/
spring.datasource/url
/opt/sas/viya/home/bin/sas-bootstrap-config kv delete config/subjectContacts/
spring.datasource/driver-class-name
/opt/sas/viya/home/bin/sas-bootstrap-config kv delete config/subjectContacts/
spring.datasource/password
/opt/sas/viya/home/bin/sas-bootstrap-config kv delete config/subjectContacts/
spring.datasource/username
/opt/sas/viya/home/bin/sas-bootstrap-config kv delete config/subjectContacts/
spring.datasource/platform"
```

To delete the keys for the treatment definitions database, enter the following commands:

```
kubectl -n $namespace exec -i  $(kubectl -n $namespace get pods | grep  sas-consul-
server | grep Running|cut -d " " -f 1) -c sas-consul-server -- bash  -s <<<
"export CONSUL_HTTP_ADDR=https://localhost:8501
export SSL_CERT_FILE=/opt/sas/viya/config/etc/SASSecurityCertificateFramework/
cacerts/trustedcerts.pem
export CONSUL_TOKEN=$(sudo cat /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/tokens/consul/default/client.token)
/opt/sas/viya/home/bin/sas-bootstrap-config kv delete config/treatmentDefinitions/
application/schema
/opt/sas/viya/home/bin/sas-bootstrap-config kv delete config/treatmentDefinitions/
sas.datasource.initializer/enabled
/opt/sas/viya/home/bin/sas-bootstrap-config kv delete config/treatmentDefinitions/
sas.verify.resource/database
/opt/sas/viya/home/bin/sas-bootstrap-config kv delete config/treatmentDefinitions/
spring.datasource/url
/opt/sas/viya/home/bin/sas-bootstrap-config kv delete config/treatmentDefinitions/
spring.datasource/driver-class-name
/opt/sas/viya/home/bin/sas-bootstrap-config kv delete config/treatmentDefinitions/
spring.datasource/password
/opt/sas/viya/home/bin/sas-bootstrap-config kv delete config/treatmentDefinitions/
spring.datasource/username
/opt/sas/viya/home/bin/sas-bootstrap-config kv delete config/treatmentDefinitions/
spring.datasource/platform"
```

d Open the **`/tmp/consul.txt`** file, and search for configuration instances for each of the following media types:

```
application/vnd.sas.configuration.config.application
application/vnd.sas.configuration.config.sas.datasource.initializer
application/vnd.sas.configuration.config.sas.verify.resource
application/vnd.sas.configuration.config.spring.datasource
```

TIP If your subject contacts and treatment definition databases were configured together, then these four configuration instances apply to both databases. If your databases were configured separately, then you will find four configuration instances for each database.

e For each configuration instance, record the instance ID. In the following example, the ID for the sas.datasource.initializer configuration instance is 0b16f426-aaa2-42a8-a0bc-047c69f93cab.

```
configurationservice/configurations/0b16f426-aaa2-42a8-a0bc-047c69f93cab={
  "version" : 1,
  "id" : "0b16f426-aaa2-42a8-a0bc-047c69f93cab",
  "metadata" : {
    "isDefault" : false,
    "services" : [ "subjectContacts" ],
    "mediaType" : "application/vnd.sas.configuration.config.sas.datasource.initializer+json;
                   version=1",
    "createdBy" : "edmdev",
    "modifiedBy" : "edmdev",
    "creationTimeStamp" : 1605463464064,
    "modifiedTimeStamp" : 1605463464064
  },
  "enabled" : false
}
```

IMPORTANT Verify that the configuration instance is for the correct media type before you proceed.

2 Use REST API calls to delete the configuration instances associated with each ID. For example, you can use the following CURL command to delete a configuration instance. Substitute the appropriate values for your site for the token, host, port, and ID.

```
curl  -header "Authorization: bearer token"  -X DELETE http://host:port/configuration/
configurations/ID
```

3 (Optional) Repeat Step 1b and Step 1d to verify that the configuration instances have been deleted.

4 Restart the services (sas-subject-contacts, sas-decisions-definitions, or both) for which you deleted the database configuration instances. For more information, see “Managing a Specific Server or Service” in *SAS Viya Platform: General Management of Servers and Services*.

5 Delete the temporary file that was created in Step 1b. This file contains sensitive information.

```
rm /tmp/consul.txt
```

Enable the Check-Out and Commit Feature

To enable the check-out and commit feature for a folder:

1 Specify the folder path in the checkout.checkoutEnabledFolderPaths configuration option for the appropriate service. Set the option for the services that are associated with the object types that you want to save in the folder.

Table 2 Check-Out and Commit Configuration Options

Object Types	Configuration Properties
Rule sets	"sas.businessrules.checkout"
Decisions Code files	"sas.decisions.checkout"
Lookup tables	"sas.referencedata.checkout"
Treatments Treatment groups	"sas.treatmentdefinitions.checkout"

2 (Optional) Set the permissions for the folders that you add to the checkout.checkoutEnabledFolderPaths configuration properties if you want to force users to check out objects in those folders before they are modified. For instructions, see "Set Permissions for Check-Out Folders" on page 59.

Enabling Performance Logging

About Performance Logging

When performance logging is enabled, the Micro Analytic Score service records the total execution time for decisions. The service also records individual times for each node in the decisions except for branch nodes and condition nodes. Times are also logged for nodes that are inside subdecisions. The service writes these times to the SAS Micro Analytic Service log file.

Log entries can include the following information:

package name
: the name of the generated DS2 package.

node name
: the name of node that is displayed in the SAS Intelligent Decisioning user interface.

  The log entry for the entire decision does not include the node name.

node ID
: a unique ID for each node. When multiple decisions are running simultaneously, the logging for both decisions might be interleaved in the SAS Micro Analytic Service log file. If those decisions both contain a node with the same name, you can use the node ID to identify the node.

  The log entry for the entire decision does not include the node name.

total duration
: the elapsed time, in seconds, that it took the node or package to execute. If the number is very small, it is displayed in scientific notation such as 1.9073486328125E-6.

For example, the following log entry shows the execution time for the node named cellPhone_demo in the package cellPhone_demo_0:

```
2019-07-03T15:46:56,659 [00000008] DEBUG App.tk.SID.Perf -
Package Name: cellPhone_demo_0, Node Name: cellPhone_demo,
Node ID: cbe4d5ea-05fe-442a-a2d2-26d5d2d754ea, TOTAL DURATION:0.04584789276123
```

The following log entry shows the execution time for the node named AggregatePromoPredicator in a subdecision, for which the time is very small:

```
2019-07-03T15:46:56,615 [00000008] DEBUG App.tk.SID.Perf -
Package Name: c_VY76XRZ7GFEP3DRTS3BVA6SPLM, Node Name: AggregatePromoPredicator,
Node ID: 04200f5f-2903-44aa-90a2-156890a7fc18, TOTAL DURATION:1.9073486328125E-6
```

The following entry shows the total execution time for the cellPhone_demo_0 package:

```
2019-07-03T15:47:01,264 [00000008] DEBUG App.tk.SID.Perf -
Package Name: cellPhone_demo_0, TOTAL DURATION:1.5830275082711
```

Enable Performance Logging

1 Create the App.tk.SID.Perf logger. This logger captures the execution times for the decision nodes. For instructions, see “Create the App.tk.SID.Perf Logger” below.For more information about logging, see “SAS Micro Analytic Service Logging” in *SAS Micro Analytic Service: Programming and Administration Guide*.

2 Verify that the includeLoggingInGeneratedCode configuration property is turned on. When this property is on, additional LOG statements are added to the generated DS2 code. These statements write performance information to the SAS Micro Analytic Service log. For more information, see “Decisions Service Properties” on page 8.

Create the App.tk.SID.Perf Logger

1 Click ☰ and select **Manage Environment** to switch to SAS Environment Manager.

2 Click 🔧.

3 Select **All services** in the **View** menu, and then select **Micro Analytic Score service**.

4 Click **New Configuration**. The Select Definition window appears.

5 Select **logging.level**. The New logging.level Configuration window appears.

6 Select **DEBUG** for the logging level.

7 Enter **`App.tk.SID.Perf`** for the logger name, and click **Save**.

8 Restart the SAS Micro Analytic Score service. Use the Kubernetes commands that are described in "Managing a Specific Server or Service" in *SAS Viya Platform: General Management of Servers and Services*.

Enabling Variable Assignment Logging

About Variable Assignment Logging

When variable assignment logging is enabled, users can select the **Enable variable assignment logging** option for scenario tests, which are run in CAS. This option adds the following statement to the code that is generated when the test is run:

```
DS2_OPTIONS TRACEVARIABLES;
```

Variable assignment logging writes detailed information to a CAS log each time the value of a variable changes. This information enables the user to track how a variable's value changes as a decision executes. For more information, see "Using Variable Assignment Logging" in *SAS Intelligent Decisioning: User's Guide*.

About The sessionlogconfig Configuration Instance

The contents of the `sas.cas.instance.config: sessionlogconfig` configuration instance for the cas-shared-default service has the following structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<logging:configuration xmlns:logging="http://www.sas.com/xml/logging/1.0/">

appender and logger definitions

<!--Root logger -->
<root>
   root logger definition
</root>

</logging:configuration>
```

By default, the contents defines an example configuration for an appender named ProcessFile and the root logger. The example appender, if enabled, causes CAS sessions to write debug log files to the /tmp directory when a new session starts. Setting the CAS_SESSION_LOGGING environment variable to True enables this appender.

You must also set this variable to True in order to enable the appender for variable assignment logging. If do not want to enable the ProcessFile appender, delete or comment out the example appender definition and modify the root logger definition. See Step 2 of “Enable Variable Assignment Logging”.

Enable Variable Assignment Logging

IMPORTANT All the CAS configuration instances are disabled by default, which means that processing is not enabled for any modifications to the configuration instances in SAS Environment Manager. In order to modify the cas-shared-default configuration instances, a Kubernetes administrator must set SAS_ALLOW_ADMIN_SCRIPTS to True in `sas-shared-config configMap`. For instructions, see the file **`$deploy/sas-bases/overlays/sas-programming-environment/README.md`**. For more information, see “Edit Server Configuration Instances” in *SAS Environment Manager: User’s Guide*.

Also, you must restart the CAS server after you make changes to the cas-shared-default configuration settings.

To enable variable assignment logging, make the following configuration changes:

1 In SAS Environment Manager, edit the `sas.cas.instance.config: sessionlogconfig` configuration instance for the cas-shared-default service.

2 (Optional) Delete or comment out the ProcessFile example appender definition, and delete or comment out the following line from the root logger definition:

```
<appender-ref ref="ProcessFile"/>
```

3 Add the App.TableServices.DS2.Runtime.TraceVariables logger. Copy and paste the following appender and logger definitions above the root logger definition:

```
<!-- Arke Session Logging -->
 <appender name="CASArkeAppender" class="ArkeAppender">
  <param name="Client" value="sas-cas-server-controller"/>
  <param name="Address" value="sas.edm.tracing.decisions"/>
  <param name="Type" value="Queue"/>
  <param name="Subject" value="variable.assignment"/>
  <param name="Durable" value="TRUE"/>
  <layout type="Json">
   <param name="Individual" value="true"/>
   <param name="version#" value="%S{eventModel.version}"/>
   <param name="type" value="%S{eventModel.type}"/>
   <param name="payloadType" value="%S{eventModel.payloadtype}"/>
   <param name="timeStamp" value="%d{LEMZone}"/>
   <param name="headers.__tenant" value="%S{eventModel.headers.__tenant}"/>
   <param name="headers.sas-content-type" value="%S{eventModel.headers.sas-content-
type}"/>
   <param name="headers.sas-deployment-id" value="%S{OSENV.SAS_DEPLOYMENT_ID|viya}"/>
   <param name="headers.sas-event-source" value="%S{eventModel.headers.sas-event-
source}"/>
   <param name="headers.sas-published-timestamp" value="%d{LEMZone}"/>
```

```
      <param name="payload.version#" value="%S{eventModel.payload.version}"/>
      <param name="payload.level" value="%S{eventModel.payload.level}"/>
      <param name="payload.source" value="%S{OSENV.SAS_PAYLOAD_SOURCE|sas}"/>
      <param name="payload.messageID" value="%S{eventModel.payload.messageId}"/>
      <param name="payload.message" value="%m"/>
      <param name="payload.timeStamp" value="%d{LEMZone}"/>
      <param name="payload.parameters{}" value="%S{eventModel.payload.parameters}"/>
      <param name="payload.properties.thread" value="%t"/>
      <param name="payload.properties._lineNumber_" value="%L"/>
      <param name="payload.properties._sourceFile_" value="%F"/>
      <param name="payload.properties.hostname" value="%S{hostname}"/>
    </layout>
  </appender>

  <logger name="App.TableServices.DS2.Runtime.TraceVariables">
   <appender-ref ref="CASArkeAppender"/>
   <level value="Trace"/>
  </logger>
```

For more specific instructions, see "Manage CAS Server Logging" in *SAS Viya: Logging*.

TIP You can specify `debug` instead of `trace`.

For more information about the logger, see "App.TableServices.DS2.Runtime.TraceVariables" in *SAS DS2 Programmer's Guide* and "Example: Logging Trace Variables" in *SAS DS2 Programmer's Guide*.

4 Edit the contents of the `sas.cas.instance.config: settings` configuration instance for the cas-shared-default service in SAS Environment Manager. To enable CAS session logging, uncomment the following line:

```
export CAS_SESSION_LOGGING="true"
```

Note: If you did not delete the example appender definition in Step 2, then the ProcessFile example appender is also enabled when you export this variable.

5 To include rule-fired information and path-tracking information in the log files, turn on the sas.decisions.nodetraces.includeRuleFiredPathTrackInfoInVariableAssignmentLogging configuration property. If this option is turned off, users cannot run rule-fired analyses or path-tracking analyses for scenario tests. For more information, see "sas.decisions.nodetraces" on page 13.

TIP If the log files that are generated by either the **Enable variable change path logging** option or the **Enable variable assignment logging** option are very large, you might need to increase the JVM heap size for the Subject Contact and Files services. Because of the storage required by these options, do not select both of these options in the same test. For more information, see "Subject Contact Service Properties" on page 18.

For more information about SAS Intelligent Decisioning configuration options, see "Decisions Service Properties" on page 8. For information about single-server systems and distributed systems, see *SAS Cloud Analytic Services: Fundamentals*

Enable the Asset Approval Workflow

SAS Intelligent Decisioning provides a predefined approval workflow for decisions. This workflow is named SID Asset Approval. When the workflow is enabled, a new instance of the workflow is started each time a new decision or a new version of an existing decision is created in SAS Intelligent Decisioning. The **Move to status** button appears in the interface when a user opens a decision that is in a workflow. As the users that are responsible for developing, testing, reviewing, approving, deploying the decision version each finish their tasks, they set the status of the workflow to the appropriate value, such as Developing, Approved, or Deployment-ready. For more information about the workflow, see "Using SAS Workflow with SAS Intelligent Decisioning" in *SAS Intelligent Decisioning: User's Guide*.

To enable this workflow for decisions in SAS Intelligent Decisioning:

1 Turn on the sas.decisions.workflow.enabled configuration property.

2 (Optional) Turn on the sas.decisions.workflow.authorMayApprove configuration property. This property specifies whether a user who authors a decision can also approve the decision.

3 (Recommended) Turn on the sas.businessrules.publish.hideRuleSetPublish configuration property. This property specifies whether the **Publish** button is hidden for rule sets. It is recommended that you turn on this property so that all published content is developed in the workflow.

4 Define the custom user groups for the workflow, and assign the appropriate users to each group. For more information, see "Define Asset Approval Workflow User Groups" on page 57.

5 Grant the appropriate users access to the workflow history. For more information, see "Granting Access to the History of Workflow Status Changes" on page 58.

6 (Optional) Configure the length of time that workflow history items are retained. For more information, see "Configure History Event Properties" in *SAS Workflow Manager: Administrator's Guide*.

Migrating Content

About Migration

You can move from previous releases or versions of SAS to the latest version. You can move from SAS 9.4 to SAS Viya, from SAS Viya 3.5 to SAS Viya 4, or from SAS Viya 4 to a different SAS Viya 4 deployment. You can migrate content, or you can migrate a full system. For information about the SAS Viya migration processes, see *System Migration and Content Migration*.

For information about migrating SAS Intelligent Decisioning content, see the following topics:

- “Transfer Business Rules Content from SAS 9.4 to SAS Viya”
- “Transfer Treatments from SAS 9.4 to SAS Viya”
- “Transfer Content from SAS Viya 3.5 Environments”
- “Transfer Content between SAS Viya 4 Environments”
- “Transfer Content That Was Published to a CAS Destination”

Transfer Business Rules Content from SAS 9.4 to SAS Viya

Note: The transfer process does not support transferring content to SAS Viya from SAS Business Rules Manager 2.2, or earlier or from SAS Decision Manager 2.2.

To transfer content from SAS Business Rules Manager on SAS 9.4 or SAS Decision Manager on SAS 9.4 to SAS Intelligent Decisioning on SAS Viya, use the dcmtransfer plug-in to the sas-viya command-line interface.To transfer content from SAS Business Rules Manager on SAS 9.4 or SAS Decision Manager on SAS 9.4 to SAS Intelligent Decisioning on SAS Viya, use the dcmtransfer plug-in to the sas-admin command-line interface. For more information, see the following topics:

- “About the SAS Intelligent Decisioning CLIs” in *SAS Intelligent Decisioning: Command-Line Interfaces*
- “Command-Line Interface: Preliminary Instructions” in *SAS Viya Platform: Using the Command-Line Interface*
- “dcmtransfer Plug-In” in *SAS Intelligent Decisioning: Command-Line Interfaces*

IMPORTANT SAS Intelligent Decisioning on SAS Viya does not support rule flows. Simple rule flows are transferred as decisions. You cannot transfer complex rule flows (rule flows that use BY-group processing) from SAS 9.4 to SAS Viya. You must recreate these rule flows as rule sets in the target environment.

Note: For rule expressions that use macros, the macros are transferred, but you must replace the macros in the rule sets after the rule sets are imported to SAS Viya.

IMPORTANT In order to use a lookup table that has been transferred to a new environment, you must do one of the following:

- activate the table in the new environment. For instructions for activating a lookup table by using the user interface, see “Activating a Lookup Table” in *SAS Intelligent Decisioning: User’s Guide*.
- transfer the formats data on SAS Cloud Analytic Services (CAS) and the SAS Micro Analytic Service module for the SAS Micro Analytic Service destination. The module name is shown on the **Properties** tab for the lookup table.

Test definitions and test results for rule sets, models, and decisions are not transferred automatically when rule sets, models, and decisions are transferred. To transfer test definitions and test results that are in a folder, you can transfer the folder. To transfer test definitions and test results that were not saved in a folder, you must list the URI for each definition and results table in the transfer request. (Beginning with SAS Intelligent Decisioning 5.3, you can save test definitions and test results in a folder.)

Output tables in CAS must be transferred manually. However, it is recommended that you re-create and rerun the test in the target environment instead of transferring the old output tables and test information.

Transfer Treatments from SAS 9.4 to SAS Viya

You can transfer treatments from SAS Real-Time Decision Manager on SAS 9.4 to SAS Intelligent Decisioning on SAS Viya. This process uses the sasmaextract integration utility that is distributed with SAS Customer Intelligence.

The transfer process transfers custom detail names, values, and whether the custom details are dynamic or static. Custom details in SAS Real-Time Decision Manager become treatment attributes in SAS Intelligent Decisioning. The detail label becomes the attribute name. Static details become fixed attributes.

Some custom detail properties such as whether a detail is required and the number of decimal places for a detail value are not applicable in SAS Intelligent Decisioning and are not transferred.

Some custom detail properties are transferred but handled differently in SAS Intelligent Decisioning.

- SAS Intelligent Decisioning does not support the data types time, week, month, quarter, or year. Custom details of these types become character attributes in SAS Intelligent Decisioning.
- The Decision Management API enables you to create attributes that define a range of values, but the SAS Intelligent Decisioning user interface displays range values as `[object Object]`.
- If a custom detail has a drop-down list of numeric values, the detail is transferred as an attribute that has a list of values. However, the list of attribute values contains only the default value from the drop-down list in SAS Real-Time Decision Manager.
- If a custom detail has a drop-down list of date values, the detail is not transferred as a list. It is transferred as a date attribute with the default value from the drop-down list in SAS Real-Time Decision Manager.

To transfer treatments from a SAS 9.4 environment to a SAS Viya environment, complete these steps:

1 Download and unpack or unzip the SAS Intelligent Decisioning CLIs for your platform. The download file is available at https://support.sas.com/downloads/browse.htm?fil=&cat=612. For more information about the download file, see "About the SAS Intelligent Decisioning CLIs" in *SAS Intelligent Decisioning: Command-Line Interfaces* and "Downloading and Installing the CLIs" in *SAS Intelligent Decisioning: Command-Line Interfaces*.

   This download file contains an input stylesheet and an output stylesheet for use with the sasmaextract command: pass_thru.xslt and transform_treatments.xslt.

   > **IMPORTANT** Do not modify these stylesheets.

2 Log on to the machine where SAS Customer Intelligence on SAS 9.4 is running.

3. Change to the directory **C:\Program Files\SASHome\SASMarketingAutomationIntegrationUtilities\*release***.

4. Copy the two stylesheets from the download package into the current directory.

5. Create a file named **extract_request.xml** that contains the `<MAExtractRequest>` extract request. Specify the criteria for the treatments that you want to transfer. For example, to extract all of the treatments in the folder MyTreatments, specify the following request:

   ```
   <?xml version="1.0"?>
   <MAExtractRequest detail="ALL">
       <TreatmentDO>
           <Folder operator="=">
               <Name operator="=">MyTreatments</Name>
           </Folder>
       </TreatmentDO>
   </MAExtractRequest>
   ```

   You can also extract treatments based on treatment names, IDs, descriptions, and other criteria. For more information about information on specifying the criteria in this file, see "Treatments" in *SAS Customer Intelligence Integration Utilities: User's Guide*.

6. Execute the sasmaextract command. Specify **pass_thru.xslt** as the input style sheet and **transform_treatments.xslt** as the output style sheet.

   ```
   sasmaextract domain\user-ID password authorization-domain
        business-context-of-the-treatments
        extract_request.xml
        output-filename
        pass_thru.xslt
        transform_treatments.xslt
   ```

   For more information about the sasmaextract command, see "Using the Extract Utility (Sasmaextract)" in *SAS Customer Intelligence Integration Utilities: User's Guide*.

7. Copy the output file to a location where it can be accessed by the sas-rtdmobjectmigration-cli CLI.

8. Log off from the machine where SAS 9.4 is installed.

9. On SAS Viya, run the sas-rtdmobjectmigration-cli CLI to import the treatments. For more information, see "Transfer Objects from SAS 9.4 to SAS Viya" in *SAS Intelligent Decisioning: Command-Line Interfaces*.

Transfer Content from SAS Viya 3.5 Environments

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 on Viya 3.5 to export the content. For information, see *SAS Viya 3.5 Administration: Promotion (Import and Export)*. Notes that are associated with specific versions of the object are not transferred with the object. You can add notes to the source objects in the new environment, or manually transfer the notes. For more information about transferring version notes, see "Transfer Version Comments" in *SAS Intelligent Decisioning: Command-Line Interfaces*.

Transfer Content between SAS Viya 4 Environments

To transfer SAS Intelligent Decisioning content between Viya 4 environments, use SAS Environment Manager or the transfer plug-in to the sas-viya CLI. When you export an object in SAS Environment Manager, you can select the **Include dependencies** check box to include dependent objects in the transfer package that SAS Environment Manager creates. When you use the sas-viya CLI to export an object, you can specify the `--include-dependencies` option to include dependent objects in the transfer package.

The transfer service inspects the object that you are transferring and adds any additional objects that it finds to the package. For example, if you select the check box or specify the option when you export a treatment, the transfer package includes the eligibility rule set for the treatment and any version notes that are associated with different versions of the treatment. If you export a decision, the transfer package includes all of the information for any rule sets, treatment groups, lookup tables, subdecisions, and other objects (except models) that are used in the decision. It also includes objects that are used in subdecisions.

Some object types are handled differently:

- Global variables are automatically transferred when you transfer a rule set or a decision regardless of whether you select the option for including dependencies. However, if a global variable already exists in the target environment where you import the content, SAS Intelligent Decisioning does not overwrite the existing global variable with the imported version.
- Version notes are included in the transfer package, so you do not need to transfer the notes separately. Version notes in the transfer package are displayed as `unnamed` resources when you view the contents of the package in SAS Environment Manager.
- Models are not included in the transfer package for a decision. For information about transferring models, see “Migrating Content between SAS Viya Environments” in *SAS Model Manager: Administrator’s Guide*.
- Custom functions are not included directly in a decision, so they are not considered dependent objects and are not included in the transfer package.
- Version tags are not included in the transfer package for any objects. It is recommended that you assign new tags in the target environment.

For more information about promotion process for Viya 4, see *SAS Viya Platform: Content Migration from SAS Viya 4*.

IMPORTANT After a treatment group is transferred to a new environment, you must activate the group in the new environment in order to use the group in a decision. For more information, see “Activate a Treatment Group” in *SAS Intelligent Decisioning: User’s Guide* or “Activate Treatment Groups” in *SAS Intelligent Decisioning: Decision Management REST API Examples*.

IMPORTANT In order to use a lookup table that has been transferred to a new environment, you must do one of the following:

- activate the table in the new environment. For instructions for activating a lookup table by using the user interface, see "Activating a Lookup Table" in *SAS Intelligent Decisioning: User's Guide*.
- transfer the formats data on SAS Cloud Analytic Services (CAS) and the SAS Micro Analytic Service module for the SAS Micro Analytic Service destination. The module name is shown on the **Properties** tab for the lookup table.

Test definitions and test results for rule sets, models, and decisions are not transferred automatically when rule sets, models, and decisions are transferred. To transfer test definitions and test results that are in a folder, you can transfer the folder. To transfer test definitions and test results that were not saved in a folder, you must list the URI for each definition and results table in the transfer request. (Beginning with SAS Intelligent Decisioning 5.3, you can save test definitions and test results in a folder.)

Output tables in CAS must be transferred manually. However, it is recommended that you re-create and rerun the test in the target environment instead of transferring the old output tables and test information.

Transfer Content That Was Published to a CAS Destination

You can transfer the published module for a rule set, decision, or model by using the transfer plug-in to the sas-viyasas-admin CLI. In the CLI command, identify the module that you want to transfer by specifying its URI in the following format:

```
/modelPublish/destinations/destination_name/models/published_name
```

Only the content that was included inline in the generated module is transferred. For CAS destinations, treatment groups and global variables are always included in the generated module. Lookup tables are included in the generated module only if the sas.businessrules.lookupStaticBinding configuration option was set to On when the content was published. If this option is set to Off when the content was published, you must transfer the lookup format library. When you transfer the library, all of the lookup tables that are activated in the source environment are also transferred. For more information, see the following topics:

- "sas.businessrules.lookupStaticBinding" on page 6
- "sas.referencedata.casformats.formatsLibrary" on page 16
- "sas.referencedata.activation.activateLookupOnImport" on page 16
- "Content Executed by Published Decisions" in *SAS Intelligent Decisioning: User's Guide*

To transfer a published module between SAS Viya environments:

1 Export the published module. For example, to export a module named card_offers1_0 that was published to the destination CASPublic, specify the following CLI command:

```
sas-admin --profile source-profile transfer export
    --resource-uri "/modelPublish/destinations/CASPublic/models/card_offers1_0"
```

```
sas-viya --profile source-profile transfer export
    --resource-uri "/modelPublish/destinations/CASPublic/models/card_offers1_0"
```

If the `export` command is successful, the CLI returns the package ID:

```
The package with the ID 8f32140f-34e1-42dd-8d85-f8129ff5dabb was created.
```

TIP You can export multiple items with one transfer request by specifying the items in a JSON file and specifying the JSON file as input to the CLI command. The JSON file contains two fields named `name` and `items`. For example, to export the modules for three different versions of the card_offers object, the JSON file might contain the following data:

```
{
    "name": "Published Models",
    "items": [
        "/modelPublish/destinations/CASPublic/models/card_offers1_0",
        "/modelPublish/destinations/CASPublic/models/card_offers1_1",
        "/modelPublish/destinations/CASPublic/models/card_offers1_2"
    ]
}
```

Specify the JSON file name with the `--request` option in the CLI command. If this JSON is in a file named `request.json`, specify the following command to export all of the specified modules:

```
sas-admin --profile source-profile transfer export --request @request.json
```

```
sas-viya --profile source-profile transfer export --request @request.json
```

2 Download the JSON package file. Specify the package ID and the JSON file name in the `download` command:

```
sas-admin --profile source-profile transfer download --id "8f32140f-34e1-42dd-8d85-f8129ff5dabb"
    -f card_offers1_0.json
```

```
sas-viya --profile source-profile transfer download --id "8f32140f-34e1-42dd-8d85-f8129ff5dabb"
    -f card_offers1_0.json
```

3 Verify that the target server has a CAS publishing destination with the same name as the source server. To import the package successfully, the target server must have a CAS publishing destination with same name as the source server. For more information, see *SAS Viya: Publishing Destinations*.

4 Use the `upload` command to upload the JSON file to the destination server:

```
sas-admin --profile target-profile transfer upload -f card_offers1_0.json
```

```
sas-viya --profile target-profile transfer upload -f card_offers1_0.json
```

If the `upload` command is successful, the CLI prints a message that includes the ID of the transfer package. For example:

```
{
    "description": "",
    "id": "ef4204f7-8a79-49ae-8d45-a7877f5e1e86",
    "links": [
        ...
    ],
    "name": "Export",
    "type": "TransferPackage",
    "version": 2
}
```

5 Import the contents of the transfer package. Specify the ID of the transfer package that was returned in the previous step:

```
sas-admin --profile target-profile transfer import --id "ef4204f7-8a79-49ae-8d45-a7877f5e1e86"
```

```
sas-viya --profile target-profile transfer import --id "ef4204f7-8a79-49ae-8d45-a7877f5e1e86"
```

6 Verify that the import process completed successfully. Use the `show` command to display the export and import history for the package.

To display the export history:

```
sas-admin --profile source-profile transfer show
    --id "8f32140f-34e1-42dd-8d85-f8129ff5dabb" --history
```

```
sas-viya --profile source-profile transfer show
    --id "8f32140f-34e1-42dd-8d85-f8129ff5dabb" --history
```

To display the import history:

```
sas-admin --profile target-profile transfer show
    --id "ef4204f7-8a79-49ae-8d45-a7877f5e1e86" --history
```

```
sas-viya --profile target-profile transfer show
    --id "ef4204f7-8a79-49ae-8d45-a7877f5e1e86" --history
```

For more information about using the transfer plug-in, see “How To (CLI)” in *SAS Viya Platform: Content Migration from SAS Viya 4*.

You can execute published modules by using the CAS Model Publishing and Scoring action set. For more information, see “Executing Content That Has Been Published to SAS Cloud Analytic Services Destinations” in *SAS Intelligent Decisioning: User’s Guide*.

Managing Permissions

About Permissions

You use SAS Environment Manager to manage identities and authorization for SAS Viya. Information is available in the SAS Viya administration documentation:

- “Authorization in SAS Viya” in *SAS Viya Platform: Orientation to Authorization*
- “Identity Management: Overview” in *SAS Viya: Identity Management*
- *SAS Viya Platform: External Credentials*

You can configure user access based on folders, object types, or specific objects. You can control which categories appear in the user interface by controlling access to root endpoints. To grant full access to an object, a user must have access to all of the service endpoints (object URIs) that are associated with the object. For more information, see “Full Access and Service Endpoints” on page 54.

The default permissions for SAS Intelligent Decisioning are described in “Default Permissions” on page 49.

Default Permissions

Rule sets, decisions, treatments, treatment groups, lookup tables, and custom code files are stored in folders. When these objects are created through the user interface, the default permissions are determined by the folder in which the object is stored.

In SAS Environment Manager, each folder is assigned permissions that apply to the folder itself and permissions that apply to the objects within the folder. The permissions that apply to the objects within the folder are said to be conveyed to those objects within the folder.

Global variables and custom functions are not stored in folders. The default permissions for these objects are controlled by authorization rules.

By default, each member of the Authenticated Users group has permission to do the following:

- create rule sets, lookup tables, treatments, treatment groups, code files, global variables, custom functions, and decisions in their own `My Folder` folder
- in the `Decision Repository` folder, check out and commit objects that they created
- create subfolders within the `Decision Repository` folder
- update and delete any rule set, lookup table, treatment, treatment group, code file, global variable, custom function, or decision that was created in a folder to which the user has conveyed Update and Delete access
- delete a specific version of a rule set, lookup table, treatment, treatment group, global variable, or decision if they have Delete permission for the object
- activate any lookup table, global variable, or treatment group that was created in a folder to which the user has conveyed Update access
- publish any rule set, decision, or model to which the user has Read access
- run a publishing validation test for any rule set, decision, or model that they published
- create a test definition, including scenario test definitions, for any rule set or decision to which the user has Read access
- update or delete the test definition that they created
- run the test and view the test results for any test definition that they created
- run a rule-fired analysis or decision-path tracking analysis and view the results for any test that they created and executed

Default folders such as a user's `My Folder` folder or the `Decision Repository` folder are assigned permissions so that any subfolders that are created within these folders inherit the permissions of the parent folder (unless the parent folder's permission have been customized). For folder's that are created directly under `SAS Content`, you must set permissions in SAS Environment Manager in order to enable users to access these folders. See "Folder-Based Permissions" on page 51 for more information.

Modifying Permissions

You can modify the default permissions in the following ways:

- Modify permissions for a specific folder and the objects that are stored in that folder by using the Edit Authorization window for the folder. For more information, see the following topics:
  - "About Permission Types" on page 50
  - "Best Practices for Setting Permissions" on page 51
  - *SAS Viya: General Authorization Window*
  - "Folder-Based Permissions" on page 51
- Modify the existing rules or create new rules. For more information, see the following topics:
  - "About Permission Types" on page 50
  - "Best Practices for Setting Permissions" on page 51
  - *SAS Viya Platform: General Authorization*
  - "Rules Page" in *SAS Environment Manager: User's Guide*
  - *SAS Viya: General Authorization Window*
  - "Tasks Enabled by Default Rules for Object URIs" on page 54
  - "Rules-Based Permissions" on page 53
- Modify the existing user groups or create new ones. For more information, see the following topics:
  - "Custom Groups" in *SAS Viya: Identity Management*
  - "Manage Custom Groups" in *SAS Environment Manager: User's Guide*
  - "Granting Access to Test Results" on page 56
  - "Define Asset Approval Workflow User Groups" on page 57

About Permission Types

There are two types of permissions that you might need to set for users to work in SAS Intelligent Decisioning: folder-based permissions and rules-based permissions.

Table 3 *Permissions Types for SAS Intelligent Decisioning*

Permissions Type	When to Use
Folder-based permissions	For objects that are stored in folders, it is recommended that you use folder-based permissions to control what tasks users can perform for those objects. The objects that are stored in folders are rule sets, decisions, treatments, treatment groups, lookup tables, and custom code files. Storing test definitions and test results in a folder is optional, but it is highly recommended. Storing test definitions and test results in a folder simplifies the tasks of setting permissions and transferring the files. For more information, see "Folder-Based Permissions" on page 51.
Rules-based permissions	For tasks associated with publishing and for objects that are not stored in folders, you must use rules-based permissions. The objects that are not stored in folders are custom functions and global variables.

Permissions Type	When to Use
	The ability to store tests outside of folders is supported for legacy purposes only. If you choose not to store test definitions and test results in a folder, then you must use rules-based permissions to control access to these files. For more information, see “Rules-Based Permissions” on page 53.

Best Practices for Setting Permissions

To the extent possible, base your permission settings on the folders in which objects are stored. Use folders to organize objects according to which group of users need access to the objects. Customize the permission settings for your folders based on user groups, not on individual users.

If your site needs separate folders that are enabled for the check-out and commit feature, it is recommended that you create these folders in the `/Decision Repository` folder. Subfolders in the repository inherit the default settings for the `/Decision Repository` folder.

TIP Rules for which the setting is **Prohibit** take precedence over all other rules for that URI. For example, if the same user belongs to a prohibited group and to an authorized group, the prohibited assignment takes precedence over the authorized assignment.

When you enter a rule that specifies **Prohibit**, make the URI for that rule as specific as possible.

Folder-Based Permissions

Setting Folder-Based Permissions

You must set permissions in SAS Environment Manager for any folder that is created as a new top-level folder under `SAS Content`. For default folders such as the `My Folder` folder or the `Decision Repository` folder, you must modify the permissions of folders that you create within these two folders if you do not want them to inherit the permissions of the parent folder.

1 Click ☰, and select **Manage Environment** to switch to SAS Environment Manager.

2 Click 🗎 to navigate to the **Content** page.

3 Navigate to the folder for which you want to set permissions.

4 Right-click the folder name, and select **Edit authorization**. The Edit Authorization window appears.

5 For any permission setting that you want to change, click the effective access icon in the column for that permission, and change the setting in the pop-up window. For more information about

setting permissions in this window, see *SAS Viya: General Authorization Window* and “Folder-Based Permission Settings for SAS Intelligent Decisioning Tasks” on page 52.

6 Click **Save** to save your changes and close the Edit Authorization window.

Folder-Based Permission Settings for SAS Intelligent Decisioning Tasks

The following table lists the folder permissions that are required to work with the objects that are stored in a folder. The permissions listed in this table assume that the permissions described in “Default Permissions” have not been customized.

Table 4   Folder-Based Permission Settings for Tasks in SAS Intelligent Decisioning

Task	Folder Permissions
Make objects in a folder visible to users	Read
Save an object in a folder	Update
Create a new object	Add
Create a new version of an object	Update
Duplicate an object	Add, Update
Rename an object	Update
Copy a version of an object	Update
Activate a lookup table or a treatment group	Update
Delete an object	Delete
Move an object from folder A to folder B	Remove (folder A) Add (folder B)
Validate a decision	Read
Create a test definition	Add
Run a test	Add
Rerun a previously run test	Add, Delete
Check out and commit objects	Read, Add, Update

Rules-Based Permissions

Setting Rules-Based Permissions

1. Click ≡, and select **Manage Environment** to switch to SAS Environment Manager.
2. Click to navigate to the **Rules** page.
3. Click to create a new rule, or select a rule, and click to modify an existing rule.
4. Edit the rule's settings and click **Save**.

   You can click ⓘ in the window for information about specific fields. For the object URIs for SAS Intelligent Decisioning , see "Tasks Enabled by Default Rules for Object URIs" on page 54. For additional information about rule settings, see *SAS Viya Platform: General Authorization*.

Rules-Based Permission Settings for Publishing Tasks

The following table lists the permissions that are required to customize the ability to publish objects to specific destinations.

Table 5 Rules-Based Permission Settings for Publishing Tasks

Task	URI	Rule Type	Permissions
View available publishing destinations	/modelPublish/destinations/**	Grant	Read
Publish objects to any destination	/modelPublish/destinations/**	Grant	Read
	/modelPublish/models/**	Grant	Create
Publish objects to only one specific destination	/modelPublish/destinations/*destination-name*	Grant	Read
	/modelPublish/models/**	Grant	Create
Prohibit users from viewing a specific destination	/modelPublish/destinations/*destination-name*	Prohibit	Read

TIP Rules for which the setting is **Prohibit** take precedence of all other rules for that URI. When you enter a rule that specifies **Prohibit**, make the URI for that rule as specific as possible.

You can also specify rules for other SAS Intelligent Decisioning URIs. The SAS Intelligent Decisioning URIs are listed in “Tasks Enabled by Default Rules for Object URIs” on page 54

Full Access and Service Endpoints

In order to have full access to an object, a user must have access to the folder that contains the object, to the specific object, to any additional objects that are referenced by the object, and to the service endpoints for all object types. For example, in order to have full access to a specific decision, the user must have access to the following:

- the folder that contains the decision.
- the folder that contains the test definition and test results.
- the decision, plus any rule sets, lookup tables, models, treatment groups, code files, global variables, custom functions, and nested decisions that are included in the decision.
- the service endpoints for the object types for the folder, the decision, and all of the objects that are included in the decision. If the decision contains a model, the endpoints for the model repository and the model project (if the model is in a project) must be included.
- the service endpoints for the object types that are needed to create and run a decision test: /scoreDefinitions/definitions and /scoreExecution/executions. Alternatively, if you are using the SASScoreUsers group, the user can be a member of the SASScoreUsers group. See “Granting Access to Test Results” on page 56 for more information.
- the service endpoints that are needed to publish the decision: /modelPublish/destination and /modelPublish/destination/{*destination*}.

Service endpoints for specific object types are represented by the object URIs.These object URIs are shown in “Tasks Enabled by Default Rules for Object URIs” on page 54. You grant permissions for object URIs by creating or modifying rules in SAS Environment Manager. For more information, see “Setting Rules-Based Permissions” on page 53 and “Rules Page” in *SAS Environment Manager: User’s Guide*.

Note: If a user has access to a decision but does not have access to an object that is referenced in the decision, SAS Intelligent Decisioning displays ⊗ next to the object name.

Tasks Enabled by Default Rules for Object URIs

Permission rules for object URIs are defined in SAS Environment Manager. By default, general rules are defined for all object URIs in SAS Intelligent Decisioning. Table 6 describes the tasks that are enabled by these default rules.

Table 6 *Tasks Enabled by Default Rules for Object URIs in SAS Intelligent Decisioning*

Object URI	Tasks
/businessRules	View the Business Rules category.

Object URI	Tasks
/businessRules/ruleSets	Create, read, update, and delete rule sets. Create and delete versions of rule sets. Generate SAS code for rule sets.
/businessRules/rules	Import and export rule sets.
/decisions	View the Decisions category.
/decisions/codeFiles	Create, read, update, and delete custom code files.
/decisions/commons/validations/codeFiles	Validate code file content. Code file content is validated when you save the code file.
/decisions/flows	Create, read, update, and delete decisions. Create and delete versions of decisions. Generate SAS code for decisions.
/folders/folders	Create and read folders.
/folders/folders/*folder-ID*	Create, read, update, and delete the specified folder.
/businessRules/functionCategories	Create, read, update, and delete custom function categories.
/businessRules/functionCategories/*category-ID*/functions	Create custom functions in the specified category.
/businessRules/functions/*function-ID*	Read, update, and delete the specified custom function.
/modelPublish/models/**	Publish rule sets, models, and decisions to a publishing destination.
/modelPublish/destinations	Define new publish destinations.
/modelPublish/destinations/{*destinationName*}	Update or delete an existing destination. Publish content to the specified destination, and read published content in the specified destination.
/referenceData	View the Lookup Tables and Global Variables categories. Create, read, update, and delete lookup tables and global variables. Import and export lookup tables. Create, delete, and activate versions of lookup tables and global variables.
/referenceData/domains	Create, read, update, and delete lookup tables. Create, delete, and activate versions of lookup tables.
/referenceData/domainEntries	Import and export lookup tables.
/referenceData/globalVariables	Create, read, update, and delete global variables. Create, delete, and activate versions of global variables.

Object URI	Tasks
/scoreDefinitions/definitions/**	Create, read, update, and delete rule set tests, decision tests, model tests, and publishing validation tests in the user interface and in the score definition service.
/scoreExecution/executions/**	Run rule set tests, decision tests, model tests, and publishing validation tests in the user interface and the score execution service. Run rule-fired analyses and decision-path tracking analyses.
/subjectContacts/contacts/**	Create, read, update, and delete subject contact records.
/subjectContacts/traces/**	Create, read, update, and delete variable assignment records and variable change path records that are generated by scenario tests for decisions.
/treatmentDefinitions	View the Treatment Definitions category.
/treatmentDefinitions/ definitions/**	Create, read, update, and delete treatment definitions.
/treatmentDefinitions/ definitionGroups/**	Create, read, update, and delete treatment group definitions.

Granting Access to Test Results

Granting Access to Tests That Are Created in SAS Intelligent Decisioning 5.3 or Later

When you create a new rule set test or decision test in SAS Intelligent Decisioning 5.3 or later, you can specify a folder in which the test definition and the test results are stored. For tests that are saved in folders, access is based on the permissions for the folders, the object types, or the specific objects as described in “Full Access and Service Endpoints” on page 54.

Note: You can control access to tests that are created in SAS Intelligent Decisioning 5.4 or later by using the SASScoreUsers group. However, it is recommended that you store test definitions and results in folders. The ability to store tests outside of folders and the use of the SASScoreUsers group for rule set and decision tests is supported for legacy purposes only.

Granting Access to Tests That Were Created in SAS Decision Manager 5.1 or 5.2 on SAS Viya

In SAS Decision Manager 5.1 and 5.2, you could not save test definitions and test results in a folder. By default, SAS Decision Manager created rules that gave only the user who created a test permission to view, update, or delete the test definition and to run the test. Only a user who ran a test could view the test results and run rule-fired analyses or decision-path tracking analyses.

To grant other users access to test definitions or test results that were created in SAS Decision Manager 5.1 or 5.2, do one of the following:

- Add the users to the SASScoreUsers group. SAS Intelligent Decisioning configures the SASScoreUsers group automatically. Members of this group have full access to test definitions and results. These permissions enable access through the user interface, the score definition service, and the score execution service. For instructions on adding users to a group, see "Manage Custom Groups" in *SAS Environment Manager: User's Guide*.
- Create rules in SAS Environment Manager that grant the users access to the URIs that were generated when a particular test was run. The Test Results page, Rule-Fired Analysis page, and Decision Path Tracking page for a test list the URIs to all of the test results. Specify the URIs of the results in the **Object URI** field in the New Rule window. See "Rules Page" in *SAS Environment Manager: User's Guide* for more information.

  Note: Each time a test is run, the IDs for the test results are regenerated. Therefore, the URI to the test results changes.

Define Asset Approval Workflow User Groups

If you enable the SID Asset Approval workflow by turning on the sas.decisions.workflow.enabled configuration property, then you must define the custom user groups shown in Table 7 on page 58. Define these groups in SAS Environment Manager, and assign the appropriate users to each group.

Membership in these groups is required to be able to set the workflow status values for a decision version. The group determines which status changes the group's members can make for a version that is under development.

To create the workflow user groups:

1 Sign in to SAS Environment Manager as an administrator.

   Note: If you are already logged in to SAS Intelligent Decisioning as an administrator, access SAS Environment Manager by clicking ≡ and selecting **Manage Environment**.

2 Click to navigate to the Users category view.

3 In the **View** menu, select **Custom groups**.

4 Complete the following steps for each group listed in Table 7 on page 58:

   a Click . The New Custom Group window appears.

b Enter a name for the group.

c Enter the group ID as it appears in Table 7 on page 58.

d (Optional) Enter a description for the group.

e Click **Save**.

f Add the appropriate users to the new group. For instructions, see "Add or Remove Custom Group Members" in *SAS Environment Manager: User's Guide*.

Table 7 SAS Intelligent Decisioning Asset Approval Workflow Groups

Group ID	Permissions
`SIDWFAuthor`	Change status values: ■ from Developing to Review-ready ■ from Review-ready to Developing
`SIDWFReviewer`	Change status values: ■ from Review-ready to Approved ■ from Review-ready to Developing
`SIDWFDeployer`	Change status values: ■ from Approved to Deployment-ready ■ from Deployment-ready to Deployed ■ from Deployment-ready to Approved ■ from Deployment-ready to Developing ■ from Approved to Developing
`SIDWFAdmin`	Make all status changes **TIP** Members of the groups listed in "Granting Access to the History of Workflow Status Changes" on page 58 also have permission to make all status changes and to view workflow status change histories.

Granting Access to the History of Workflow Status Changes

Groups That Can Access Workflow Status Changes

To display the history of workflow status changes for decisions, users must be a member of one of the following groups:

- SAS Administrators. Members of this group can view and modify workflow history items, but they have access to workflow history items only if the Workflow Administrator group is not defined. For more information, see "Predefined Custom Groups" in *SAS Viya: Identity Management*.
- Application Administrators. Members of this group can view and modify workflow history items, but they have access to workflow history items only if the Workflow Administrator group is not defined. For more information, see "Predefined Custom Groups" in *SAS Viya: Identity Management*.
- Workflow Administrator. Members of this group can view and modify workflow history items. For more information, see "Configure the Default Workflow Administrator Group" in *SAS Workflow Manager: Administrator's Guide*.
- WorkflowHistoryView. Members of this group can view workflow history records, but they cannot modify the workflow history. This group can view workflow history items even if the Workflow Administrator group is defined. For more information, see "Define the WorkflowHistoryView Administrators Group" on page 59.

Define the WorkflowHistoryView Administrators Group

1 Sign in to SAS Environment Manager as an administrator.

Note: If you are already logged in to SAS Intelligent Decisioning as an administrator, access SAS Environment Manager by clicking ≡ and selecting **Manage Environment**.

2 Click to navigate to the Users category view.

3 In the **View** menu, select **Custom groups**.

4 Click . The New Custom Group window appears.

5 Enter a name for the group.

6 Enter the group ID `WorkflowHistoryView`.

7 (Optional) Enter `View the workflow status change history` as the group description.

8 Click **Save**.

9 Add the appropriate users to the new group. For instructions, see "Add or Remove Custom Group Members" in *SAS Environment Manager: User's Guide*.

Set Permissions for Check-Out Folders

TIP If your site needs separate folders that are enabled for the check-out and commit feature, it is recommended that you create these folders in the `/Decision Repository` folder. Subfolders in the repository inherit the default settings for the decision repository.

If you create check-out folders outside of the `/Decision Repository` folder, you must manually set permissions for these folders in order to force users to check out the objects in these folders before modifying the objects. To manually set the permissions for check-out folders:

1 Click ☰, and select **Manage Environment**.

2 Click 🗎, navigate to the folder, and copy the URI of the folder into your paste buffer. You paste this URI into the **Object URI** or **Container URI** fields in the New Rule window in subsequent steps.

3 Click ⚟ to display the **Rules** page.

4 Click ✱, and enter the values shown in the following table in the New Rule window.

Table 8 Permissions for Managing Folder Subdirectories

Field	Value
Object URI	Enter the URI of the check-out folder: `/folders/folders/`*`guid`*
Principal type	**Authenticated Users**
Rule type	**Grant**
Permissions	**Read**, **Add**
Description	`Enable authenticated users to read and add subdirectories in the check-out folder.`

5 Click **Save**.

6 Click ✱, and enter the values shown in the following table in the New Rule window.

Table 9 Permissions for Managing Folder Contents

Field	Value
Container URI	Enter the URI of the check-out folder: `/folders/folders/`*`guid`*
Principal type	**Authenticated Users**
Rule type	**Grant**
Permissions	**Read**, **Add**
Description	`Enable authenticated users to add items to and read objects from the check-out folder.`

7 Click **Save**.

8 Click ✱, and enter the values shown in the following table in the New Rule window.

Table 10 Permissions for Committing Updated Versions

Field	Value
Container URI	Enter the URI of the check-out folder: `/folders/folders/`*`guid`*
Principal type	**Authenticated Users**
Rule type	**Grant**
Condition	`requestUri().endsWith('/revisions')`
Permissions	**Update**
Description	`Enable authenticated users to create new versions in the check-out folder.`

9 Click **Save**.

10 Click [✱, and enter the values shown in the following table in the New Rule window.

Table 11 Permissions for Committing Lookup Tables

Field	Value
Container URI	Enter the URI of the check-out folder: `/folders/folders/`*`guid`*
Principal type	**Authenticated Users**
Rule type	**Grant**
Condition	`requestUri().startsWith('/referenceData/domains/') && requestUri().contains('/contents') && ! requestUri().endsWith('/entries')`
Permissions	**Update**
Description	`Enable authenticated users to commit lookup tables that are in the check-out folder.`

11 Click **Save**.

12 Click [✱, and enter the values shown in the following table in the New Rule window.

Table 12 Permissions for Activating Treatment Groups

Field	Value
Container URI	Enter the URI of the check-out folder: `/folders/folders/`*`guid`*
Principal type	**Authenticated Users**
Rule type	**Grant**
Condition	`requestUri().startsWith('/treatmentDefinitions/definitionGroups/') && requestUri().endsWith('/active')`
Permission s	**Update**
Description	`Enable authenticated users to activate treatment groups that are in the check-out folder.`

13 Click **Save**.

Grant Permission to Create Versions in Check-Out Folders by Using Tags

For information about using tags to create a new version of a decision, see "Creating a New Version by Using Tags" in *SAS Intelligent Decisioning: User's Guide*.

1 Sign in to SAS Environment Manager as an administrator.

Note: If you are already logged in to SAS Intelligent Decisioning as an administrator, access SAS Environment Manager by clicking ≡ and selecting **Manage Environment**.

2 Create a new group for the users to whom you want to grant permission. For instructions, see "Custom Groups" in *SAS Viya: Identity Management*.

3 Click to display the **Rules** page.

4 Click to open the New Rule window, and enter the values shown in the following table.

Table 13 Permissions for Creating Decision Versions in Check-Out-Enabled Folders by Using Tags

Field	Value
Object URI	Enter the following capability: `/capability/sas/sid/createVersionFromTagInCheckoutFolder`

Field	Value
Principal type	**Group**
Principal	Enter the name of the group that you created in Step 2.
Rule type	**Grant**
Permissions	**Read**
Description	`Enable users in the group` *`group-name`* `to create new versions of decisions that are in a check-out folder by using tags.`

5 Click **Save**.

Reloading Lookup Tables When CAS Is Restarted

IMPORTANT All the CAS configuration instances are disabled by default, which means that processing is not enabled for any modifications to the configuration instances in SAS Environment Manager. In order to modify the cas-shared-default configuration instances, a Kubernetes administrator must set SAS_ALLOW_ADMIN_SCRIPTS to True in `sas-shared-config configMap`. For instructions, see the file **`$deploy/sas-bases/overlays/sas-programming-environment/README.md`**. For more information, see “Edit Server Configuration Instances” in *SAS Environment Manager: User’s Guide*.

Also, you must restart the CAS server after you make changes to the cas-shared-default configuration settings.

Whenever SAS Cloud Analytic Services (CAS) is restarted, you must either manually reactivate lookup tables, or you must enable them to be reloaded automatically. To enable the lookup tables to be reloaded automatically, add the formats library to the formats search path. The name of the formats library is specified by the sas.referencedata.casformats.formatsLibrary configuration property. The default formats library is USERFORMATS3. To add the default formats library to the formats search path:

1 Sign in to SAS Environment Manager as an administrator.

Note: If you are already logged in to SAS Intelligent Decisioning as an administrator, access SAS Environment Manager by clicking ☰ and selecting **Manage Environment**.

2 Click 🔧 to navigate to the Configuration category view.

3 In the **View** menu, select **Definitions**.

4 Select the **sas.cas.instance.config** definition.

5 Click ✎ beside the **cas-shared-default: startup** configuration instance to edit its properties. The Edit Configuration window appears.

6 Copy and paste the following code into the **contents** field.

```
s:sessionProp_addFmtLib
{caslib="Formats",fmtLibName="userformats3",name="userformats3.sashdat",promote=true}
newFmtSearch = " userformats3"
newFmtSearch = ((cas.fmtsearch or "") .. " " .. newFmtSearch)
s:configuration_setServOpt
{fmtsearch=newFmtSearch}

s:table_loadTable{caslib="Formats", casOut={caslib="Formats",replication=0.0},
path="userformats3.sashdat", promote=true}
```

7 Click **Save**.

8 Restart CAS. See "SAS Cloud Analytic Services: How To" in *SAS Viya: SAS Cloud Analytic Services* for more information.

Note: The values that are specified for the CAS library and format library must match the values that are specified for the casformats.backupLibrary and casformats.formatsLibrary configuration properties. For more information, see "Reference Data Service Properties" on page 15.

For more information, see "Configuration Page" in *SAS Environment Manager: User's Guide*, *SAS Viya Platform: Configuration Properties*, and "Managing User-Defined Formats in SAS Viya" in *SAS Viya: Data*.

Managing Test Data

When you run a rule set, model, or decision test, several files are created. The URI to the test definition and all of the test results are displayed on the Test Results page.

By default, when you re-run an existing test, the previous test results are not deleted before the new results are generated. This behavior is controlled by the `deleteExecutions` configuration property. To automatically delete test results when a test is re-run, set the `deleteExecutions` configuration property to True. See "Score Execution Service Properties" on page 17 for more information.

When a test definition is deleted, the associated test results are normally deleted. However, the deletion transaction might be interrupted or the user might not have permission to delete output tables on CAS. To delete results files such as log files, code files, and CAS tables that are not deleted when the associated test is deleted, use the `sas-scoreexecution-cli` command-line interface. See "scoreexecution Plug-In" in *SAS Intelligent Decisioning: Command-Line Interfaces* for more information.

THE POWER TO KNOW®