THE POWER TO KNOW.

SAS® Intelligent Decisioning: User's Guide

2023.05 - 2023.06*

* 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.

SAS® Documentation
August 15, 2023

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2023. *SAS® Intelligent Decisioning: User’s Guide*. Cary, NC: SAS Institute Inc.

SAS® Intelligent Decisioning: User’s Guide

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

August 2023

v_029-P2:edmug

Contents

Chapter 1 / *Introduction to SAS Intelligent Decisioning* **1**
Enterprise Decision Management Systems 2
About Business Rules 2
SAS Intelligent Decisioning Features 2
Support for Models in Decisions 4
Sign in to SAS Intelligent Decisioning 4
SAS Intelligent Decisioning Home Page 5
Using SAS Intelligent Decisioning as a Progressive Web App 5
Sorting, Searching, and Filtering in Category Views 6
SAS Intelligent Decisioning Settings 8
Manage Folders and Folder Content 9
View the Properties of Any Object 9
Manage Comments for Any Object 10

Chapter 2 / *Working with Rule Sets* **11**
About Rules and Rule Set Types 13
About Condition and Action Expressions 14
Create a New Rule Set 15
Managing the Variables in a Rule Set 16
Defining New Rules in a Rule Set 24
Defining Expressions in Rules and Assignment Statements 33
Managing Rules 41
Copy a Rule Set URL 43
Compare Rule Set Content 43
Compare Rule Set Code 45
Generate PDF Documentation for a Rule Set 46
Managing Rule Sets 47
Managing Versions of Rule Sets 49
Determine Which Objects Use a Rule Set 52
Checking Out and Committing Rule Set Versions 53
Managing Version Tags for Rule Sets 58
Publishing a Rule Set 59
Testing Rule Sets 63
Manage Comments for a Rule Set Test 76
Executing Published Content 77
Importing and Exporting Rule Sets 80

Chapter 3 / *Working with Treatments and Treatment Groups* **83**
About Treatments and Decisions 84
Example: A Decision That Includes a Treatment Group 85
About Attributes, Eligibility Rules, and Effective Dates 88
Arbitrating Treatments 89
About Channels 90
Define a Treatment 90
Define a Treatment Group 91
Activating a Treatment Group 93
Managing Attributes and Aliases 95
Delete Treatments from a Treatment Group 97

Managing Eligibility Rules for a Treatment 97
Copy a Treatment or Treatment Group URL 98
Compare Treatment or Treatment Group Content 99
Compare Treatment Group Code 101
Generate PDF Documentation for a Treatment Group 101
Managing Treatments and Treatment Groups 102
Managing Versions of Treatments and Treatment Groups 104
Determine Which Objects Use a Treatment or Treatment Group 106
Checking Out and Committing Treatment or Treatment Group Versions 107
Managing Version Tags for Treatment and Treatment Groups 111

Chapter 4 / *Working with Lookup Tables and Functions* **113**
About Lookup Tables and Functions 114
About Labels for Lookup Keys and Values 115
Predefined Lookup Tables 116
Create a New Lookup Table 116
Importing and Exporting Lookup Tables 117
Add Lookup Table Entries 119
Edit Lookup Table Entries 120
Delete Lookup Table Entries 121
Edit Lookup Key or Value Labels 121
Copy a Lookup Table URL 121
Compare Lookup Table Content 121
Generate PDF Documentation for a Lookup Table 123
Managing Lookup Tables 123
Managing Versions of Lookup Tables 125
Determine Which Objects Use a Lookup Table 128
Checking Out and Committing Lookup Table Versions 128
Managing Version Tags for Lookup Tables 133
Activating a Lookup Table 134
Dictionary 137

Chapter 5 / *Managing Global Variables* **141**
Using Global Variables 141
Create a New Global Variable 142
Managing Global Variables 143
Managing Versions of Global Variables 144
Determine Which Objects Use a Global Variable 146
Managing Version Tags for Global Variables 146
Manage the Global Variable Recycle Bin 147
Activating a Global Variable 148

Chapter 6 / *Using Custom Code Files* **151**
Introduction to Custom Code Files 152
DS2 Code Files 153
Python Code Files 157
Data Query Files 159
Custom Context Files 163
Create a New Code File 167
Managing the Variables in a DS2 or Python Code File 168
Validate SQL, DS2, or Context Files 175
Copy a Code File URL 176
Compare Generated Code for Code Files 176
Managing Code Files 177
Managing Versions of Code Files 178
Determine Which Objects Use a Code File 181

Checking Out and Committing Code File Versions . . . 182
Managing Version Tags for Code Files . . . 186
Testing Code Files . . . 187
Manage Comments for a Code File Test . . . 195

Chapter 7 / Using Custom Functions . . . 197
About Custom Functions . . . 198
Using Custom Functions in a Rule Set or Code File . . . 198
Create a Custom Function . . . 199
Using the Custom Function Editor . . . 200
Rules for Writing Custom Function Files . . . 201
Managing the Variables in a Custom Function File . . . 202
Validate a Custom Function File . . . 207
Testing a Custom Function . . . 207
Compare Custom Functions . . . 208
Managing Custom Function Categories . . . 208
Managing Custom Functions . . . 209
Restore or Permanently Delete Hidden Functions or Hidden Categories . . . 211

Chapter 8 / Working with Decisions . . . 213
About Decisions . . . 215
Using SAS Workflow with SAS Intelligent Decisioning . . . 216
Create a Decision . . . 220
Views for Editing a Decision . . . 221
Managing the Variables in a Decision . . . 222
Adding Objects to a Decision . . . 230
Mapping Variables within a Decision . . . 247
Searching for Objects in a Decision Diagram . . . 251
Editing the Objects in a Decision . . . 251
Update Decisions to Use New Object Versions . . . 253
Validate a Decision in the Decision Editor . . . 254
Change the Workflow Status of a Decision . . . 254
Copy a Decision URL . . . 255
Compare Decision Content . . . 255
Compare Decision Code . . . 258
Generate PDF Documentation for a Decision . . . 258
Managing Decisions . . . 260
Managing Versions of Decisions . . . 262
Determine Which Objects Use a Decision . . . 271
Checking Out and Committing Decision Versions . . . 272
Managing Version Tags for Decisions . . . 277
Publishing a Decision . . . 278
Testing Decisions . . . 284
Manage Comments for Decision Nodes and Tests . . . 306
Executing Published Content . . . 307
View All Published Decisions . . . 310

Appendix 1 / Querying the Subject Contact Database . . . 313
About the Subject Contact Database . . . 313
How to Query the Subject Contacts Database . . . 314
Using Views Defined by SAS Intelligent Decisioning . . . 315
Columns in the dcm_trt_fixed_attr_vw and dcm_trt_dynamic_attr_vw Views . . . 316
Columns in the dcm_treatments_contacts_vw View . . . 318

Appendix 2 / Import File Formats . . . 325
Introduction . . . 325

CSV Format of the Variable Import File 325
JSON Format of the Variable Import File 327

1

Introduction to SAS Intelligent Decisioning

Enterprise Decision Management Systems **2**
About Business Rules **2**
SAS Intelligent Decisioning Features **2**
Support for Models in Decisions **4**
Sign in to SAS Intelligent Decisioning **4**
SAS Intelligent Decisioning Home Page **5**
Using SAS Intelligent Decisioning as a Progressive Web App **5**
Benefits of Using a Progressive Web App 5
Installing SAS Intelligent Decisioning as a Progressive Web App 6
Sorting, Searching, and Filtering in Category Views **6**
Sort Object Lists 6
Search Object Lists 7
Filter Object Lists 7
Manage Saved Filters 7
SAS Intelligent Decisioning Settings **8**
Manage Folders and Folder Content **9**
View the Properties of Any Object **9**
Manage Comments for Any Object **10**

Enterprise Decision Management Systems

Enterprise decision management systems can transform how businesses make decisions. They enable businesses to use the information they already have to make better decisions—decisions that are based on predictive analytics rather than on past history. Decision management systems automate the process of making decisions, particularly day-to-day operational decisions. They improve the speed, efficiency, and accuracy of routine business processes, in part by reducing the need for human intervention. Automating decisions with SAS Intelligent Decisioning provides a streamlined mechanism for controlling and monitoring the rules and processes used by your organization. By automating decisions, organizations in every industry can improve interactions with customers, partners, suppliers, and employees. In addition, organizations that are highly regulated, such as financial services, health care, and insurance, can more easily achieve compliance as a result of repeatable, traceable decisions. Explicitly defining a decision makes your organization's decision-making process transparent, and enables you to monitor the process for accuracy.

SAS Intelligent Decisioning helps organizations leverage data, author and manage business rules, and integrate analytical models with other business logic. It enables them to create and optimize efficient business decisions within a single interface that gives users a consistent experience.

About Business Rules

Business rules capture the logic of business decisions and are a core component of decision management systems. Business rules enable you to codify the decision-making process used by your organization. Business rules make the decision-making process transparent and adaptable, enabling organizations to respond quickly to new information about customers and markets. They enable organizations to identify and deal with fraud, avoid unnecessary risk, and find opportunities hidden in customer data.

SAS Intelligent Decisioning Features

You can use SAS Intelligent Decisioning to create a database of business rules, combine those rules together into decisions, and publish the decisions for use by other applications. SAS Intelligent Decisioning provides the following capabilities:

business rule authoring
: A business rule specifies conditions to be evaluated and action to be taken if those conditions are satisfied. For example, you can create a rule that determines whether a particular customer has a mortgage. That same rule can then add the outstanding balance of the mortgage to a running total of the customer's debt. With SAS Intelligent Decisioning, you define the conditions and actions for each rule.

rule set management and publishing
: A rule set is a logical collection of rules. A single rule set can have many rules, but it generally corresponds to a single step in a decision. For example, you can have a rule set that determines a customer's asset balance and another rule set that determines a customer's debt level. You can use SAS Intelligent Decisioning to easily create new rule sets, reorder the rules in a rule set, add new rules to existing rule sets, and so on. When a rule set is published, the versioning features of SAS Intelligent Decisioning create a static version of the rule set. This static version helps you enforce integrity and governance over the rules that are put into production.

treatment authoring and management
: A treatment is an offer that can be sent to a customer as part of an inbound marketing campaign. For example, when a customer visits your company's website, a customer service application can capture information about the customer. The application can invoke a decision that includes a set of treatments and pass the customer's information to the decision. The decision determines which of the treatments (offers) are suitable to present to that customer at that time. With SAS Intelligent Decisioning, you can author treatments, combine them into treatment groups, and add the groups to decisions.

lookup table authoring and management
: Lookup tables are tables of key-value pairs. You can use lookup tables for tasks such as retrieving a part name based on a part number or retrieving a location name based on a ZIP code. SAS Intelligent Decisioning provides two predefined lookup tables for use with treatments: one for treatment channels and one for subject levels.

code file authoring and management
: You can define custom code files to do things that are not possible in rules, models, or treatments. For example, you can define a code file that makes HTTP calls to REST APIs, interacts with a database, manipulates files in the file system, or performs custom data transformations. In SAS Intelligent Decisioning, you can easily create, edit, and manage custom data query files, DS2 files, and Python files. You can create and edit data query files by using the SQL editor or by launching SAS Studio.

global variable authoring and management
: You can use global variables to define common variables for use in multiple rule sets, models, and decisions. For example, you might want to define global variables for benchmark interest rates, current exchange rates, or inventory levels for specific products. By using global variables, you can set these values in one place and refer to the variables in multiple objects.

custom function authoring and management
: You can use custom functions to perform actions that are not available with the standard functions that SAS provides. Custom functions also enable you to encapsulate and reuse business logic. You can use custom function in rule sets and in DS2 code files.

decision authoring and publishing
: You can combine rule sets, analytical models, treatment groups, custom code files, record contacts nodes, and branches (conditional logic) into decisions. You can also add a decision to another decision. You can investigate various scenarios, test and refine the decision logic, and then publish the decisions for use in batch applications and online transactions. After a decision has been published, it is available for use by other applications.

Support for Models in Decisions

Support for a model depends on the tool that you use to create the model and the model's score code type. For example, decisions that use certain model score code types cannot be published or can be published only to SAS Micro Analytic Service destinations. For more information, see "High-Level Model Support Matrix for Primary Functions" in *SAS Model Manager: User's Guide*.

Sign in to SAS Intelligent Decisioning

Note: If you are already signed in to SAS Drive, you can access SAS Intelligent Decisioning by clicking ≡ and selecting **Build Decisions**.

To sign in to SAS Intelligent Decisioning:

1 In the address bar of your web browser, enter the URL for SAS Intelligent Decisioning and press **Enter**. The Sign In page appears.

   Note: Contact your system administrator if you need the URL for SAS Intelligent Decisioning. The default URL is http://*host_name*/SASDecisionManager.

2 Enter a user ID and password.

3 Click **Sign In**. If this is your first time signing in to SAS Intelligent Decisioning, the Welcome to SAS window appears.

4 Do one of the following:

   - Set up a profile and customize user-specific settings. For more information, see "Settings" in *SAS Viya Platform: General Usage Help for Web Applications* under General Usage in the Help Center and "SAS Intelligent Decisioning Settings" on page 8.
   - Click **Skip setup** if you do not want to set up a profile.You can edit your profile settings later by clicking the user icon in the upper right.

   After you exit the profile setup page, the SAS Intelligent Decisioning home page appears.

SAS Intelligent Decisioning Home Page

By default, the SAS Intelligent Decisioning Home page appears when you sign in to the application. To return to the home page at any time, click in the left navigation bar.

The Home page displays up to 100 of the most recent objects that you have opened, the date on which each object was last modified, and the user ID of the user who modified it. You can click and to switch between the list view and tile view of the recent objects. In the tile view, you can click to display additional information for a specific object. To open an object, either click on the object name, or click ⋮ and select **Open**.

You can toggle the display of category labels in the left navigation bar by clicking « and ». To create a new object, click and select the object type. You can sort the list of recent objects by name or by date modified, and you can filter the list by name.

The Stay Connected panel on the right contains links to new topics in SAS Support Communities pages. It also contains links to an overview video and to documentation for SAS Intelligent Decisioning.

Using SAS Intelligent Decisioning as a Progressive Web App

Benefits of Using a Progressive Web App

The benefits of using SAS Intelligent Decisioning as a Progressive Web App (PWA) include the following:

- Application persistence – By default, your session will never time out, so you can restart your work more quickly. Your administrator can control the timing with a configuration property in SAS Environment Manager.
- Performance – When installed as a PWA, SAS Intelligent Decisioning typically starts faster than when accessed in the browser.
- Desktop experience – Installing SAS Intelligent Decisioning as a PWA confers all the benefits of a traditional installation. You can launch SAS Intelligent Decisioning from the Start menu or Taskbar, and you do not need to sort through countless tabs to find the correct instance of SAS Intelligent Decisioning.
- Renaming – You can rename each PWA instance to quickly access different environments, such as development, test, or production servers.

Installing SAS Intelligent Decisioning as a Progressive Web App

SAS Intelligent Decisioning must be deployed with TLS and HTTPS enabled.

To install SAS Intelligent Decisioning as a PWA:

1. Open a Chromium-based web browser and sign in to SAS Intelligent Decisioning.
2. Open the web browser’s menu and select **Install SAS**.

   Note: The **Install SAS** option might be located under another menu option. For example, in Microsoft Edge it is located under **Apps**.

SAS Intelligent Decisioning is now installed as a desktop program named **SAS**. After one web application has been installed as a PWA, all other SAS applications are accessible in the PWA.

To uninstall the PWA for SAS Intelligent Decisioning:

1. Open the PWA for SAS Intelligent Decisioning.
2. In the PWA menu, select **Uninstall SAS**.

Sorting, Searching, and Filtering in Category Views

You can search, filter, and sort the list of objects in any category view.

Sort Object Lists

To sort the object list based on the values of a column, right-click the column heading, and select **Sort** ⇨ **Sort (ascending)** or **Sort (descending)**. If the column is sorted in ascending order, ↑ appears beside the column heading. When the column is sorted in descending order, ↓ appears. To add another column to the sort, right-click the column heading, and select **Add to sort** ⇨ **Sort (ascending)** or **Sort (descending)**. To clear a column filter, right-click the column heading and select **Sort** ⇨ **Remove sort**.

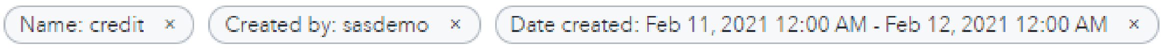
Search Object Lists

To search the **Name** column, enter the string for which you want to search in the search field above the object list, and click ⌕. The search is case insensitive. To clear the search text, click ⊗.

TIP See also "Searching for Objects in a Decision Diagram" on page 251.

Filter Object Lists

To filter the view, click **Filter**, enter or select the filter criteria in the Filter window, then click **Filter**. SAS Intelligent Decisioning filters the object list and displays a token for each filter criteria. For example:


Name: credit × Created by: sasdemo × Date created: Feb 11, 2021 12:00 AM - Feb 12, 2021 12:00 AM ×

TIP If you specify a folder location as part of the filter criteria, the filter searches the selected folder and recursively searches all of its subfolders.

The node count for a decision does not include any nodes that are in nested decisions.

Filters remain in effect until you remove them. To remove a specific filter, click × for that filter. To remove all filters, click ✕ at the right of all of the filter tokens.

Manage Saved Filters

You can save a filter, and then reapply that filter to a category view later without having to redefine the filter criteria. You can save up to 10 filters.

To save a filter:

1 Do one of the following:

- Filter the object list as described in "Filter Object Lists", click ▾ beside the **Filter** button, and then click **Save filter**.
- Click **Filter**, enter or select the filter criteria in the Filter window, and then click **Save filter**.

The Save Filter window appears.

2 Enter a name for the filter. Filter names are limited to 100 characters and can contain only alphanumeric characters and underscores (_).

3 (Optional) Enter a description of the filter.

4 Click **Save and filter** to save the filter and apply the filter to the object list.

To apply a saved filter to a category's object list, click ▾ beside the **Filter** button, and select the filter that you want to use. If the filter specifies criteria that do not apply to the current object list, those criteria are ignored. For example, if you apply a filter that specifies a criterion for location to the Global Variables category view, that criterion is ignored.

To edit a saved filter:

1 Apply the saved filter, and click **Filter** to open the Filter window.

2 Edit the filter criteria, and click **Save filter**.

To delete a saved filter, click ▾ beside the **Filter** button, and click × for the filter that you want to delete.

SAS Intelligent Decisioning Settings

In addition to the global settings that are available with SAS Viya applications, SAS Intelligent Decisioning provides two additional settings. To access these settings, click the user icon in the top right corner of the application window, select **Settings** to open the Settings window, and click **SAS Intelligent Decisioning**.

- By default, if you add an object to a decision and that decision does not already have a variable of the same name and data type as that object's variable, then SAS Intelligent Decisioning displays the following message: `Some objects in the decision define variables for which no corresponding decision variables have been created`. You can turn on the **Create variables automatically in decisions** setting if you want SAS Intelligent Decisioning to automatically create decision variables. For more information, see "Mapping Variables within a Decision" on page 247.
- When you edit custom functions, custom code files, and rule expressions, the editor can display a list of SAS functions, custom functions, and variables whose names match the letters that you type (an autocomplete list is displayed). By default, the editor displays SAS function names and custom function names in the autocomplete list. You can control what is displayed in this list by selecting or clearing the check boxes for **SAS functions**, **Custom functions**, and **Variables**. If you clear all three check boxes, SAS Intelligent Decisioning does not display the autocomplete list.

Note: In order for changes to settings to take effect, you must sign out and sign back in to SAS Intelligent Decisioning.

Manage Folders and Folder Content

In the Manage Folders window, with the appropriate permissions, you can create new folders, delete, move, and rename existing folders or objects, and restore objects from the recycle bin. Objects that you delete in SAS Intelligent Decisioning are moved to the recycle bin. For information about permissions, see “Managing Permissions” in *SAS Intelligent Decisioning: Administrator’s Guide*.

Note: When you delete a specific version of an object, that version is permanently deleted. It is not moved to the recycle bin, and it cannot be recovered. When you delete an entire object, that object is moved to the recycle bin and can be restored.

To open the Manage Folders window, click ⋮ , and select **Manage folders** in any category view except the Custom Functions and Global Variables category views.

- To create a new folder, click , type the folder name, and press Enter.
- To rename an item, right-click the item, select **Rename**, type the new name, and press Enter.
- To move an item, right-click the item, select **Move to folder**, select the new folder, and press Enter.
- To delete an item, right-click the item, and select **Delete**. The deleted item is moved to the recycle bin.
- To restore an item from the recycle bin, right-click on the item in the recycle bin, and select **Restore**.
- To permanently delete an item, right-click on the item in the recycle bin, and select **Delete**. The item is deleted from the recycle bin and cannot be restored.
- To empty the recycle bin, right-click **Recycle Bin** and select **Empty recycle bin**.

For more information about folders, see *SAS Viya Platform: Folders*.

For information about the global variables recycle bin, see “Manage the Global Variable Recycle Bin” on page 147. For information about hiding custom function categories, see “Hide A Custom Function Category” on page 209.

View the Properties of Any Object

In the category views, you can view the properties of any object. Select the check box for the object, and click in the property pane. Click » to close the properties pane.

Manage Comments for Any Object

In the category views, you can associate comments and attachments with any object.

To open the **Comments** pane for an object, select the check box for the object, and click in the properties pane.

To add a new comment, enter the comment in the text box and click **Post**.

To add an attachment to a comment, click , select the file that you want to attach, and click **Post**. (The attachment icon appears after you enter text in text box.) You cannot attach executable files such as BAT and EXE files.

To reply to a comment, click **Reply**, enter your reply in the text box, and click **Post**.

To delete a comment, click for that comment.

Note: You can also add comments to rule set tests, code file tests, decision tests, and to the Start and End nodes within a decision. Comments that you add to tests are not displayed with the comments in the category views. Comments that you add to the Start and End nodes in a decision are displayed in the Decisions category view. For more information, see "Manage Comments for a Rule Set Test" on page 76, "Manage Comments for a Code File Test" on page 195, and "Manage Comments for Decision Nodes and Tests" on page 306.

2

Working with Rule Sets

About Rules and Rule Set Types 13

About Condition and Action Expressions 14

Create a New Rule Set 15

Managing the Variables in a Rule Set 16
About Variables 16
Add Variables from a Data Table 18
Add Variables from a Rule Set, Decision, or Code File 19
Add Global Variables to a Rule Set 19
Create a Custom Variable Dynamically 20
Create Custom Variables on the Variables Tab 20
Duplicate a Variable 21
Importing and Exporting Variables 22
Delete Variables 23
Edit Variable Properties 23
Edit Metadata for Data Grid Variables 23
Determine Which Objects Use a Particular Variable 23

Defining New Rules in a Rule Set 24
Add a Stand-Alone Assignment Statement 24
Add an IF-THEN or IF-THEN-ELSE Rule to an Assignment or Common Rule Set . 25
Adding a Common Rule Set Reference to an Assignment Rule Set 26
Add a New Rule in a Filtering Rule Set 29
Copy Rules From Another Rule Set 30
Controlling Which Conditions Are Evaluated 31
How Rules Are Evaluated and When Rule-Fired Records Are Generated 32

Defining Expressions in Rules and Assignment Statements 33
About Defining Expressions 33
Using the Expression Editor 33
Enter LOOKUP and LOOKUPVALUE Expressions 35
Entering Literal Data Values 36
Operators for Use in Expressions 37
Using the LIKE Operator 38
Using Functions in Expressions 40
Working with Missing Values 40
Delete Expressions or ELSE Rules 41

Managing Rules 41
Duplicate a Rule 41
Delete a Rule 41
Rename a Rule 42
Edit a Rule Description 42
Reorder Rules 43

Copy a Rule Set URL 43

Compare Rule Set Content 43

Compare Rule Set Code 45

Generate PDF Documentation for a Rule Set 46

Managing Rule Sets 47
Duplicating Rule Sets 47
Delete Rule Sets 48
Rename Rule Sets 48
Move Rule Sets to a Different Folder 48

Managing Versions of Rule Sets 49
Set the Displayed Version 49
Create a New Version 49
Create a New Version Automatically When You Publish a Rule Set 50
Copy the Content of a Version 50
Delete a Version 51
Upgrade Decisions to Use a New Version of a Rule Set 52

Determine Which Objects Use a Rule Set 52

Checking Out and Committing Rule Set Versions 53
About Checking Out and Committing Versions 53
Checking Out a Rule Set from within a Decision 54
Checking Out and Committing a Common Rule Set from within an Assignment Rule Set 54
Checking Out and Committing a Lookup Table from within a Rule Set 55
Committing a Lookup Table Together with a Rule Set 55
Concurrently Checking Out and Committing Rule Set Versions 56
Check Out and Commit a Rule Set Version 56
Determine Who Has a Version Checked Out 57
Opening the Original Object 57
Undoing a Check Out 57

Managing Version Tags for Rule Sets 58
Add a Version Tag 58
Remove a Version Tag 59
Modify a Version Tag 59

Publishing a Rule Set 59
Introduction to Publishing 59
Publish Rule Sets 60

Testing Rule Sets 63
Ways to Test a Rule Set 63
Test a Rule Set 63
Test a Scenario 67
Run a Rule-Fired Analysis 70
Validate a Published Rule Set 73
Working with Test and Validation Test Output Data 75

Status Icons for Tests . . . 76

Manage Comments for a Rule Set Test . . . **76**

Executing Published Content . . . **77**
Executing Content Published to SAS Micro Analytic Service Destinations . . . 77
Executing Content That Has Been Published to SAS Cloud Analytic Services Destinations . . . 77
Content Executed by Published Rule Sets . . . 79

Importing and Exporting Rule Sets . . . **80**
Import a Rule Set . . . 80
Export a Rule Set . . . 81

About Rules and Rule Set Types

A rule specifies conditions to be evaluated, and it can also specify actions to be taken if those conditions are satisfied. Rules are grouped together into rule sets. Rule sets are logical collections of rules that are grouped together because of interactions or dependencies between the rules or because they are processed together after they are published.

A rule set can be one of following types:

filtering rule sets
: contain rules that correspond to the form:

```
IF condition_expressions
```

These rules contain only IF statements. They have conditional expressions but do not have action expressions. Filtering rule sets enable you to select only certain records for processing. Only the records for which the conditions evaluate to True are processed by the remaining objects in the decision.

Filtering rule sets are also used as the eligibility rule set in treatments. In a treatment, the eligibility rule set defines who is eligible to receive the offer that is defined in the treatment.

assignment rule sets
: contain rules that correspond to one of the following forms:

```
ASSIGN variable variable_or_value
```

```
IF condition_expressions THEN action_expressions
```

These rules are either assignment statements, IF-THEN statements, or IF-THEN-ELSE statements. IF-THEN and IF-THEN-ELSE statements have both conditional expressions and action expressions.

Assignment statements contain only an action expression. Assignment statements always execute unless a RETURN action stops the execution of the rule set before execution reaches the assignment statement. See Step 8 of “Add an IF-THEN or IF-THEN-ELSE Rule to an Assignment or Common Rule Set” on page 25 for information about the RETURN action.

Assignment rule sets can also include common rule sets.

common rule sets
: enable you to share the same rules among different assignment rule sets without redefining the rules in each assignment rule set. Common rule sets can contain assignment statements, IF-THEN statements, and IF-THEN-ELSE statements just like an assignment rule set can. You add common rule sets to assignment rule sets. When the assignment rule set executes, the rules in the common rule set execute as if they were defined directly in the assignment rule set. Any change to a common rule set affects all assignment rule sets that include the common rule set.

About Condition and Action Expressions

An assignment rule corresponds to this form:

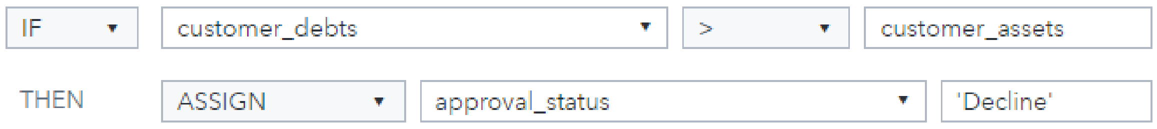
```
IF condition_expressions THEN action_expressions
```

For example, suppose you have the following rule:

```
IF customer_debt > customer_assets THEN approval_status = 'Decline'
```

In this case, `customer_debt > customer_assets` is a condition expression, and `approval_status = "Decline"` is an action expression.

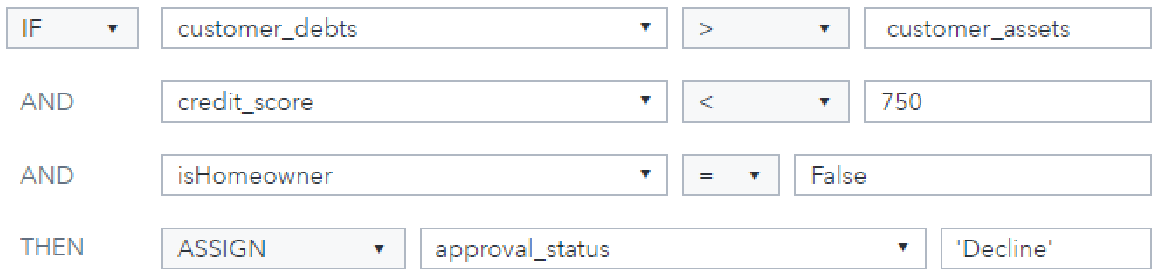
For example, the following figure shows the rule above as it appears in the rule set editor:



IF ▾	customer_debts ▾	> ▾	customer_assets
THEN	ASSIGN ▾	approval_status ▾	'Decline'

Note: Filtering rules do not contain action expressions.

A single assignment rule can contain multiple condition expressions and action expressions. Multiple condition expressions within the same rule are joined together with the AND operand. For example, suppose you define the following rule in SAS Intelligent Decisioning:



IF ▾	customer_debts ▾	> ▾	customer_assets
AND	credit_score ▾	< ▾	750
AND	isHomeowner ▾	= ▾	False
THEN	ASSIGN ▾	approval_status ▾	'Decline'

SAS Intelligent Decisioning generates the following rule:

```
IF ((customer_debts > customer_assets) AND (credit_score < 750)
AND (isHomeowner = false)) THEN approval_status = 'Decline'
```

Create a New Rule Set

1 Click ⊞ to navigate to the Rule Sets category view.

2 Click **New Rule Set**. The New Rule Set window appears.

3 Enter a name for the rule set if you do not want to use the default name. Rule set names are limited to 100 characters and must be unique within a folder. Rule set names are case-sensitive.

Note: Some publishing destinations restrict the characters that can be used in the published name of a rule set. For more information, see Table 2.5 on page 61.

4 Select the rule set type. See “About Rules and Rule Set Types” on page 13 for descriptions of each rule set type.

IMPORTANT You can publish decisions that include filtering rule sets to SAS Micro Analytic Service destinations and to container destinations, but you cannot publish the filtering rule sets themselves to SAS Micro Analytic Service destinations, to Git destinations, or to container destinations.

5 (Optional) Enter a description for the new rule set. Descriptions are limited to 1000 characters.

TIP You can edit the description later on the **Properties** tab.

6 Click 🗀, and select the folder where you want to save the rule set.

7 Click **Save**. SAS Intelligent Decisioning opens the new rule set and displays the **Variables** tab.

TIP Objects that are saved in a folder for which the check-out and commit feature is enabled, such as the `Decision Repository` folder, must be checked out before they can be edited.

8 (Optional) If the rule set is in a folder for which the check-out and commit feature is enabled, click the **Versions** tab and check out the latest version of the rule set. For more information, see “Check Out and Commit a Rule Set Version” on page 56.

9 Add variables and rules to the rule set. For more information, see the following topics:

- “Managing the Variables in a Rule Set” on page 16

- "Add a Stand-Alone Assignment Statement" on page 24
- "Defining New Rules in a Rule Set" on page 24

Managing the Variables in a Rule Set

About Variables

The Properties of a Variable

Property	Description
Name	Variable names must start with a letter or an underscore (_), and they can contain only alphanumeric characters and the underscore. Variable names can be up to 32 characters long. They must be unique within a rule set. **Note:** SAS Intelligent Decisioning does not support double-byte character set (DBCS) characters in variable names. **Note:** Do not use any of these operators or keywords as variable names: AND, OR, IN, NOT, LIKE, TRUE, or FALSE. Do not use _N_ or any DS2 reserved word as a variable name. See "Reserved Words in the DS2 Language" in *SAS DS2 Programmer's Guide* for information about reserved words in the DS2 language.
Data type	SAS Intelligent Decisioning supports the following data types: Boolean, character, data grid, date, datetime, decimal, and integer. Binary and varying-length binary variables are supported only in decisions. Binary variables are supported only as input variables or temporary variables in order to support models that require binary data. For Boolean variables, you can select **`True`** or **`False`** for the initial value. However, SAS Intelligent Decisioning represents Boolean values by using the numbers 1 and 0 in the code that it generates. When SAS Intelligent Decisioning is evaluating Boolean expressions, any non-zero number is considered True. When you are entering expressions, specify `1` for True and `0` for False. **Note:** For content that will be published to a SAS Micro Analytic Service destination, the values of date and datetime input variables must be numeric. They cannot be formatted values such as `18jul2019`. Also, integer variables are converted to decimal variables when the content is published.

Property	Description
Input and **Output**	A variable can be an input variable, an output variable, both, or neither (a temporary variable). See “Input Variables, Output Variables, and Temporary Variables” on page 17 for more information.
Length	The length for Boolean and numeric variable types is set automatically. For output-only data grid variables, the length is set to the value that you specify. For character variables and data grid variables that are input-only or input-output variables, the variableLengthOverridden configuration property determines how the length is set. By default, this property is set to Off, and the length is set to the length in the input data. When the variableLengthOverridden property is set to On, the length of input-only and input-output character variables and data grid variables is set to the value that you specify. For more information, see “sas.businessrules.variableLengthOverridden” in *SAS Intelligent Decisioning: Administrator’s Guide*. The maximum length for character variables (outside of a data grid) and data grid variables is 10485760 characters. The maximum length for character variables within a data grid is 32767 characters.
Initial value	You can specify an initial value for all data types except data grids. Initial values are used only at run time and only for output-only variables. **Note:** For content that will be published to a SAS Micro Analytic Service destination, the values of date and datetime input variables must be numeric. They cannot be formatted values such as `18jul2019`. Also, integer variables are converted to decimal variables when the content is published.
Description	Descriptions are limited to 256 characters.

Input Variables, Output Variables, and Temporary Variables

For each variable, you must specify whether the variable is an input variable, an output variable, both an input variable and an output variable, or a temporary variable.

- Input variables are variables that are present in the input table for a rule set. When a rule set is deployed in a production system, all input variables must be mapped to table columns in input data. When you test a rule set in SAS Intelligent Decisioning, for each input variable, you can either map it to a table column or specify a constant as its input value. If you choose not to map a variable to either a table column or a static value, SAS Intelligent Decisioning displays a warning message. When you create or edit a variable, clear the **Input** check box for any variable that you do not want to be mapped to a column in an input table or for which you do not want to specify a value.

> **TIP** If the sas.businessrules.inputVariableReadOnly configuration option 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. For more information, see "sas.businessrules.inputVariableReadOnly" in *SAS Intelligent Decisioning: Administrator's Guide*.

- Output variables are variables that are written to the output table that is created when a rule set is run. When you create or edit a variable, clear the **Output** check box for any variable that you want to exclude from the output data.
- Temporary variables are variables that are not present in the input data, and they are not written to the output table. To create a temporary variable for use only while a rule set is executing, clear both the **Input** and **Output** check boxes.

When you create a new variable, it is created as an input-output variable by default.

Note: Binary variables are supported in decisions only as input variables or temporary variables in order to support models that require binary data.

Add Variables from a Data Table

1. On the **Variables** tab, click the **Rule Set Variables** subtab.
2. Click **Add variable** and select **Data table**. The Choose Data window appears, and the list of SAS Cloud Analytic Services (CAS) tables that are loaded into memory is displayed on the **Available** tab.

   If the table that you need does not appear in the list of available tables, try the following solutions:

   - If the table appears on the **Data Sources** tab, right-click on the table, and select **Load** to load the table into memory. If the table does not appear on the **Available** tab, click ⟳.
   - If the table does not appear on the **Data Sources** tab, import the data. The process of importing the data loads it into memory. For information about importing data from different sources, see "Making Data Available to CAS" in *SAS Data Explorer: User's Guide*.
3. Select the table from which you want to import variables, and click **OK**. The Add Variables window appears.
4. Select the variables that you want to import and click +›. To import all of the variables in the table, click +».
5. Click **Add** to add the select variables, or click **Add and replace** to replace existing variables that have the same name.

6 On the **Variables** tab, select or clear the **Input** and **Output** check boxes as necessary. See "Input Variables, Output Variables, and Temporary Variables" on page 17 for more information.

> **TIP** To filter the variable list, right-click on the **Variable** column heading, enter a filter string in the **Filter** box, and press Enter. To clear the filter, delete the string from the **Filter** box, and press Enter.

Add Variables from a Rule Set, Decision, or Code File

1 On the **Variables** tab, click the **Rule Set Variables** subtab.

2 Click **Add variable**, and select **Rule set**, **Decision**, or **Code file**. The Choose an Item window appears.

3 Select the object from which you want to import variables, and click **OK**. The Add Variables window appears.

4 Select the variables that you want to import and click **+›**. To import all of the variables in the table, click **+»**.

5 Click **Add** to add the selected variables, or select **Add and replace** to replace existing variables that have the same name.

6 On the **Variables** tab, select or clear the **Input** and **Output** check boxes as necessary. See "Input Variables, Output Variables, and Temporary Variables" on page 17 for more information.

Add Global Variables to a Rule Set

In order to use a global variable in a rule set, the global variable must be activated. Instructions for defining and activating global variables are in "Create a New Global Variable" on page 142 and "Activate a Global Variable" on page 149.

Note: You can add a global variable that has not been activated to a rule set and publish the rule set, but the variable's value is set to missing until the global variable is activated

1 On the **Variables** tab, click the **Global Variables** subtab.

2 Click **Select Variables**. The Select Variables window appears.

3 Select the check boxes for the variables that you want to add to the rule set, and click **OK**.

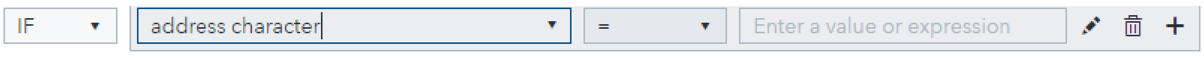
4 (Optional) Select the **Output** check box if you want the value of the variable to be written to the output table that is generated when the rule set is run.

Note: If the rule set will be included in and run as part of a decision, it is recommended that you do not select the **Output** check box for global variables in the rule set. Instead, select the **Output** check box for the decision variable of the same name after you add the rule set to the decision. For more information, see "Using Global Variables" on page 141.

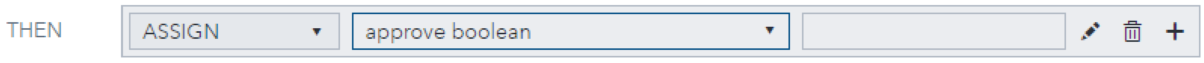
TIP In the Global Variables category view, the **Value** column displays the value of the latest version of the variable. The **Activated Value** column displays the value of the currently active version. A check mark in the **Deleted** column indicates that the variable has been deleted from the list of global variables.

Create a Custom Variable Dynamically

To create a variable in a rule set, you can enter the new variable name in any condition variable field and in the variable field of assignment statements. By default, SAS Intelligent Decisioning creates a variable of type Decimal. To create a variable of a different type, enter the variable name, a space, and then the data type. If you enter the name in a condition variable field, SAS Intelligent Decisioning creates the variable as an input-output variable. If you enter the name in an assignment statement, SAS Intelligent Decisioning creates the variable as an output variable. For example, you can create an input-output variable of type Character that is named **address** by entering it in the condition variable field:



You can create an output-only variable of type Boolean named **approve** by entering it in an assignment statement, followed by a space and the data type:



Create Custom Variables on the Variables Tab

Note: For information about data grid variables, see "Defining Data Grid Variables" in *SAS Intelligent Decisioning: Using Data Grids*.

To create custom variables on the **Variables** tab:

1 Click the **Rule Set Variables** subtab.

2 Click **Add variable** and select **Custom variable**. The Add Variables window appears.

3 Complete these steps for each variable that you want to add:

   a Enter the name of the new variable, and select the data type of the variable. See “The Properties of a Variable” on page 16 for additional information.

   > **TIP** To re-add a variable that was used in a locked version, you must specify the same data type that was used in the previous version.

   b (Optional) Click **Optional** to display the **Description**, **Initial value**, and **Length** fields.

   c (Optional) Enter a length, initial value, and description for the new variable. Whether you can specify an intial value or length for the variable depends on the variable's data type. See “The Properties of a Variable” on page 16 for additional information.

   d Click **Add**. SAS Intelligent Decisioning adds the new variable to the table of variables at the bottom of the window. By default, variables are added to the table as both input and output variables.

   e Verify that the **Input** and **Output** check boxes are selected correctly for each variable.

   - Clear the **Input** check box for any variable that you do not want to be mapped to a column in an input table or for which you do not want to specify a value.
   - Clear the **Output** check box for any variable that you want to exclude from the output data.
   - Clear both the **Input** and **Output** check boxes to create a temporary variable.

4 (Optional) Modify the variable properties in the table of variables.

5 Click **OK** to add the variables and close the Add Variables window.

Duplicate a Variable

1 On the **Variables** tab, click the **Rule Set Variables** subtab.

2 Select the variable that you want to duplicate, click ⋮, and select **Duplicate**. The Duplicate Variable window appears.

3 Enter a new name for the duplicate variable.

4 (Optional) Enter a description for the variable.

5 Click **Duplicate**.

Importing and Exporting Variables

Import Variables

You can import variables from either comma-delimited (CSV) files or from JavaScript Object Notation (JSON) files. These files must conform to formats described in Appendix 2, “Import File Formats,” on page 325. These formats are the same formats that are created when you export variables.

1 Open the rule set into which you want to import variables.

2 On the **Variables** tab, click **Import**, and select either **Comma-delimited (*.csv)** or **JSON (*.json)**. The Import File window appears.

3 Click **Browse** and select the file from which you want to import variables.

4 Specify the encoding of the CSV file.

5 Select **Add variables** to append the variables to the current list of variables, or select **Replace variables** to replace the current list of variables with the imported variables.

> TIP To add new columns to an existing data grid variable, select **Replace variables**.

Note: If you replace the list of variables in a rule set and a variable that is used in a locked version of the rule set is not included in the import file, you cannot save the rule set.

6 Click **Import**, and then click 🖫 to import the variables and save the rule set.

Export Variables

1 Open the rule set from which you want to export variables.

2 On the **Rule Set Variables** tab, select the variables that you want to export.

3 Click **Export**, and select the file type to which you want to export the variables. The result depends on which browser you are using and the browser’s default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Delete Variables

Note: You cannot delete a variable if it is used in the current version of an object. You can delete variables from the current version if they are used only in locked versions, but those variables are still included in the locked versions.

1 On the **Variables** tab, click the **Rule Set Variables** subtab.

2 Select the check box for the variables that you want to delete, click ⋮, and select **Delete**.

Edit Variable Properties

On the **Variables** tab, click the **Rule Set Variables** subtab, and click on the name of the variable that you want to edit. The Edit Variable window appears. Edit the properties as needed, and then click **OK**.

Note: When you rename a variable, references to that variable within the same object change to use the new name. You cannot change the name or data type of a variable that is used in locked versions.

See “The Properties of a Variable” on page 16 for additional information.

Edit Metadata for Data Grid Variables

For information, see “Editing Data Grid Variable Metadata” in *SAS Intelligent Decisioning: Using Data Grids*.

Determine Which Objects Use a Particular Variable

On either the **Rule Set Variables** subtab or the **Global Variables** subtab, select the check box for the variable, click ⋮, and select **View used by report**.

In the report, you can use the **Filter** field to filter the list of objects based on the object names. If the variable that was selected in Step 1 above is a data grid variable, and if you are interested only in a specific column within the data grid, you can select the column in the **Column name** field, and then click **Apply**. SAS Intelligent Decisioning narrows the search results to only the objects that use the data grid with the selected column.

Click on an object name to open the object. Click ⓘ next to an object name to display the date on which the object was last modified and the ID of the user who modified the object. For decisions and code files that have been checked out, SAS Intelligent Decisioning also displays the IDs of the users that have checked out the objects. For decisions, SAS Intelligent Decisioning includes the decision's workflow status.

Click ⊡ next to nested decision names to display the list of objects within the nested decision that use the selected variable. Click ▷ and ◢ to open and close the list objects within the nested decision.

Click **Export** to export the Used By report to a PDF file. In the PDF file, you can click on an object name to open the object in a new browser tab.

Defining New Rules in a Rule Set

Add a Stand-Alone Assignment Statement

Stand-alone assignment statements always execute unless a RETURN action stops the execution of the rule set before execution reaches the assignment statement. Rule-fired data is not generated for standalone assignment statements.

1 On the **Rule Set** tab, click **Add Assignment** if the rule set is empty or, if the rule set contains at least one statement, select **Add** ⇨ **Add assignment**. The application adds an assignment statement to the top of the rule set, below any existing assignment statements.

> **TIP** If you do not have permission to update an object, SAS Intelligent Decisioning displays `(Read-Only)` in the title bar next to the object name, and the buttons for modifying and saving the object are disabled.

2 Import or create any variables that are required for the assignment statement that have not already been added to the rule set. You can add or create the variables on the **Variables** tab, or you can define variables dynamically as you author the statement. See "Managing the Variables in a Rule Set" on page 16 for more information.

3 Select the variable to which you want to assign a value.

4 Enter the expression for the variable in the expression field. See "About Defining Expressions" on page 33 for additional information.

5 (Optional) Move the assignment statement to a different position in the rule set. To move the statement, click ↑ or ↓.

6 Click 🖫 to save the rule set. SAS Intelligent Decisioning validates the syntax of the expressions. If it does not detect any problems, it saves the rule set.

Add an IF-THEN or IF-THEN-ELSE Rule to an Assignment or Common Rule Set

1 Create or open the rule set. If no variables are defined in the rule set, SAS Intelligent Decisioning displays the **Variables** tab. Otherwise, it displays the **Rule Set** tab.

> **TIP** If you do not have permission to update an object, SAS Intelligent Decisioning displays `(Read-Only)` in the title bar next to the object name, and the buttons for modifying and saving the object are disabled.

2 Import or create any variables that are required for the new rule that have not already been added to the rule set. You can add or create the variables on the **Variables** tab, or you can define variables dynamically as you author the rule. See "Managing the Variables in a Rule Set" on page 16 for more information.

3 Click the **Rule Set** tab.

4 Click **Add Rule** if the rule set is empty. If the rule set contains at least one statement, click **+Add Rule** or select **Add** ⇨ **Add rule**. SAS Intelligent Decisioning adds a new IF-THEN rule to the rule set.

5 (Optional) Define the condition expression for the rule. See "About Defining Expressions" on page 33 for additional information.

To add additional condition expressions to the selected rule, select **Add** ⇨ **Condition**.

6 Define the action expressions for the rule. See "About Defining Expressions" on page 33 for additional information.

To add additional action expressions to the selected rule, select **Add** ⇨ **Action**.

> **TIP** To move condition or action expressions up or down within an IF, THEN, or ELSE clause, select the expression and click ↑ or ↓.

7 (Optional) Change the rule operator to **ELSE** or **OR**. If the rule is the first rule in a rule set, the rule operator must be IF.

When you change the operator on a rule from IF to ELSE, the condition expression is preserved, and the rule becomes an ELSE clause with a condition. If you change IF or ELSE to OR, the condition expression is preserved, but the action expression is deleted. For more information, see "Controlling Which Conditions Are Evaluated" on page 31.

8 (Optional) Change the operator on the THEN clause from **ASSIGN** to **RETURN**. The RETURN action stops the execution of any additional statements in the rule set. See "Controlling Which Conditions Are Evaluated" on page 31 for more information.

9 (Optional) Select **Add** ⇨ **ELSE rule** to add an ELSE clause to the currently selected rule. The ELSE clause does not have a condition, but you can add one by selecting **Add** ⇨ **Condition**.

10 (Optional) Define the condition and action expressions for the ELSE clause.

11 (Optional) Change the order of the rules. Rules are evaluated sequentially. To move a rule up or down within a rule set, select the rule and click ↑ or ↓.

12 (Optional) Change the name of the rule. Rule names are limited to 100 characters and must be unique within a rule set. For instructions, see “Rename a Rule” on page 42.

TIP Assigning logical names to the rules makes it easier to determine which rules fired when you review rule-fired data.

13 (Optional) Clear the **Record rule-fired data** check box if you do not want a rule-fired record to be written each time this rule fires. See “How Rules Are Evaluated and When Rule-Fired Records Are Generated” on page 32 for more information.

14 Click 🖫 to save the rule set. SAS Intelligent Decisioning validates the syntax of the expressions. If it does not detect any problems, it saves the rule set.

Adding a Common Rule Set Reference to an Assignment Rule Set

About Rule Set Variables and Mapping

Common rule sets and assignment rule sets each define their own variables. When you add a common rule set to an assignment rule set, you must map the variables that are defined in the common rule set to variables that are used in the assignment rule set. By default, if the assignment rule set defines variables of the same name and data type as the variables in the common rule set, then SAS Intelligent Decisioning automatically maps the common rule set variables to the assignment rule set variables. If variables of the same name and data type do not exist in the assignment rule set, you must do one of the following:

- create variables in the assignment rule set that have the same name and data type as the variables in the common rule set. You can use the **Add missing variables** option to create these assignment rule set variables and automatically map the variables. This option is available in both the Add Common Rule Set References wizard and in the Edit Common Rule Set Reference window.
- manually map the common rule set variables to assignment rule set variables that have different names. If the assignment rule set variables do not already exist when you add the common rule set reference in the wizard, you can edit the properties of the reference and map the variables in the Edit Common Rule Set Reference window.

Add a Common Rule Set Reference to an Assignment Rule Set

1 Open the assignment rule set. If no variables are defined in the rule set, SAS Intelligent Decisioning displays the **Variables** tab. Otherwise, it displays the **Rule Set** tab.

> TIP If you do not have permission to update an object, SAS Intelligent Decisioning displays `(Read-Only)` in the title bar next to the object name, and the buttons for modifying and saving the object are disabled.

2 Click the **Rule Set** tab.

3 Do one of the following:

- If the rule set contains at least one rule, click **Add Rule** and select **Add reference**.
- If the rule set is empty, click **Add Rule**. In the Create New Rule window, select **Common rule set reference**, and click **OK**.

SAS Intelligent Decisioning opens the Add Common Rule Set References wizard.

4 On the Select Common Rule Sets page, select the check boxes for each common rule set that you want to add to the assignment rule set, and click **Next**.

5 On the **Select Common Rule Set Versions** page, select the version of each of the common rule sets that you want to use in the assignment rule set, then click **Next**.

> TIP To specify the latest version of the common rule set, you can select the version number of the latest version, or you can select **Use latest**. If you select **Use latest**, SAS Intelligent Decisioning always uses the most recently created version regardless of the version number.

6 (Optional) Add any global variables that are used by the common rule set to the assignment rule set. For instructions, see "Add Global Variables to a Rule Set" on page 19.

7 (Optional) On the **Map Inputs** page, for each common rule set variable in the **Input Variable** column, select the assignment rule set variable that you want to map it to in the **Maps To** column. If the assignment rule set contains variables with the same name and data type as the variables in the common rule set, SAS Intelligent Decisioning automatically maps the common rule set variables to the assignment rule set variables.

> **TIP** To create assignment rule set variables that have the same name and data type as the variables in the common rule set, click **Add Missing Variables**. If you want to create new variables with different names in the assignment rule set, you can ignore the error markers, and continue with the next step. After you click **Finish** in Step 12, you can create new variables in the assignment rule set, then edit the properties of the common rule set reference to change the variable mappings. For more information, see "Edit the Properties of a Common Rule Set Reference" on page 29.

For additional information, see "About Rule Set Variables and Mapping" on page 26.

8 Click **Next**.

9 (Optional) On the **Map Outputs** page, for each common rule set variable in the **Output Variable** column, select the assignment rule set variable that you want to map it to in the **Maps To** column. If the assignment rule set contains variables with the same name and data type as the variables in the common rule set, then SAS Intelligent Decisioning automatically maps the common rule set variables to the assignment rule set variables.

> **TIP** To create assignment rule set variables that have the same name and data type as the variables in the common rule set, click **Add Missing Variables**. If you want to create new variables with different names in the assignment rule set, you can ignore the error markers, and continue with the next step. After you click **Finish** in Step 12, you can create new variables in the assignment rule set, then edit the properties of the common rule set reference to change the variable mappings. For more information, see "Edit the Properties of a Common Rule Set Reference" on page 29.

For more information, see "About Rule Set Variables and Mapping" on page 26.

10 Click **Next**.

11 (Optional) On the **Order Rules**, page, change the order in which the common rule sets are included in the assignment rule set. To move a common rule set up or down within the assignment rule set, select the common rule set and click ↑ or ↓.

12 Click **Finish**. SAS Intelligent Decisioning adds the common rule set reference to your assignment rule set. You can click > to display the content of the common rule set inline in the assignment rule set, but the content is read-only. You can use the rule-specific actions menu to edit the reference or to open the common rule set.

Edit the Properties of a Common Rule Set Reference

1 Open the assignment rule set, click ⋮ for the reference that you want to edit, and select **Edit reference**. The Edit Common Rule Set Reference window appears.

2 (Optional) On the **Common rule set** tab, select a different common rule set or change the version of the common rule set.

3 (Optional) On the **Map Inputs** tab, change the variable mappings for the input variables in the common rule set. For more information, see Step 7 of “Add a Common Rule Set Reference to an Assignment Rule Set”.

4 (Optional) On the **Map Outputs** tab, change the variable mappings for the output variables in the common rule set. For more information, see Step 9 of “Add a Common Rule Set Reference to an Assignment Rule Set”.

5 Click **OK** to close the Edit Common Rule Set References window and return to the assignment rule set.

6 Click 🖫 to save your changes to the assignment rule set.

Add a New Rule in a Filtering Rule Set

Note: When SAS Intelligent Decisioning generates code for a filtering rule set, it joins the rules together with the OR operand.

1 Create or open the filtering rule set. If no variables are defined in the rule set, SAS Intelligent Decisioning displays the **Variables** tab. Otherwise, it displays the **Rule Set** tab.

TIP If you do not have permission to update an object, SAS Intelligent Decisioning displays `(Read-Only)` in the title bar next to the object name, and the buttons for modifying and saving the object are disabled.

2 Import or create any variables that are required for the new rule that have not already been added to the rule set. You can add or create the variables on the **Variables** tab, or you can define variables dynamically as you author the rule. See “Managing the Variables in a Rule Set” on page 16 for more information.

3 Click the **Rule Set** tab.

4 Click **Add Rule** if the rule set is empty or, if the rule set contains at least one statement, select **Add** ⇨ **Add rule** or **+Add Rule**. SAS Intelligent Decisioning adds a new IF rule to the end of the rule set.

5 Define the condition expression for the rule. See “About Defining Expressions” on page 33 for additional information.

To add additional condition expressions to the selected rule, select **Add** ⇨ **Condition**.

> **TIP** To move condition expressions up or down within an IF clause, select the expression and click ↑ or ↓.

6 (Optional) Change the order of the rules. Rules are evaluated sequentially. To move a rule up or down within a rule set, select the rule and click ↑ or ↓.

7 (Optional) Change the name of the rule. Rule names are limited to 100 characters and must be unique within a rule set. For instructions, see “Rename a Rule” on page 42.

> **TIP** Assigning logical names to the rules makes it easier to determine which rules fired when you review rule-fired data.

8 (Optional) Clear the **Record rule-fired data** check box if you do not want a rule-fired record to be written each time this rule fires. See “How Rules Are Evaluated and When Rule-Fired Records Are Generated” on page 32 for more information.

9 Click 🖫 to save the rule set. SAS Intelligent Decisioning validates the syntax of the expressions. If it does not detect any problems, it saves the rule set.

Copy Rules From Another Rule Set

The menu option **Copy from rule set** appears in the **Add** menu for a rule set. The rule set must contain at least one rule in order for this menu option to appear. Adding a rule from another rule set does not replace any of the rules in the rule set that you are editing.

1 Open the rule set into which you want to add the rule, and click the **Rule Set** tab.

2 If the rule set does not already contain a rule, click **Add rule**.

3 Select the existing rule after which you want to add IF rules or to which you want to add an ELSE rule. If you do not select a rule, then SAS Intelligent Decisioning adds copied rules to the end of the rule set.

4 Click **Add** and select **Copy from rule set**. The **Copy Rules** window appears. This window displays the latest version of every rule in every rule set that you have permission to view.

5 Select the rules that you want to copy, and click **Copy**.

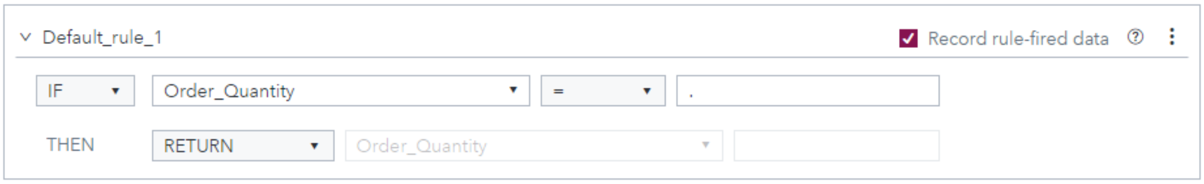
SAS Intelligent Decisioning adds the selected rules to the rule set together with any new variables that these rules need.

Note: If the copied rule contains functions or complex expressions, the variables might not be added to the rule set. Add any variables used in the expression to the rule set, and validate the rule set.

Controlling Which Conditions Are Evaluated

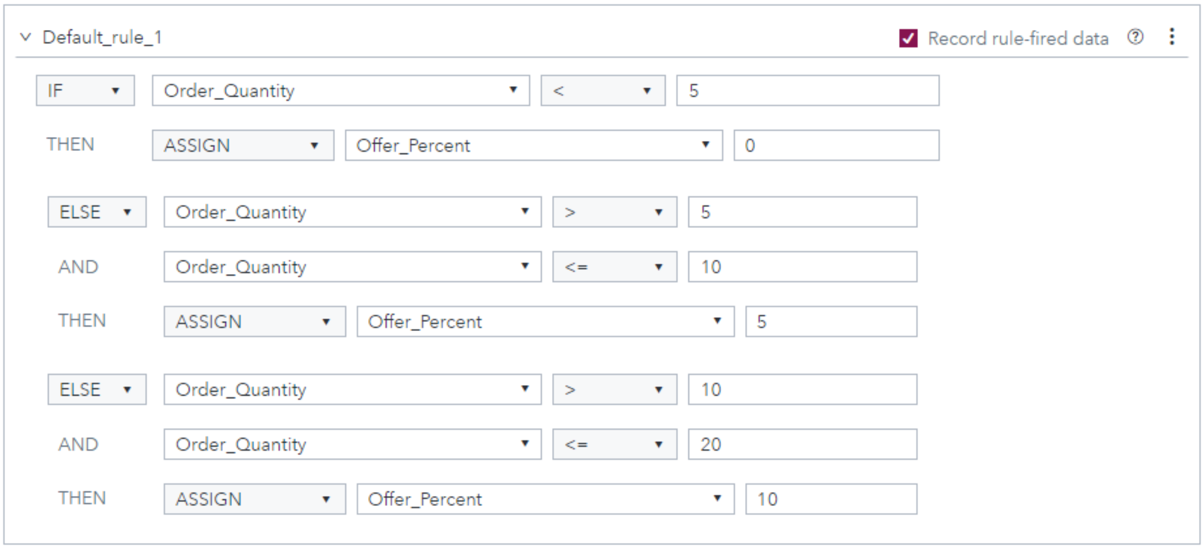
By default, rules are assigned the IF rule operator, which means that the rule's conditions are evaluated regardless of the results of previous rules. In assignment rule sets, you can control whether condition expressions are evaluated by using the RETURN action, the ELSE operator, or the OR operator.

The RETURN action stops the execution of any remaining rules in a rule set. If you are executing a single rule set, execution ends. If you are executing a decision, control moves to the next object in the decision. For example, the rule in the following figure stops the execution of any remaining rules in the rule set if the value of the Order_Quantity variable is missing.



IF	Order_Quantity	=	.
THEN	RETURN	Order_Quantity	

If you set a clause's operator to ELSE, then the clause's conditions are evaluated only if the previous clause's conditions evaluated to false. For example, given the rule set shown in the following figure, if Order_Quanitity is 9, the condition for the IF clause evaluates to false, and the condition for the first ELSE clause evaluates to true. Therefore, the action for first ELSE clause is executed, and the condition for the last ELSE clause is not evaluated. The value of Offer_Percent is set to 5.



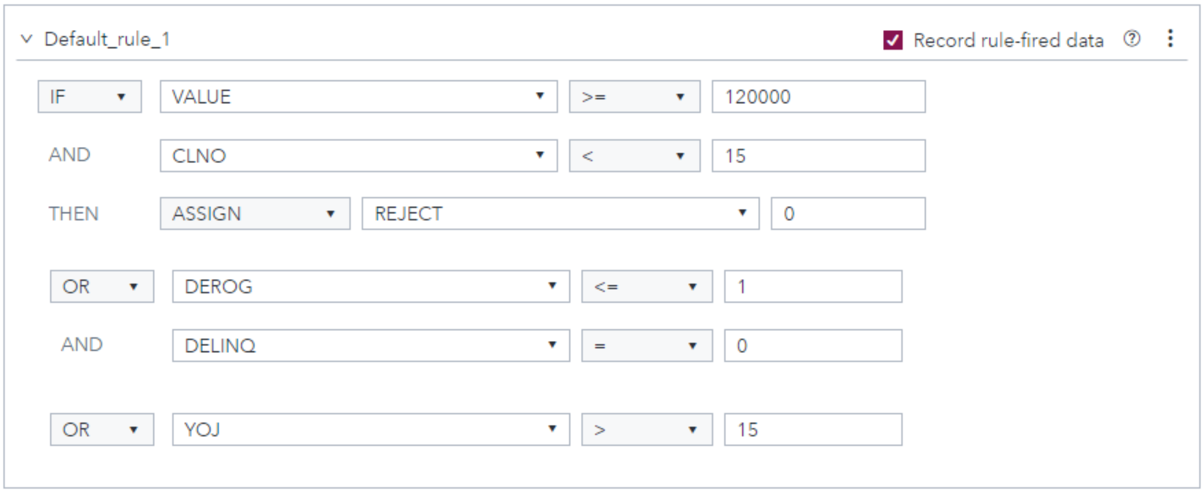
IF	Order_Quantity	<	5
THEN	ASSIGN	Offer_Percent	0
ELSE	Order_Quantity	>	5
AND	Order_Quantity	<=	10
THEN	ASSIGN	Offer_Percent	5
ELSE	Order_Quantity	>	10
AND	Order_Quantity	<=	20
THEN	ASSIGN	Offer_Percent	10

Use the OR operator to execute the same action expression for each of the several conditions. If you assign the OR operator to a rule, then you cannot enter an action expression for the rule. If any of the OR conditions evaluate to true, SAS Intelligent

Decisioning executes the action of the last rule that was assigned the IF or ELSE operator. If several consecutive rules that are all assigned the OR operator, the action is executed as soon as a set of conditions evaluates to true. The conditions for the remaining consecutive OR rules are not evaluated.

For example, given the rules in the following figure, REJECT is assigned the value of 0 (False) if any of the following conditions are true:

- `(VALUE >= 120000 AND CLNO < 15)`
- `(DEROG <= 1 AND DELINQ = 0)`
- `YOJ > 15`



Default_rule_1 — Record rule-fired data ✓

IF	VALUE	>=	120000
AND	CLNO	<	15
THEN	ASSIGN	REJECT	0
OR	DEROG	<=	1
AND	DELINQ	=	0
OR	YOJ	>	15

How Rules Are Evaluated and When Rule-Fired Records Are Generated

By default, the condition expressions for all rules in a rule set are evaluated sequentially regardless of the results of previous rules. However, in assignment rule sets, you can use the ELSE operator, the OR operator, or the RETURN action to control whether condition expressions are evaluated. See "Controlling Which Conditions Are Evaluated" on page 31 for more information.

If a rule's condition expressions evaluate to true, the rule is said to have *fired*. In assignment rule sets, SAS Intelligent Decisioning executes the rule's action expressions.

By default, every time a rule fires, it generates a rule-fired record. You can control when rule-fired records are generated by using the **Record rule-fired data** check boxes. See Step 13 in "Add an IF-THEN or IF-THEN-ELSE Rule to an Assignment or Common Rule Set" on page 25.

Note: Stand-alone assignment statements always execute unless a RETURN action stops the execution of the rule set before execution reaches the assignment statement. Rule-fired data is not generated for standalone assignment statements.

For additional information, see "Run a Rule-Fired Analysis" on page 70.

Defining Expressions in Rules and Assignment Statements

About Defining Expressions

Expressions can be up to 1024 characters long. They can contain numeric constants, character strings, variables, operators, SAS DS2 functions, data grid functions, and the SAS Intelligent Decisioning LOOKUP and LOOKUPVALUE functions. You can enter expressions directly into the expression fields, or you can use the Expression Editor to create and edit expressions.

TIP Use caution when you test for equality by using scientific notation. Two numbers that appear to be the same might evaluate to different numbers because of the precision involved in scientific notation.

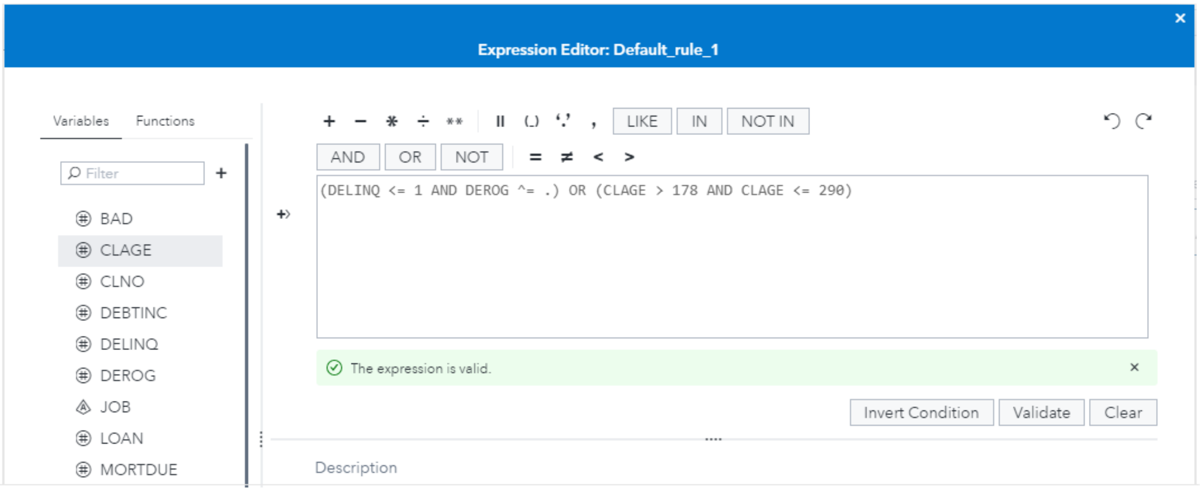
For more information about entering expressions, see the following topics:

- "Using the Expression Editor" on page 33
- "Enter LOOKUP and LOOKUPVALUE Expressions" on page 35
- "Entering Literal Data Values" on page 36
- "Operators for Use in Expressions" on page 37
- "Using the LIKE Operator" on page 38
- "Using Functions in Expressions" on page 40
- "Working with Missing Values" on page 40
- "Data Grid Functions" in *SAS Intelligent Decisioning: Using Data Grids*

Using the Expression Editor

You can use the Expression Editor to enter expressions that do not use the LOOKUP or LOOKUPVALUE functions. You must use the Expression Editor to enter expressions that use the concatenation (||) operator or the exponent operator (**).

To open the Expression Editor, click ✎ for the expression that you want to edit.



```
(DELINQ <= 1 AND DEROG ^= .) OR (CLAGE > 178 AND CLAGE <= 290)
```

The expression is valid.

You can enter expressions directly into the expression field, or you can use the lists of operators, function names, and variable names to add them to the expression.

- To add a variable, click the **Variables** tab, and double-click the variable name.
- To add a function call, click the **Functions** tab, expand the appropriate function category, select a function name, and click +›. The Expression Editor adds a basic syntax template for the function to the expression field.

  The icon beside each function name indicates the return type of the function. Functions that return character data are displayed with the ◬ icon, and functions that return numeric data are displayed with the ⊕ icon.
- To invert an entire condition expression, click **Invert Condition**. SAS Intelligent Decisioning wraps the entire expression in parentheses and adds the keyword `NOT`:

  `NOT(`*`condition`*`)`

  TIP The **Invert Condition** button is a toggle. If the entire condition is already enclosed in `NOT()`, clicking this button removes the keyword NOT and the parentheses surrounding the expression. This button does not affect NOT conditions that are embedded within the expression. (This button is available only if you are editing a condition expression in a rule set. It is not available for action expressions or for variable initial values.)
- Click **Validate** at any time to check the syntax of the expression that you are building.

  Note: If you invert the condition and click **Validate**, the validation process might return incorrect results.
- To clear the expression field, click **Clear**.
- To save the expression, click **Save**.

  IMPORTANT After you save a rule set expression in the Expression Editor, the expression field on the **Rule Set** tab becomes a read-only field. You can edit it only by launching the Expression Editor again.

TIP By default, the autocomplete feature is turned on for SAS functions and custom functions. You can turn off this feature or change what is displayed in the autocomplete list by using the SAS Intelligent Decisioning settings. You can specify that SAS Intelligent Decisioning also displays variable names. For information, see "SAS Intelligent Decisioning Settings" on page 8.

In the autocmplete list, SAS Intelligent Decisioning displays all function names and variable names that match the text that you enter. Double-click an item to add it to the code field. Single-click an item to display information about the item, such as whether it is a function or a variable. For functions, the information includes the returned data type and category. For variables, the information includes the data type and direction (input, output, or both).

The autocomplete list does not include functions that are defined in a custom context file.

TIP For rule set expressions, when SAS Intelligent Decisioning generates code for the rule set, it adds a semicolon (;) to the end of each line that you enter in the expression editor, except for lines that end with an opening parenthesis, a closing parenthesis, a comma (,), or a semicolon (;).

Enter LOOKUP and LOOKUPVALUE Expressions

Note: You can enter the LOOKUP function only in condition expressions, and you can enter the LOOKUPVALUE function only in action expressions. The LOOKUP function determines whether a lookup key exists in the lookup table, and the LOOKUPVALUE function retrieves the value associated with a key. It is best practice to use the LOOKUP function to verify that a key exists before you try to retrieve the value associated with the key.

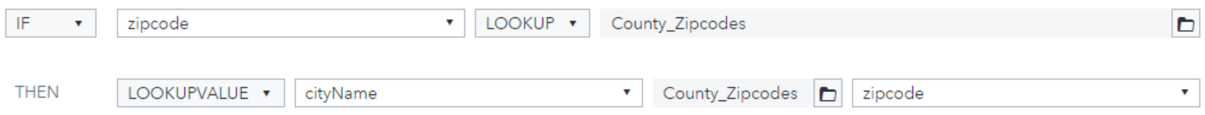
To enter a condition expression that uses the LOOKUP function, select **LOOKUP** as the rule operator, and select the lookup table in the expression field.

To enter an action expression that uses the LOOKUPVALUE function, complete these steps:

1. Select **LOOKUPVALUE** as the operator on the THEN clause.
2. In the second field of the THEN clause, select the variable to which you want to assign the value that is retrieved from the lookup table.
3. Click 🗀, and select the lookup table from which you want to retrieve the lookup value.
4. In the last field, select the variable whose value matches the value of the lookup key, or select the actual value of the lookup key. To select the value of the lookup key:
   a. Select **Select a lookup key**. SAS Intelligent Decisioning displays the list of lookup entries in the specified table.

b Select the record whose key you want to use in the expression, and click **Save**.

For example, suppose you have a lookup table in which the lookup keys are ZIP codes and the lookup values are city names. You could use the following rule to test whether the lookup table contains an entry for the ZIP code, and, if the entry exists, retrieve the city name that is associated with the ZIP code.



IF zipcode LOOKUP County_Zipcodes

THEN LOOKUPVALUE cityName County_Zipcodes zipcode

Entering Literal Data Values

Depending on whether you use the Expression Editor or enter expressions directly into the expression fields, you must enter some values differently.

Data Type	How To Enter Values	Example
Character	Enclose character strings in quotation marks. If the character string contains embedded quotation marks, use different quotation marks for the embedded and enclosing quotation marks.	`'Gold Account'` `"d'oscail"` `'d"oscail'`
Data grid	Enclose data grid JSON strings in single quotation marks. **Note:** The only time you might need to enter JSON strings manually is when you are using the DATAGRID_CREATE function.	See *SAS Intelligent Decisioning: Using Data Grids* for an explanation of the JSON syntax and a description of the DATAGRID_CREATE function.
Date	In the rule set editor, enter Date values by using the format DDMMMYYYY. Enclose each value in single quotation marks followed by `d`.	`'01AUG2017'd`
	In the Expression Editor, use the DS2 function TO_DOUBLE and specify the DATE data type in order to cast the Date value so that it can be compared correctly to other variables. See *SAS DS2 Programmer's Guide* for information about date, time, and timestamp values, and see *SAS DS2 Language Reference* for information about the TO_DOUBLE function.	`to_double(date '2017-11-04')`

Data Type	How To Enter Values	Example
Datetime	In the rule set editor, enter Datetime values by using the format DDMMMYYYY:HH:MM:SS. Use 24-hour clock notation. Enclose each value in single quotation marks followed by `dt`.	`'31AUG2017:15:00:00'dt`
	In the Expression Editor, use the DS2 function TO_DOUBLE and specify the TIMESTAMP data type in order to cast the Datetime value so that it can be compared correctly to other variables. See *SAS DS2 Programmer's Guide* for information about date, time, and timestamp values, and see *SAS DS2 Language Reference* for information about the TO_DOUBLE function.	`to_double(timestamp '2017-11-04 10:54:34.012')`
Boolean	In the rule set editor, Boolean values are not enclosed in quotation marks. Enter only the values.	`True` `False`
	In the Expression Editor, use numeric values to indicate True or False. Specify 1 for True and 0 for False.	`1` `0`

Operators for Use in Expressions

The following table lists the operators that you can use in an expression. Do not enter a space between the elements of the operators <=, >=, or ^=. Some mnemonic equivalents for these operators cannot be used in SAS Intelligent Decisioning expressions. See *SAS DS2 Programmer's Guide* for more information about specifying operators in expressions.

Table 2.1 *Operators for Use in Expressions*

Operator	Definition	Example
*	Multiply	`0.085 * sales`
/	Divide	`amount / 5`
+	Add	`num + 3`
–	Subtract	`sale - discount`

Operator	Definition	Example
**	Raises the first operand to the power of the second operand	`num1**num2`
=	Equal to	`tries = maxTriesAllowed`
||	Concatenates the first string and the second string	`string1 || string2`
!=	Not equal to	`insufficientFunds != 1`
^= ≠ (Expression Editor interface)	Not equal to	`balance ^= 'low'`
>	Greater than	`daysLate > 5`
>=	Greater than or equal to	`balance >= 1000`
<=	Less than or equal to	`balance <= 250`
<>	The maximum of the left and right operands	`num1 <> num2`
IN (*value-list*)	Equal to an item in *value-list*	`risk in ('high','medium','low')`
NOT IN (*value-list*)	Not equal to an item in *value-list*	`offerPercent not in (10,20,30)`
LIKE '*pattern*'	If the variable's value matches the expression pattern in *pattern*, the result is true.	`like 'HS%PP'`
expression AND *expression*	If both expressions are true, the result is true.	`dateExpired >= '01AUG2015'd AND dateExpired <= '31AUG2015'd`
expression OR *expression*	If either expression is true, the result is true.	`dateEnrolled >= '01JAN2015' OR member = 1`

Using the LIKE Operator

The LIKE operator determines whether the value of a variable matches a pattern-matching expression. An expression that uses the LIKE operator has the following syntax:

LIKE '*pattern-matching-expression*'

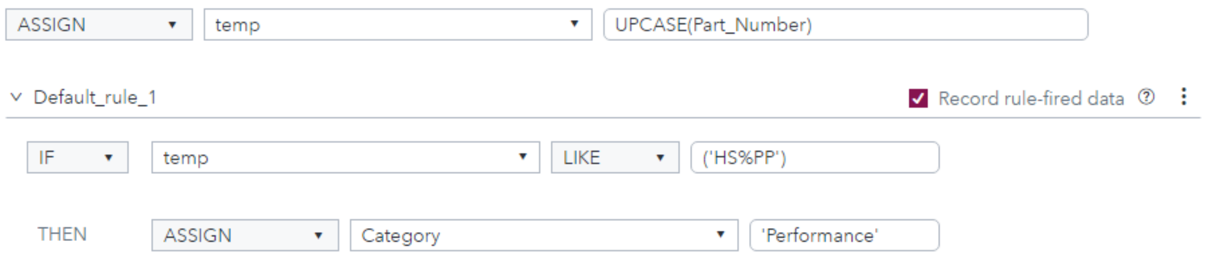
If a variable's value matches the pattern that is specified by *pattern-matching-expression*, the expression evaluates to true (1). Otherwise, the expression evaluates to false (0).

There are three classes of pattern-matching characters.

Table 2.2 Pattern-Matching Characters

Character	Description
underscore (_)	Matches any single character
percent sign (%)	Matches any sequence of zero or more characters **Note:** Be aware of the effect of trailing blanks. To match values, you might have to use the TRIM function to remove trailing blanks.
any other character	Matches that character

The LIKE expression is case sensitive. To search for mixed-case strings, use the UPCASE function to create an uppercase version of the variable that you want to search. You can use a temporary variable to store the results of the UPCASE function. Use the LIKE operator to search the uppercase version of the variable. For example, you can search the variable Part_Number for mixed-case strings that begin with HS and end with PP by using the two rules shown in the following figure.



ASSIGN temp UPCASE(Part_Number)

Default_rule_1 — Record rule-fired data

IF temp LIKE ('HS%PP')

THEN ASSIGN Category 'Performance'

The following table shows examples of the matches that result if you search a variable that could have these values: Smith, Smooth, Smothers, Smart, Smuggle.

Table 2.3 Examples of LIKE Expressions

LIKE Expression Example	Matching Results
`like 'Sm%'`	`Smith, Smooth, Smothers, Smart, Smuggle`
`like '%th'`	`Smith, Smooth`
`like 'S__gg%'`	`Smuggle`
`like 'S_o'`	(no matches)
`like 'S_o%'`	`Smooth, Smothers`

LIKE Expression Example	Matching Results
`like 'S%th'`	`Smith, Smooth`

Using Functions in Expressions

SAS Intelligent Decisioning supports the following functions in expressions:

- LOOKUP and LOOKUPVALUE functions. Condition expressions can contain the LOOKUP function, and action expressions can contain the LOOKUPVALUE function. However, if the expression contains the LOOKUP or LOOKUPVALUE function, then the expression cannot contain anything else. For more information, see "Enter LOOKUP and LOOKUPVALUE Expressions" on page 35, "LOOKUP Function" on page 137, and "LOOKUPVALUE Function" on page 138.
- SAS DS2 functions. For additional information about these functions and additional DS2 functions, see *SAS DS2 Language Reference*.
- SAS Data Quality Server functions. For information about these functions, see "Functions and CALL Routines" in *SAS Data Quality and SAS Data Quality Server: Language Reference*. These functions appear in the expression editor only if SAS Data Quality Server is installed.

  Note: You cannot publish rule sets that use SAS Data Quality functions to container destinations. Also, rule sets that use these functions fail when they are run inside the database.
- Data grid functions. For more information about these functions, see "Data Grid Functions" in *SAS Intelligent Decisioning: Using Data Grids*.
- Custom DS2 functions that you define in SAS Intelligent Decisioning or by using the sid-functions CLI. For more information, see Chapter 7, "Using Custom Functions," on page 197 and "sid-functions Plug-In" in *SAS Intelligent Decisioning: Command-Line Interfaces*.

To enter a function in an expression, use the Expression Editor. See "Using the Expression Editor" on page 33 for more information.

Working with Missing Values

You can use the MISSING function to check for missing numeric and character values. This function returns 0 (false) or 1 (true). Missing values have a value of **`false`** when you use them with logical operators such as AND or OR. You can use the MISSING function to eliminate errors, notes, and warnings in the SAS log that are caused by missing values.

In expressions, you can use the period (.) to denote missing numeric values, and two single quotation marks with no space (the empty string `''`) to denote missing character values.

For more information, see "How DS2 Processes Nulls and SAS Missing Values" in *SAS DS2 Programmer's Guide*.

TIP To specify an empty data grid that does not contain data or column metadata, use empty brackets ([]). An empty data grid is not considered missing. You cannot use the MISSING function to determine whether a data grid is missing or empty.

Delete Expressions or ELSE Rules

To delete a condition or action expression, click 🗑 for that expression.

To delete an entire ELSE rule, select the rule, right-click on the rule, and select **Delete the selected object**.

Managing Rules

Duplicate a Rule

1 Click ⋮ for the rule, and select **Duplicate rule**. The Duplicate Rule window appears.

2 Enter a name for the duplicate rule if you do not want to use the default name.

3 If you are duplicating a rule that contains an ELSE or OR clause, specify whether you want to duplicate only the IF clause or duplicate the entire rule, and click **Duplicate**.

Delete a Rule

1 Click ⋮ for the rule, and select **Delete rule**.

2 (Optional) If the rule contains an ELSE or OR clause, SAS Intelligent Decisioning prompts whether you want to delete only the IF clause or to delete the entire rule. Click **Delete All** to delete everything, or click **Delete IF** to delete only the IF clause. If you select **Delete IF**, then the ELSE or OR clause becomes an assignment statement if it does not have any condition expressions. If it has a condition expression, it becomes the new IF rule.

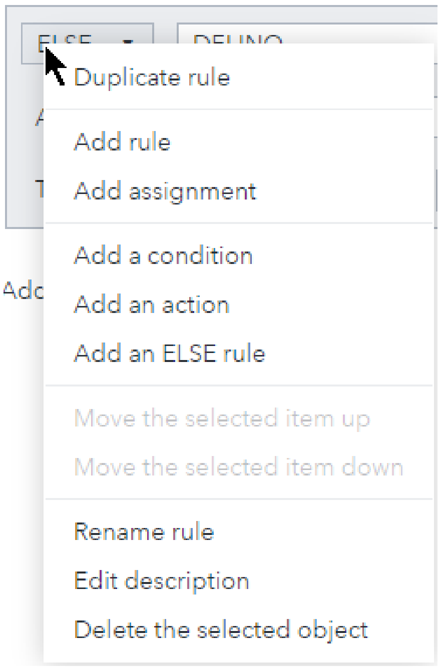
TIP To delete only the ELSE or OR clause, click 🗑 for that clause.

Rename a Rule

To rename the IF clause of a rule, select the existing rule name, and enter a new name.

To rename the ELSE or OR clause of a rule:

1 Right-click on the **ELSE** or **OR** operator and select **Rename rule**.



2 Enter the new name and click **Rename**.

TIP Rule names for ELSE and OR clauses do not appear in the rule set editor.

Edit a Rule Description

1 Click ⋮ for an IF rule, or right-click an ELSE or OR rule to display the menu.

2 Select **Edit description** from the menu. The Edit Description window appears.

3 Enter the new description, and click **Edit**.

Reorder Rules

To move a rule up or down within an IF clause, an ELSE clause, or an OR clause select the rule, and click ↑ or ↓.

Copy a Rule Set URL

To create a link for external documentation that automatically opens a rule set in SAS Intelligent Decisioning:

1. Open the rule set.
2. Click ⋮, and select **Copy rule set URL**. The Copy URL window appears, and the URL is automatically selected.
3. Click **Copy**, and then click **Close**.

   Paste the link into your documentation.

Compare Rule Set Content

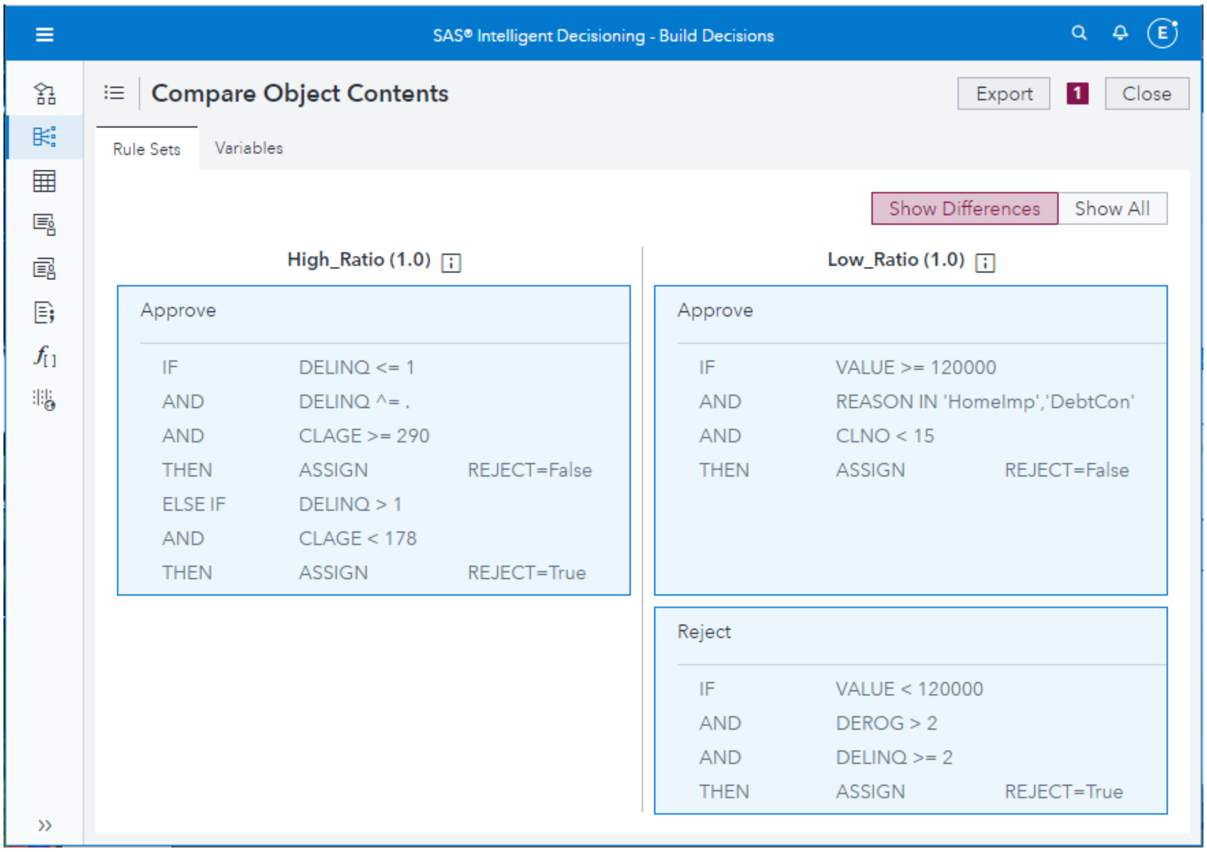
You can compare the contents of two different rule sets, or you can compare the contents of two different versions of the same rule set.

1. Select the objects that you want to compare.
   - To compare the contents of two different rule sets, select the rule sets in the category view, click ⋮, and select **Compare object contents**.
   - To compare the contents of two versions of the same rule set, open the rule set, click ⋮ on the **Versions** tab, and select **Compare object contents**.

   The Select Versions window appears.
2. Select the versions that you want to compare, and click **Compare**. SAS Intelligent Decisioning opens the two objects side-by-side in the Compare Object Contents window.

   By default, the **Rule Sets** tab displays a simplified version of the rules that are different in each object. To display a simplified version of all of the rules in each object, click **Show All**.

   TIP Click ⓘ beside a rule set name to display its location.



High_Ratio (1.0)

Approve

IF	DELINQ <= 1	
AND	DELINQ ^= .	
AND	CLAGE >= 290	
THEN	ASSIGN	REJECT=False
ELSE IF	DELINQ > 1	
AND	CLAGE < 178	
THEN	ASSIGN	REJECT=True

Low_Ratio (1.0)

Approve

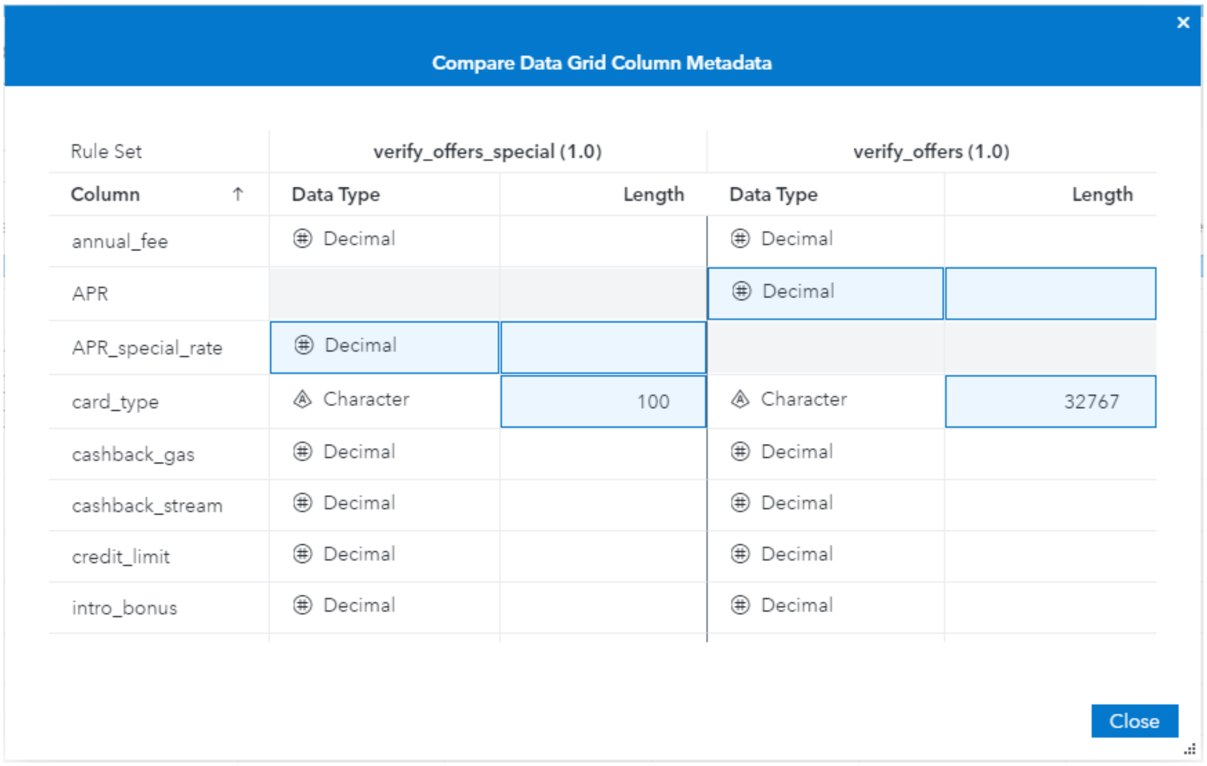
IF	VALUE >= 120000	
AND	REASON IN 'HomeImp','DebtCon'	
AND	CLNO < 15	
THEN	ASSIGN	REJECT=False

Reject

IF	VALUE < 120000	
AND	DEROG > 2	
AND	DELINQ >= 2	
THEN	ASSIGN	REJECT=True

3 (Optional) Click the **Variables** tab to display the properties of the variables in the two objects. This tab displays the name, type, initial value (if a value is defined), and length of each variable in both objects. The tab also indicates whether the variable is an input variable or an output variable. If the variable list for both objects is the same, this tab displays a message stating that the two objects are identical.

If either of the objects that you are comparing contain a data grid variable, you can click ⓘ beside the variable name to display the data type and length of each data grid column.


Compare Data Grid Column Metadata

Rule Set	verify_offers_special (1.0)		verify_offers (1.0)	
Column ↑	Data Type	Length	Data Type	Length
annual_fee	Decimal		Decimal	
APR			Decimal	
APR_special_rate	Decimal			
card_type	Character	100	Character	32767
cashback_gas	Decimal		Decimal	
cashback_stream	Decimal		Decimal	
credit_limit	Decimal		Decimal	
intro_bonus	Decimal		Decimal	

4 (Optional) Click **Export** to export the results of the comparison to a PDF file. The Export Comparison Results window appears.

5 (Optional) Select the information that you want to export, and click **Export**. To export the information on a specific tab, select the tab name. To export the data type and length of each column in data grid variables, select **Data grid metadata**. If you want the PDF file to display only the differences between the two objects, select **Show differences**. To display all of the objects' information, select **Show all**.

The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Compare Rule Set Code

You can compare the generated code of two different rule sets, or you can compare the generated code of two different versions of the same rule set.

1 Select the objects that you want to compare.

- To compare the generated code of two different rule sets, select the rule sets in the category view, click ⋮, and select **Compare code**.
- To compare the generated code of two versions of the same rule set, open the rule set, click ⋮ on the **Versions** tab, and select **Compare code**.

The Select Versions window appears.

2 Select the versions that you want to compare, and click **Compare**. SAS Intelligent Decisioning opens the two objects side-by-side in the Compare Code window and highlights the differences.

3 (Optional) Click **Export** to export the results of the comparison to a PDF file. The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Generate PDF Documentation for a Rule Set

You can generate detailed documentation for a rule set as a PDF document. The PDF includes the rule set properties, details about each of the rules, and a table of the variables that are used in the rule set. You can download additional documents for lookup tables that are used in the rule set.

1 Open the rule set.

2 Click ⋮, and select **Create document**. The Create Document window appears.

> **TIP** If this option is disabled, there might be unsaved changes. Click 🖫.

3 (Optional) Enter a name for the document if you do not want to use the default name.

4 (Optional) Select **Choose additional documents to download** to display a window from which you can download additional documents. You can download documents for any lookup tables that are used in the rule set.

5 Click **Create**. SAS Intelligent Decisioning creates the PDF. The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

   If you download additional documents in Step 4, the Download Additional Documents window appears.

6 (Optional) Click **Download** for each additional document that you want to download.

7 (Optional) Click **Close** to close the Download Additional Documents window.

Managing Rule Sets

Duplicating Rule Sets

About Duplicating Rule Sets in Folders for Which the Check-Out and Commit Feature Is Enabled

When you duplicate a rule set, SAS Intelligent Decisioning first creates an empty rule set, and then updates the rule set to add the individual rules. The default permissions for the Decision Repository enable users to create new objects, but the permissions do not include the ability to update the objects. In order to duplicate an object that is in the Decision Repository, you must check out the object.

If your site has created additional folders for which the check-out and commit feature is enabled, and if those folders have been assigned the same permissions as those assigned to the Decision Repository, you must also check out objects in those folders before you can duplicate them.

Duplicate Rule Sets

Note: You cannot duplicate a rule set if it is open.

To duplicate a single rule set:

1 In the Rule Sets view, select the rule set that you want to duplicate.

2 Click ⋮ and select **Duplicate**. The Duplicate Rule Set window appears.

3 Enter a new name for the duplicate rule set.

4 (Optional) Enter a description for the rule set.

5 Select the version of the rule set that you want to duplicate.

6 Click 🗀 and select the location where you want to save the duplicate rule set.

7 Click **Duplicate**.

To duplicate multiple rule sets:

1 In the Rule Sets view, select the rule sets that you want to duplicate.

2 Click ⋮ and select **Duplicate**. SAS Intelligent Decisioning duplicates the rule sets and appends `_Copy` to the names of the duplicate copies. If needed, a number is also appended to the names of the duplicate copies.

Note: When you duplicate a rule set, SAS Intelligent Decisioning creates a relationship between the original rule set and the duplicate rule set. If either object is changed, and you later copy the contents of one object into the other, SAS Intelligent Decisioning displays a warning message. Verify that you want to replace the current contents of the rule set before you paste the new content.

Delete Rule Sets

Note: You cannot delete a rule set if it is open.

In the Rule Sets view, select the rule sets that you want to delete, click ⋮, and select **Delete**. SAS Intelligent Decisioning moves the rule sets to the recycle bin. For more information, see "Manage Folders and Folder Content" on page 9.

Rename Rule Sets

Note: You cannot rename a rule set if it is open.

1 In the Rule Sets view, select the rule set that you want to rename.

2 Click ⋮ and select **Rename**. The Rename window appears.

3 Enter a new name for the rule set, and click **Rename**.

Move Rule Sets to a Different Folder

1 In the Rule Sets view, select the rule sets that you want to move.

2 Click ⋮ and select **Move**. The Choose a Location window appears.

3 Select the location to which you want to move the rule sets, and click **OK**.

Managing Versions of Rule Sets

Set the Displayed Version

The displayed version is the version whose information is displayed on the other tabs, such as the **Properties**, **Rule set**, and **Variables** tabs. On the **Versions** tab, a ✓ in the **Displayed Version** column indicates the displayed version. To change the displayed version, click the version number for the version that you want to view. The displayed version is shown in the title bar.

Create a New Version

Note: For objects that are stored in locations for which the check-out and commit feature is enabled, you cannot manually create a new version. The only way to create a new version is to check out an existing version and commit a new version. For information, see "Check Out and Commit a Rule Set Version" on page 56.

Note: The current version of an object is the version with the highest version number. When you create a new version, SAS Intelligent Decisioning locks the current version before it creates the new version.

IMPORTANT You cannot unlock a locked version. You cannot save changes to a version that is locked. If you modify a version that is locked and click 🖫, SAS Intelligent Decisioning asks you if you want to replace the current unlocked version with your edited version.

To manually create a new version:

1 On the **Versions** tab, click the version number for the existing version that you want to use as the basis for the new version.

2 Click **New Version**. The New Version window appears.

3 Select the version type: **Minor** or **Major**. Version numbers follow the format *major.minor*. If you select **Major**, the number to the left of the period is incremented. If you select **Minor**, the number to the right of the period is incremented.

4 (Optional) For each version tag that you want to add, either enter a custom tag or begin typing and select a tag from the list of previously entered tags. Press Enter after each tag.

Note: A tag is limited to 100 characters.

TIP All tags that have been previously entered for any object are saved in the list that is displayed when you start typing. You cannot delete tags from this list.

5 (Optional) Enter information about the new version in the **Notes** field.

TIP You can edit these notes at any time on the **Versions** tab.

6 Click **Save**.

Create a New Version Automatically When You Publish a Rule Set

You can automatically create a new version of a rule set when you publish the current version by selecting **Publish and Lock**.For information, see “Publish Rule Sets” on page 60.

Copy the Content of a Version

You can copy the content of an object’s version in the category view or on the **Version** tab for the object.

1 In the category view, complete these steps:

   a Select the rule set whose contents you want to copy.

   b Click ⋮, and select **Copy version**. The Copy Version window appears.

   c Select the version whose contents you want to copy.

   Alternatively, on the **Versions** tab of the rule set whose contents you want to copy:

   a Select the version whose contents you want to copy.

   b Click ⋮, and select **Copy version**. The Copy Version window appears.

2 Click 🗀, and select the target rule set into which you want to paste the contents of the version. You can paste version contents only into an object of the same type. If the rule set from which you are copying the version is a filtering rule set, then you must paste the contents into another filtering rule set. If it is an assignment rule set, then you must paste the version contents into another assignment rule set.

When you paste the contents, SAS Intelligent Decisioning creates a new version of the target rule set. The target object contains only the pasted content.

3 Select whether you want to create a new major or minor version.

4 (Optional) Modify the notes that will be associated with new version.

5 (Optional) Add tags that will be associated with the new version. Tags that are associated with a source object version are not automatically added to the new version. See "Add a Version Tag" on page 58

6 Click **Paste Version**, and then click **Yes**.

TIP The input and output designations for a variable on the **Variables** tab for the new version are removed, and the variable is treated as a temporary variable in the following situations:

- The new version of the target object does not use a variable that is used in an earlier version.
- The source and target object originally had a variable of the same name, and you deleted the variable from the source object before you copied it into the target object.

Note: When you copy the contents of a source object into a target object, SAS Intelligent Decisioning creates a relationship between the two objects. If the source object is modified after you copy its contents, and you later copy the contents of the target object back into the source object, SAS Intelligent Decisioning displays a warning message. Verify that you want to replace the current contents of the source object before you paste the new content.

Delete a Version

IMPORTANT When you delete a specific version, that version is deleted permanently. It is not moved into the recycle bin, and it cannot be restored.

Note: In order to be able to delete a specific version of an object, you must have permission to delete the object itself. Also, the configuration option sas.businessrules.deleteVersions must be turned on.

On the **Versions** tab, select the version that you want to delete, click ⋮, and select **Delete**.

You cannot delete the current version.

Upgrade Decisions to Use a New Version of a Rule Set

If you create a new version of a rule set that is already used in other decisions, you can upgrade the decisions to use the new version.

1 On the **Versions** tab for the rule set, click ⋮, and select **Upgrade decisions**. The Upgrade Decisions window appears. This window lists all of the decisions that include the rule set.

2 In the **Version to upgrade to** field, select the version of the rule set to which you want to upgrade the decisions.

3 Select **Automatically map variables** if you want SAS Intelligent Decisioning to automatically map new rule set variables in the decisions where the rule set is used.

  For information about how SAS Intelligent Decisioning maps variables, see "About Decision Variables and Mapping" on page 247.

4 Select the check boxes for the decisions that you want to upgrade, and click **Upgrade Decisions**.

TIP To update all of the objects that are used in a decision, see "Update Decisions to Use New Object Versions" on page 253.

Determine Which Objects Use a Rule Set

To list the objects that use a specific assignment or filtering rule set:

1 On the **Rule sets** category page, select the check box for the rule set, click ⋮, and select **View used by report**. The All Objects that Use the Selected Item window appears. This window lists all objects that use any version of the selected rule set.

  Note: Common rule sets are not listed in the Used By report.

2 (Optional) Select a specific version of the rule set. SAS Intelligent Decisioning narrows the list to include only the objects that use the selected version of the rule set.

Note: The **View used by report** option is also available from within an open rule set.

In the report, you can use the **Filter** field to filter the list of objects based on the object names. Click on an object name to open the object. Click ⓘ next to an object name to display the date on which the object was last modified and the ID of the user who modified the object. For decisions and code files that have been checked out, SAS Intelligent Decisioning also displays the IDs of the users that have checked out the objects. For decisions, SAS Intelligent Decisioning includes the decision's workflow status.

Click **Export** to export the Used By report to a PDF file. In the PDF file, you can click on an object name to open the object in a new browser tab.

Checking Out and Committing Rule Set Versions

About Checking Out and Committing Versions

Your administrator can enable the check-out and commit feature for rule sets that are in any folder by specifying the folder in the sas.businessrules.checkout.checkoutEnabledFolderPaths configuration option. Enabling this feature for a folder does not automatically modify the permissions for the folder or for the objects in it. You can still modify a rule set in the folder without checking it out, but you are expected to check out the latest version before you edit it. However, your administrator might also set permissions that require you to check out rule sets in these folders before you can edit them. For more information, see "sas.businessrules.checkout" in *SAS Intelligent Decisioning: Administrator's Guide* and "Set Permissions for Check-Out Folders" in *SAS Intelligent Decisioning: Administrator's Guide*.

By default, SAS Intelligent Decisioning defines a folder where you can store rule sets that must be checked out before they can be edited. This folder is the `Decision Repository` folder, and it is the default value for the sas.businessrules.checkout.checkoutEnabledFolderPaths configuration option. The default permissions for this folder require that non-administrative users check out a version and commit their changes to the checked-out version. Users who do not have administrative permissions cannot edit the rule sets in `Decision Repository` without first checking them out.

If a version can be or must be checked out before it is modified, the **Check Out** button appears at the top of the **Versions** tab for that object. You can check out any version of an object. You can check out only one copy of a version at a time.

TIP If the sas.businessrules.checkout.allowConcurrentCheckout option is turned off, and a user has checked out a rule set version, the **Check Out** button for that rule set is disabled for all other users. For more information, see "Concurrently Checking Out and Committing Rule Set Versions" on page 56.

When you check out a version, SAS Intelligent Decisioning writes a working copy of the version into your `My Folder` folder and opens the working copy. SAS Intelligent Decisioning adds "`(Checked Out)`" to the name that is displayed at the top of the window.

While you have a version checked out, the Rule Sets category view shows two rule sets with the same name, but the folders listed in the **Location** column differ for each rule set. The original version is in the location specified by the sas.businessrules.checkout.checkoutEnabledFolderPaths configuration option, and the checked-out copy is in your `My Folder` folder.

> **TIP** If an object that you have checked out does not appear in the category view, click ↻ to refresh the category view.

A **Commit** button appears at the top of the **Versions** tab for the checked-out version. When you are finished editing the checked-out version, you must commit your changes in order for other users to be able to see them. When you commit your changes, SAS Intelligent Decisioning creates a new version with your changes.

If the parent object is deleted before you commit your changes, you will not be able to commit your changes.

You cannot publish the checked-out version that is in `My Folder`. To publish a version with your changes, you must commit your changes, and publish the committed version.

Checking Out a Rule Set from within a Decision

See "Checking Out and Committing Objects from within A Decision" on page 274 for information.

Checking Out and Committing a Common Rule Set from within an Assignment Rule Set

You can check out a common rule set from within an assignment rule set if the following conditions are true:

- The assignment rule set is checked out.
- The common rule set is stored in a folder for which the check-out feature is enabled.
- The assignment rule set uses the latest version of the common rule set. (See Add a Common Rule Set Reference to an Assignment Rule Set on page 27.)

When you check out the common rule set, the assignment rule set is modified to use the checked-out copy of the common rule set. If you make changes to the common rule set and save it, you can click ↻ in the assignment rule set to display the changes in the rule set editor for the assignment rule set. When you commit the assignment rule set, all checked-out common rule sets that are used in the assignment rule set are also committed. The newly committed version of the assignment rule set is modified to use the newly committed versions of the common rule sets. Any additional assignment rule sets that use the same common rule sets and that are still checked out are also modified to use the newly committed versions of the common rule sets.

To check out or commit a common rule set from within a checked-out assignment rule set, click ⋮ to open the context menu for the common rule set reference, and select **Check out common rule set** or **Commit common rule set**.

Checking Out and Committing a Lookup Table from within a Rule Set

You can check out a lookup table from within a rule set if the following conditions are true:

- The rule set is checked out.
- The lookup table is stored in a folder for which the check-out feature is enabled.
- The rule set does not have any unsaved changes. If the rule set contains any unsaved changes, the **Check out lookup** option is disabled.

To check out or commit a lookup table from within a checked-out rule set, right-click on the LOOKUP or LOOKUPVALUE expression, and select **Check out lookup** or **Commit lookup**. SAS Intelligent Decisioning displays as asterisk in front of the name of any lookup table that is checked out.

When you check out a lookup table from within a rule set, the checked-out rule set is modified to use the checked-out copy of the lookup table, and SAS Intelligent Decisioning enables the **Commit lookup** option in the menu. When you commit the rule set, all of the checked-out lookup tables that are used in the rule set are also committed.

The newly committed versions of the lookup tables are not automatically activated when they are committed. You must activate the newly committed versions of the lookup tables. For more instructions, see “Activate a Lookup Table” on page 137.

Rule sets that are not checked out always use the activated versions of lookup tables. If you do not activate the newly committed versions of the lookup tables, all rule sets that use the lookup tables will continue to use the previously activated versions of each table.

To cancel the check out of a lookup table, right-click on the LOOKUP or LOOKUPVALUE expression, and select **Cancel check out**.

Committing a Lookup Table Together with a Rule Set

If you check out a lookup table from within a rule set and then commit the rule set, the lookup table is also committed. For more information, see “Checking Out and Committing a Lookup Table from within a Rule Set” on page 55.

Alternatively, you can check out a rule set that uses a lookup table, open the lookup table in the **Lookup tables** category view and check it out, then change the reference in the rule set to use the checked-out version of the lookup table. The lookup table is committed when you commit the rule set.

The newly committed version of the lookup table is not automatically activated when they are committed. You must activate the newly committed version of the lookup table. For more instructions, see "Activate a Lookup Table" on page 137.

Concurrently Checking Out and Committing Rule Set Versions

The ability for multiple users to check out the same rule set at the same time is controlled by the sas.businessrules.checkout.allowConcurrentCheckout configuration option. This option is turned on by default.

When this option is turned on, different users can check out the same version of the same object at the same time. Because the objects that are checked out are saved in each user's `My Folder` location, the default permissions allow individual users to see only the copies that they have checked out.

When this option is turned off and a user has checked out an object, the **Check Out** button for that object is disabled for all other users.

If multiple users check out the same version of the same object at the same time, each user's changes are preserved in a new version when they commit their changes. One user's changes do not overwrite another user's changes.

IMPORTANT If two users attempt to commit changes to the same object simultaneously, the first user's attempt will succeed but the second user might see an error message that the commit has failed. If the second user subsequently commits their changes, the **Modified By** column on the **Versions** tab for both the version committed by the first user and the version committed by the second user displays the user ID of the second user.

Check Out and Commit a Rule Set Version

1 On the **Versions** tab, click **Check Out**.

   SAS Intelligent Decisioning updates the **Properties** tab to indicate that the version is checked out.

2 Modify the checked-out version as needed, and save it.

TIP To discard the changes and delete the checked-out version from `My Folder`, you can commit the object without saving it first. However, committing the object without saving creates a new version of the object whose contents match the contents of the previous version. For information on undoing a check out, see "Undoing a Check Out" on page 57.

3 On the **Versions** tab, click **Commit**. The Commit Rule Set Version window appears.

4 Select the version type: **Minor** or **Major**. Version numbers follow the format *major.minor*. If you select **Major**, the number to the left of the period is incremented, and the minor number is reset to zero. If you select **Minor**, the number to the right of the period is incremented.

5 (Optional) In the **Version tags** field, enter any version tags that you want to associate with the new version. Press Enter after each tag. Tags are limited to 100 characters each and cannot contain the following characters: $ ' " # & ? ( ) \ /

6 (Optional) Enter information about the new version in the **Notes** field.

7 Click **Commit**. SAS Intelligent Decisioning creates a new version with your changes, and deletes the working copy from your **`My Folder`** folder.

Determine Who Has a Version Checked Out

If the current version of an object is checked out, the IDs of the users that checked it out and the timestamps when each user checked it out appear in the **Checked out by** field on the **Properties** tab for the original object. You can also display this information by clicking ↗ beside the version number on the **Versions** tab.

Opening the Original Object

When you check out an object, SAS Intelligent Decisioning adds the field **Original object link** to the **Properties** tab for the checked-out object. This field contains a link to the original object that was checked out. You can use this link to verify that you have checked out the correct version and to compare the original content with the modified content in the checked-out version.

Undoing a Check Out

How you undo the checkout of an object depends on how the object was checked out.

If both an object and a decision that uses the object are checked out at the same time, or if you checked out the object from within the decision, click ⋮ on the object's node in the decision diagram, and select **Cancel checkout**.

If you check out a common rule set from within an assignment rule set, click ⋮ in the assignment rule set editor to open the context menu for the common rule set, and select **Cancel check-out**.

You can discard a checked-out version and any changes that you made by deleting the working copy of the version from your `My Folder` folder if the following conditions are true:

- You have not checked out a decision that uses the object.
- The object was not checked out at the same time as a decision that uses the object, or the object was not checked out from within the decision after the decision was checked out.

The deleted version is moved to the recycle bin. See "Delete Rule Sets" on page 48.

Managing Version Tags for Rule Sets

Add a Version Tag

Version tags enable you to better organize and group your content. Version tags are associated with specific versions of an object and not with the entire object. You can add the same tag to any version of any object. To add a tag to a rule set version:

1. On the **Versions** tab, position your cursor in the **Version Tags** column for the version that you want to tag. If the version is not tagged, + appears. If the version has at least one tag, ✎ appears.
2. Click + to open the New Version Tags window, or click ✎ to open the Edit Version Tags window.
3. For each tag that you want to add, either enter a custom tag or begin typing and select a tag from the list of previously entered tags. Press Enter after each tag. Tags are limited to 100 characters each and cannot contain the following characters: $ ' " # & ? ( ) \ /

TIP All tags that have been previously entered for any object are saved in the list that is displayed when you start typing. You cannot delete tags from this list.

TIP To filter the version list based on a tag, right-click on the **Version Tags** column heading, enter a filter string in the **Filter** box, and press Enter. To clear the filter, delete the string from the **Filter** box, and press Enter.

4 Click **Close** to close the window.

Remove a Version Tag

1 On the **Versions** tab, position your cursor in the **Version Tags** column for the version whose tag you want to remove, and click ✎.

2 Click × beside the tag that you want to remove.

3 Click **OK** to close the window. The tag remains in the list of previously entered tags that is displayed when you add a tag, but the tag is no longer associated with version.

Modify a Version Tag

You cannot modify a version tag that already exists. To change the content of an existing tag, delete the tag as described in "Remove a Version Tag" on page 59, and then add the tag again as described in "Add a Version Tag" on page 58.

Publishing a Rule Set

Introduction to Publishing

Publishing content makes it available to other applications. Publishing a rule set creates an entity that can be managed and run in another environment. When you publish content, SAS Intelligent Decisioning generates code for that content and writes it to the destination. The following table describes what form the generated code takes for each destination type.

Table 2.4 What SAS Intelligent Decisioning Publishes to Each Destination Type

Destination Type	**What** SAS Intelligent Decisioning **Does**
SAS Cloud Analytic Services (CAS), Teradata, or Apache Hadoop	Adds a row to the model table for the destination
SAS Micro Analytic Service	Writes a Micro Analytic Service module in the service

Destination Type	**What** SAS Intelligent Decisioning **Does**
Git	Creates a directory in the remote Git repository with the same name as the published object and writes the generated code to a file named `scoreResource.txt`.
Container destinations	Creates a *SAS Container Runtime* container and writes it to the destination. **Note:** The container destination types to which you can publish content from SAS Intelligent Decisioning are Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), and private Docker repositories.

The rows in the model tables, the Micro Analytic Service modules, and the SAS Container Runtime containers all become callable REST API endpoints, independent of SAS Intelligent Decisioning. For content that is published to Git destinations, you can use the SAS Intelligent Decisioning Git Deployment CLI to deploy content from the remote repository either to a CAS destination or to a SAS Micro Analytic Service destination. The deployed content then becomes a callable REST API endpoint.

Publish Rule Sets

1 Open the rule set, and click the **Versions** tab.

By default, the version that is published when you click **Publish** is the displayed version.

2 (Optional) Click the version number to change the displayed version to the version that you want to publish.

3 (Optional) Enter any tags that you want to associate with the published object. Press Enter after each tag.

4 Either click **Publish** or select **Publish and Lock**.

- To publish the selected version without also creating a new minor version, click **Publish**.
- To lock and publish the selected version and create a new minor version at the same time, click ⋮, and select **Publish and Lock**.

The Publish Rule Sets window appears.

5 Select the destination to which you want to publish. The publishing destinations that are available to you depend on what is configured at your site. See *SAS Viya Platform: Publishing Destinations* for more information.

Note: You cannot publish filtering rule sets to SAS Micro Analytic Service destinations, to Git destinations, or to container destinations. Also, you cannot publish rule sets that use SAS Data Quality functions to SAS Micro Analytic Service destinations, and rule sets that use these functions will fail if they are run inside the database.

6 (Optional) In the **ITEMS TO PUBLISH** section, complete the following steps for each item that you are publishing:

a Edit the **Published name** if you do not want to use the default name for the published module. The maximum length and character restrictions differ depending on your destination. See Table 2.5.

Table 2.5 *Requirements and Restrictions for Published Names*

Destination	Maximum Length	Requirements And Restrictions
Container destinations	127	The published name must start with a letter or an underscore. It cannot contain spaces, multi-byte characters, or special characters other than the underscore. The name that you enter is assigned to the SAS_SCR_APP_PATH environment variable. The value of this variable determines the module name. For more information, see "Changing the Endpoint Name for a Container" in *SAS Container Runtime: Programming and Administration Guide*.
Git	128	The published name cannot contain forward slashes (/), single quotation marks ('), or double quotation marks (").
SAS Micro Analytic Service	100	The published name cannot contain the following characters: ! @ # $ % ^ & * ( ) | = ~` \ / . { } " ' ;
SAS Cloud Analytic Services (CAS)	128	The published name cannot contain single or double quotation marks.
Teradata	128	The published name must start with a letter or an underscore. It cannot contain spaces, multi-byte characters, or special characters other than the underscore.
Apache Hadoop	128	The published name cannot contain colons (:) or double quotation marks.

b If you have previously published the rule set, turn on the **Replace item with the same name** option in order to replace the previously published rule set of the same name in the same destination.

c Select the **Rule-fired tracking** check box if you want the published rule set to generate rule-fired data.

> **TIP** This rule-fired data is recorded in the ruleFiredFlags column in the output table. The rule-fired data that is recorded when you select **Record rule-fired data** for a record contacts node is recorded in the subject contact history.
>
> This option is disabled if you are publishing content to a SAS Micro Analytic Service destination, a Git destination, or a container destination.

7 Publish the rule sets.

To publish content to a SAS Cloud Analytic Services (CAS) destination, you must reload the CAS destination table in order to make the newly published items available to other applications. You do not need to reload the destination table when you publish content to other destination types.

Select one of the following options:

Publish
publishes the rule sets and, if you are publishing content to a CAS destination, automatically reloads the CAS destination table. If another user is executing the code for an item that was previously published to CAS while the destination table is being reloaded, reloading the table might cause temporary problems with accessing the table content. After the table is reloaded, all authorized users can access all items in the table.

Publish without reloading
publishes the rule sets but does not reload the CAS destination table. You must manually reload the table in order for the newly published items to be accessible.

Publish and lock
locks and publishes the rule set and creates a new minor version, but does not reload the CAS destination table. You must manually reload the table in order for the newly published items to be accessible.

Note: This option is not available unless you chose **Publish and Lock** in Step 4.

The Publishing Results window appears. It displays the names of the published items, their status, and a link to the log that was generated during the publishing process.

8 After the status changes to **Published successfully**, click **Close** to close the Publishing Results window.

> **TIP** To view the publishing history for a rule set, click the **History** tab.

9 (Optional) Click **Close** to close the rule set.

Testing Rule Sets

Ways to Test a Rule Set

There are three types of tests:

Basic test
: executes the rule set in the SAS Cloud Analytic Services (CAS) destination using the input table that you specify. You can also specify a debugging variable. For more information, see "Create and Run a New Test" on page 63.

Scenario test
: enables you to enter specific input values and the output values that you expect the test to generate. A scenario test identifies differences between the output that you expect to see and the actual output that is generated when the test is run. You can also compare the test definitions and test results of different scenarios. Scenario tests are also run in CAS. For more information, see "Test a Scenario" on page 289.

Publishing validation test
: executes the rule set in a publishing destination using the input table that you specify. When you publish the rule set, a validation test is automatically defined for that rule set in that destination. For more information, see "Validate a Published Rule Set" on page 73.

Test a Rule Set

Create and Run a New Test

Testing a rule set is optional, but doing so is a best practice. Testing enables you to discover any problems before the rule set is published and incorporated into a production system.

> **IMPORTANT** If you are a testing rule set that uses functions that are defined in a custom context file, verify that the context file is specified in the **Test custom context** field on the **Properties** tab before running the test.
>
> If you are testing a rule set that uses a lookup table and both the rule set and the lookup table are checked out, the rule set test uses the checked-out version of the lookup table. If the lookup table is not checked out, the test uses the activated version of the lookup table.

1 On the **Scoring** tab, click the **Tests** tab.

2 Click **New Test**. The New Test window appears.

3 Enter a name for the test if you do not want to use the default name. The name cannot contain forward slashes (/) or curly braces ({}).

4 (Optional) Enter a description for the test. Descriptions are limited to 1000 characters.

5 (Optional) Click 🗀 for the **Location** field, and select the folder where you want to save the test definition and results.

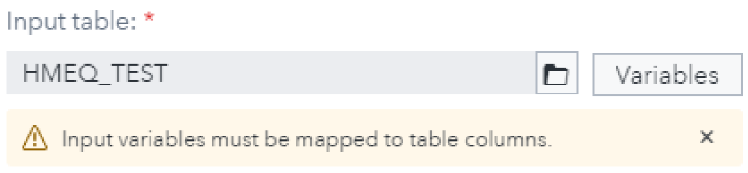
> **TIP** Selecting a location is optional, but it is highly recommended. Storing test definitions and test results in a folder simplifies the tasks of setting permissions and transferring the test files.

6 Click 🗀 for the **Input table** field, select the input table for the test, and click **OK**.

7 Verify or change the variable mappings. To run a full test, map all of the input variables to columns in the input table that you selected for the test. To run a partial test, you can map only the input variables that are needed for the test.

SAS Intelligent Decisioning automatically maps the input variables in the rule set to columns in the input table when the names and data types of the variables match those of the table columns.

If any input variables are not mapped to input columns or to static values, the application displays a warning message. At run time, SAS Intelligent Decisioning assigns missing values to input variables that are not mapped.



Input variables must be mapped to table columns.

You can change the automatic variable mappings in the Variable Mappings window.

To change variable mappings:

a Click **Variables**. The Variable Mappings window appears.

b For each input variable, select the table column to which the variable should be mapped. If the input table contains more than 25 columns, click **More columns** to display additional column names. Alternatively, for Decimal, Integer, and Character variables, you can select **Use value** for the table column, and specify a literal value in the **Value** column. When you are entering literal values, remember these rules:

- Do not enclose character strings in quotation marks.
- To specify a missing value for character variables, select **Use value** and leave the **Value** column empty. When SAS Intelligent Decisioning generates code, it generates an empty string (`''`). For numeric values, enter a period (.).

Note: For content that will be published to a SAS Micro Analytic Service destination, the values of date and datetime input variables must be numeric. They cannot be formatted values such as `18jul2019`. Also, integer variables are converted to decimal variables when the content is published.

c Click **OK** to close the Variable Mappings window.

8 (Optional) Click **Advanced** to display the advanced options.

9 (Optional) Click 🗀 and select the library where you want to write the output of the test.

10 (Optional) Enter a name for the test results output table if you do not want to use the default name. The default name is `test-name_object-name_timestamp_output`.

11 (Optional) Select the version of the rule set that you want to test.

12 (Optional) Select the variable that you want to serve as an input debug variable. You can specify an input-only variable or an input-output variable. The rule set writes the name and value of this variable to the log for each input record that is processed. It writes the value just before the logic of the rule set is executed for the input record.

For more information, see "Debugging Rule Set Tests" on page 66.

13 (Optional) Select **Preserve unmapped columns in the output table** if you want columns that are not mapped to an output variable to be written to the output table.

14 (Optional) If you are testing a filtering rule set, select **Keep filtered records in output table** to include records in the test output that would normally be excluded. When you select this option, SAS Intelligent Decisioning creates a new column in the output table named _filter_. If the value in this column is 0, then the record does not match any of the conditions in the rule set, and the record is filtered out of the output when the rule set is run in a production environment. If the value in this column is 1, then the record matches at least one condition in the rule set, and the record is included in the output.

Note: This option is ignored for assignment rule sets.

15 Click **Run** to run the test. Alternatively, click **Save** to save the test definition without running it.

The status of the test is indicated by the icon in the **Status** column. For explanations of each icon, see "Status Icons for Tests" on page 76.

16 Click ▤ in the **Results** column to view the results of the test.

17 In the test results window, click **Test Results** in the navigation panel to display the URIs and other information for the test. Click **Output**, **Code**, or **Log** to display the output data set, the code that was generated by SAS Intelligent Decisioning, or the SAS log that was generated when the code was run.

TIP You can click on the values of character variables to display the entire value in a separate window. For data grid variables, you can choose to view the variable value in three different formats:

- Click the **Data Grid** tab to view the data grid value as a table.
- Click the **Formatted** tab to view the data grid as a formatted JSON character string.
- Click the **Plain** tab to view the data grid as an unformatted character string.

TIP On the **Log** page, you can click ⭳ to download the log file.

Debugging Rule Set Tests

When you create a test, you can specify a variable to use as a debugging variable in the **Input debug variable** field. You can specify an input-only variable or an input-output variable. The rule set writes the name and value of this variable to the log for each input record that is processed. It writes the value just before the logic of the rule set is executed for the input record. For more information, see Step 12 of "Create and Run a New Test" on page 63.

When you specify an input debug variable, you can use the sas.businessRules.messageOrder configuration option to control whether the log messages are written as they are produced or after the rule set executes. For more information, see "sas.businessrules.messageOrder" in *SAS Intelligent Decisioning: Administrator's Guide*.

When you specify an input debug variable, SAS Intelligent Decisioning automatically sets the maximum number of threads that can be allocated for the test to 1. Setting the thread count to 1 ensures that the variable's values are written to the log in the correct order and are not affected by different threads completing at different times.

To capture variable values for input-only or temporary variables after the rule set logic has executed for a specific record, you can specify that the variable is an output variable, and then re-run the test. Before publishing the rule set to a production environment, return the input and output settings for the variable to their previous settings. For more information, see "Input Variables, Output Variables, and Temporary Variables" on page 17 and "Edit Variable Properties" on page 23.

Test a Scenario

Create and Run a Scenario Test

IMPORTANT If you are a testing rule set that uses functions that are defined in a custom context file, verify that the context file is specified in the **Test custom context** field on the **Properties** tab before running the test.

If you are testing a rule set that uses a lookup table and both the rule set and the lookup table are checked out, the rule set test uses the checked-out version of the lookup table. If the lookup table is not checked out, the test uses the activated version of the lookup table.

1 On the **Scoring** tab, click the **Scenarios** tab.

2 Click **New Test**. The New Scenario Test window appears.

3 Enter a name for the test if you do not want to use the default name. The name cannot contain forward slashes (/) or curly braces ({}).

4 (Optional) Click 🗀 for the **Test definition location** field, and select the folder where you want to save the test definition.

TIP Selecting a test definition location is optional, but it is highly recommended. Storing test definitions in a folder simplifies the tasks of setting permissions and transferring the test files.

5 Click 🗀 for the **Output table location** field, and select the folder where you want to save the test results.

6 (Optional) Select the version of the rule set that you want to test.

7 (Optional) Enter a description for the test. Descriptions are limited to 1000 characters.

8 Enter the values that you want to use for each input variable. You do not have to enter values for every input variable. At run time, SAS Intelligent Decisioning uses missing values to input variables for which you do not specify a value.

Note: Values longer than 32767 characters for character variables that are either input-only or input-output will be truncated. You cannot enter input values for variables of type binary or of type varying-length binary.

To enter values for the columns in a data grid variable:

a Click in the **Value** field, and then click ✎. The Edit Data Grid window appears.

b Click + to add the row, and then enter the values for each column.

Repeat this step for each row of values that you want to add to the data grid.

TIP By default, data grid column names appear across the top of the data grid view, and row numbers appear down the left side. You can click to change the view of the data grid so that row numbers appear across the top and data grid column names appear down the left side.

c Click **OK** to save the data grid values and close the Edit Data Grid window.

9 (Optional) For each output variable, select the **Include** check box and enter the expected output value. The **Include** check box controls whether a variable's expected value is used to determine the status of a scenario test. If you select **Include** for a variable and the test does not return the expected value, the test status is set to Completed with warnings (⚠). If you do not select the check box, the application ignores the expected value of that variable when it determines the status of the test.

To enter expected values for the columns in a data grid variable, click in the **Expected Output** field, and follow the instructions in Step 8.

Note: A scenario test cannot verify issues with trailing spaces. For example, it cannot distinguish between a string that contains a single space `' '` and a string that contains three spaces `'   '`.

10 Click **Run** to run the test. Alternatively, click **Save** to save the test definition without running it.

The status of the test is indicated by the icon in the **Status** column. For explanations of each icon, see "Status Icons for Tests" on page 76.

11 Click in the **Results** column to view the results of the test.

12 On the Test Results page, click **Test Results** in the navigation panel to display the URIs and other information for the test. Click **Input** to display the values of input variables and dynamic attributes. Click **Output**, **Code**, or **Log** to display the output data set, the code that was generated by SAS Intelligent Decisioning, or the SAS log that was generated when the code was run.

On the **Output** page, click **Show All** to display the expected and actual values of all output variables. Click **Show Differences** to display only the variables whose expected values do not match the actual values that were returned by the test.

TIP You can click on the values of character variables to display the entire value in a separate window. For data grid variables, you can choose to view the variable value in three different formats:

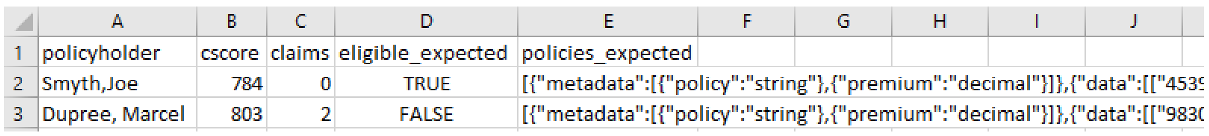
- Click the **Data Grid** tab to view the data grid value as a table.
- Click the **Formatted** tab to view the data grid as a formatted JSON character string.
- Click the **Plain** tab to view the data grid as an unformatted character string.

TIP On the **Output** page, click **Export** to export the output table as a comma-separated values (CSV) file. On the **Log** page, click to download the log file.

Import Scenario Test Definitions

You can import scenario test values from a comma-delimited (CSV) file. Each line in the CSV file is imported as one scenario test. In the CSV file, add a column of values for each variable. In the header row, enter the names of the input variables and of the output variables with `_expected` appended to the name.

For example, suppose your scenario test has the input variables `policyholder`, `cscore`, and `claims`, and it has the output variables `eligible` and `policies`. An import file for this test might appear in a spreadsheet application as shown in the following figure.



	A	B	C	D	E	F	G	H	I	J
1	policyholder	cscore	claims	eligible_expected	policies_expected					
2	Smyth,Joe	784	0	TRUE	[{"metadata":[{"policy":"string"},{"premium":"decimal"}]},{"data":[["453					
3	Dupree, Marcel	803	2	FALSE	[{"metadata":[{"policy":"string"},{"premium":"decimal"}]},{"data":[["983					

The policies output variable in this example is a data grid variable. To enter values for data grid variables, use the JSON string format described in "Introduction to Data Grids" in *SAS Intelligent Decisioning: Using Data Grids*.

The format for date and datetime variables depends on your locale. Use the same format that is created by the date and datetime pickers when you click or to enter initial values for custom variables on the **Variables** tab.

To import scenario test definitions:

1. On the **Scoring** tab, click the **Scenarios** subtab, and then click **Import Scenarios**. The Import Scenarios window appears.
2. In the **Import from** field, click and select the CSV file that contains the scenario test values.

Note: The import file is limited to 10 MB.

3 Select or enter the encoding of the CSV file.

4 Enter a prefix for the scenario test definitions. SAS Intelligent Decisioning appends a number to this prefix for each test definition. The prefix can include double-byte characters and special characters, including single quotation marks.

5 (Optional) Click 🗀 for the **Folder location** field, and select the folder where you want to save the test definitions.

6 Click 🗀 for the **Output table location** field, and select the folder where you want to save the test results.

7 Click **Import**.

Compare Different Scenarios

You can display the scenario definitions or results of two or more tests side-by-side. On the **Scenarios** tab, select two or more tests, click ⋮ , and select one of the following options:

Compare ⇨ Definitions
displays the input values and the expected output values that you entered for both tests. To edit the input values for a scenario test definition, click ✎ next to the test name under **Input Table**. To edit the expected output values for a test, click ✎ next to the test name under **Output Table**.

Compare ⇨ Results
displays the input values and the actual output values that were generated by the test. To display both the expected values and the actual values in the output table, select **Display expected values**. For each variable for which you selected **Include** in Step 9 on page 68, the application highlights the variable if the actual and expected values do not match.

Click **Export** to export the results comparison as a comma-separated values (CSV) file. The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Run a Rule-Fired Analysis

If a rule's conditions evaluate to true, then the rule is said to have *fired*. Rule-fired data includes summary information about how many times each rule fired and detailed information for each time that a rule evaluates to true. See "How Rules Are Evaluated and When Rule-Fired Records Are Generated" on page 32 for more information.

Note: Rule-fired data is recorded only for rule sets, including rule sets in nested decisions and filtering rule sets that are included directly in a decision. Rule-fired data is not recorded for other objects, including filtering rule sets that are used as eligibility rule sets in treatments.

For filtering rule sets that are included directly in a decision, rule-fired data is recorded for an input record if the rule set does not filter out the record. When the rule set filters out an input record, rule-fired data is not recorded for that input record.

For rule sets that iterate over a data grid (in other words, the rule sets score the rows in the data grid), the rule-fired data indicates that the rules in the rule set fire once for the entire data grid instead of firing once for each row in the data grid.

If the sas.decisions.nodetraces.includeRuleFiredPathTrackInfoInVariableAssignmentLogging configuration option is turned off, you cannot run rule-fired analyses for scenario tests. For more information, see "Decisions Service Properties" in *SAS Intelligent Decisioning: Administrator's Guide*.

TIP This rule-fired analysis uses the data that is in the ruleFiredFlags column in the test results output table. To analyze rule-fired data that is in the subject contact history, use the %DCM_GET_SUBJECTCONTACT_HISTORY and %DCM_RULEFIRE_DETAIL macros. For more information, see *SAS Intelligent Decisioning: Macro Guide*.

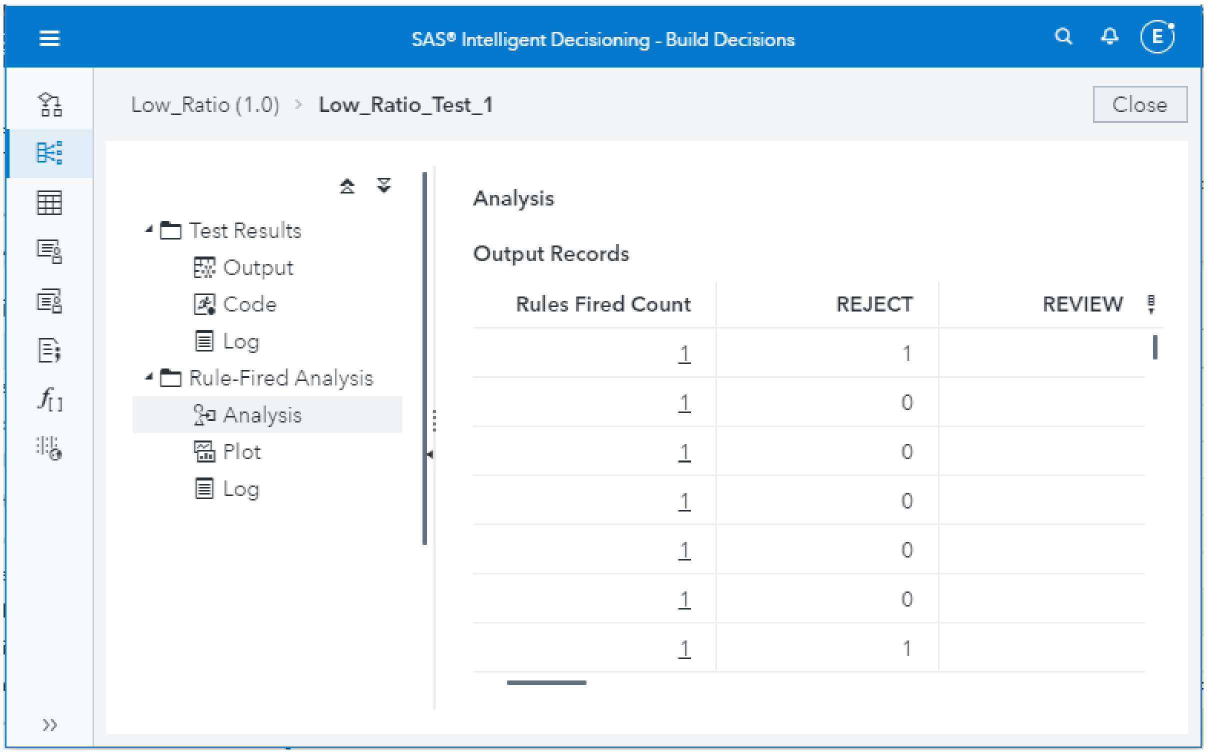
To run a rule-fired analysis:

1 In the test results window, click **Rule-Fired Analysis** in the navigation panel.

2 Click **Run Rule-Fired Analysis**. SAS Intelligent Decisioning analyzes the test results to determine which rules fired for each row in the input table, and displays the Analysis page.

  The Analysis page displays the number of rules that fired for each output record that was generated by the decision. The number in the **Rules Fired Count** column is a link to more information. You can click on this link to display the rules that fired for that output row.

  For example, the following displays shows the rule-fired analysis for the low_ratio rule set.


SAS® Intelligent Decisioning - Build Decisions

Low_Ratio (1.0) > Low_Ratio_Test_1 Close

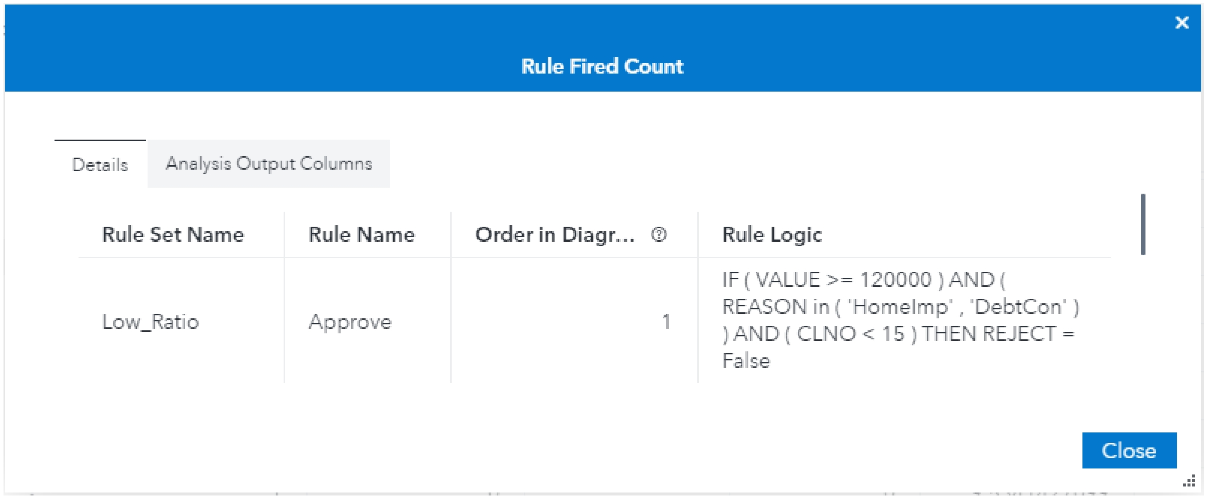
- Test Results
  - Output
  - Code
  - Log
- Rule-Fired Analysis
  - Analysis
  - Plot
  - Log

Analysis

Output Records

Rules Fired Count	REJECT	REVIEW
1	1	
1	0	
1	0	
1	0	
1	0	
1	0	
1	1	

3 Click on a number in the **Rule Fired Count** column. SAS Intelligent Decisioning displays the Rule Fired Count window. This window shows which rules produced the selected output record.


Rule Fired Count

Details | Analysis Output Columns

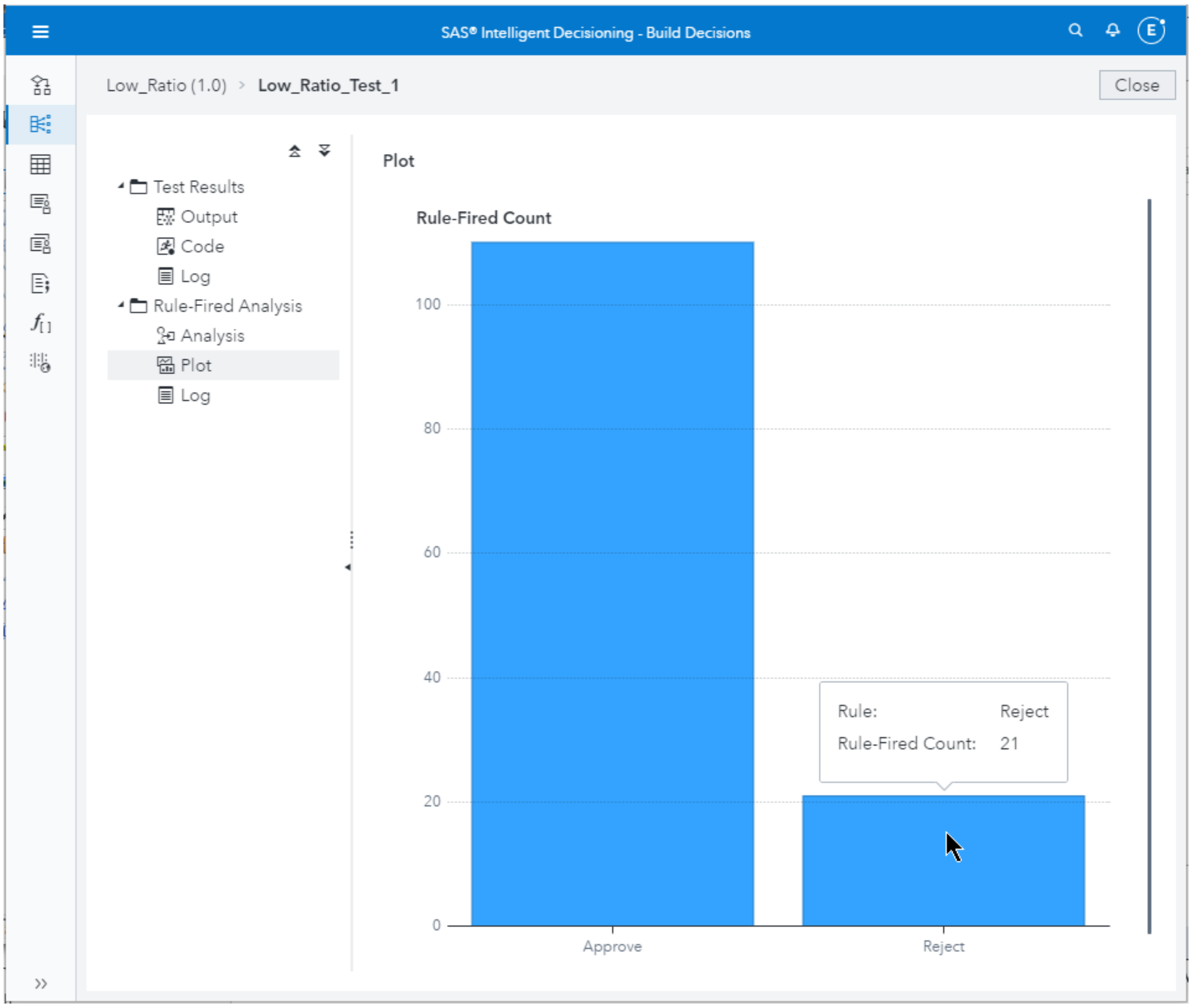
Rule Set Name	Rule Name	Order in Diagr...	Rule Logic
Low_Ratio	Approve	1	IF (VALUE >= 120000) AND (REASON in ('HomeImp' , 'DebtCon')) AND (CLNO < 15) THEN REJECT = False

Close

Note: The value in the **Rule Order** column is the ordinal number of the rule as it occurs in the rule set diagram. The values in this column do not indicate the order in which the rules in the rule set fired.

4 Click **Close** to close the Rule Fired Count window.

5 Click **Plot** in the navigation panel. SAS Intelligent Decisioning displays a bar chart that shows how many times each rule fired. Position your cursor over a bar to display the name of the rule and the number of times that the rule fired.



6 Click **Rule-Fired Analysis** in the navigation panel to display the URIs and other information for the rule-fired test.

7 Click **Close** to close the rule set.

Validate a Published Rule Set

You can test the published rule set in the target publishing destination. When you publish the rule set to any destination type except Git, a validation test is automatically defined for that rule set in that destination. To run the publishing validation test:

1 On the **Scoring** tab, click the **Publishing Validation** tab. The ○ icon in the **Status** column indicates that the test is not ready to run. The ◉ icon indicates that the test is ready to run.

2 Click on the test name. The Edit Publishing Validation Test window appears.

Note: To generate the name of the publishing validation test, SAS Intelligent Decisioning appends a timestamp to the rule set name. The timestamp indicates when the rule set was published.

3 (Optional) Click 🗀 in the **Location** field, and select the folder where you want to save the test definition and results.

> **TIP** Selecting a location 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 test files.

4 Click 🗀 in the **Input table** field, and select the input table for the test.

Note: For content that will be published to a SAS Micro Analytic Service destination, the values of date and datetime input variables must be numeric. They cannot be formatted values such as `18jul2019`. Also, integer variables are converted to decimal variables when the content is published.

Note: If the input table contains a character column, and that column contains control characters in any row, do not use the table as input for publishing validation tests.

If you are validating content that was published to SAS Micro Analytic Service, the time required to run the test depends on the number of worker threads on your system, the number of threads in the middle tier, and the network latency between CAS and the middle tier server. It is recommended that you select an input table with as few input records as needed to accurately test the published content. See *SAS Micro Analytic Service: Programming and Administration Guide* for more information.

5 (Optional) Expand the **Advanced** section, click 🗀 in the **Output data library** field, and select a library to store the validation test output data.

6 Click **Run** to run the test. Alternatively, click **Save** to save the test definition without running it.

The status of the test is indicated by the icon in the **Status** column. For explanations of each icon, see "Status Icons for Tests" on page 76.

7 Click ▤ in the **Results** column to view the test results.

8 In the test results window, click **Test Results** in the navigation panel to display the URIs and other information for the test. Click **Output**, **Code**, or **Log** to display the output data set, the code that was generated by SAS Intelligent Decisioning, or the SAS log that was generated when the code was run.

TIP You can click on the values of character variables to display the entire value in a separate window. For data grid variables, you can choose to view the variable value in three different formats:

- Click the **Data Grid** tab to view the data grid value as a table.
- Click the **Formatted** tab to view the data grid as a formatted JSON character string.
- Click the **Plain** tab to view the data grid as an unformatted character string.

TIP On the **Log** page, you can click ⤓ to download the log file.

9 Click **Close** to close the rule set.

Working with Test and Validation Test Output Data

After you run a test or a publishing validation test, you can work with the output table in other SAS applications to analyze the data, create and compare models, discover relationships hidden in the data, and generate reports based on the data.

Note: The actions available to you depend on the applications that are available at your site.

On the Test Results page, select the **Output** table in the navigation pane, click **Actions**, and select one of the following options:

Explore Lineage
opens SAS Lineage Viewer. SAS Lineage Viewer enables you to better understand the relationships between objects in your SAS Viya applications. These objects include data, transformation processes, reports, and visualizations. For more information, see *SAS Lineage: User's Guide*.

Explore and Visualize Data
opens the output table in SAS Visual Analytics. SAS Visual Analytics enables you to create, test, and compare models based on the patterns discovered during exploration of the data. You can export the model before or after performing model comparison for use with other SAS products or to put the model into production. SAS Visual Analytics supports a range of visualization, discovery, and reporting features. For more information, see *Welcome to SAS Visual Analytics*.

Prepare Data
opens the output table in SAS Data Studio. SAS Data Studio enables you to perform data transforms such as joining tables, appending data to a table, transposing columns, creating calculated columns, and so on. For more information, see *SAS Data Studio: User's Guide*.

Status Icons for Tests

Icon	Status
●	The test is not ready to run. The test definition is not complete, or it might contain errors.
◎	The test has been defined and can be run. Some input variables have not been mapped or have not been assigned a value, so the test might execute only a subset of the rules in the rule set.
◉	The test is defined correctly and is ready to run.
◌	The test is running.
✓	The test completed successfully.
✓⚠	The test completed, but warnings were issued in the SAS log. The URI to the log file is shown on the Test Results page. On the Test Results page, click **Test Results** in the navigation panel to display the URI.
⊗	The test did not run successfully. Check the SAS log for information. The URI to the log file is shown on the Test Results page. On the Test Results page, click **Test Results** in the navigation panel to display the URI.

Manage Comments for a Rule Set Test

You can associate comments and attachments with rule set tests.

To open the Comments properties pane, select the rule set test on the **Scoring** tab, and click 💬 in the property pane.

To add a new comment, enter the comment in the text box and click **Post**.

To add an attachment to a comment, click 📎, select the file that you want to attach, and click **Post**. (The attachment icon appears after you enter text in text box.) You cannot attach executable files such as BAT and EXE files.

To reply to a comment, click **Reply**, enter your reply in the text box, and click **Post**.

To delete a comment, click 🗑 for that comment.

Executing Published Content

How you execute published content depends on the destination to which the content is published.

Executing Content Published to SAS Micro Analytic Service Destinations

The user who is executing the published content must be authenticated. In SAS Viya, authentication options vary, based on which interface and operating system are used in your environment. External mechanisms include direct LDAP authentication, host authentication, Kerberos, Security Assertion Markup Language (SAML), and OAuth 2.0 with OpenID Connect. For additional information, see *SAS Viya Platform: Authentication*

When a rule set is published from SAS Intelligent Decisioning to a SAS Micro Analytic Service destination, an EXECUTE step is created in the published module. For information about the request and response data formats used in this step, see Execute a step in the REST API documentation for the Micro Analytic Score API.

IMPORTANT If the SAS Micro Analytic Service configuration property `service.removetrailingunderscoresfrominputs` is set to **`False`**, add an underscore to the name of each input variable. If this option is set to **`True`**, do not add underscores. Your administrator can add this property to the `supplementalProperties` section in the `sas.microanalyticservice.system` configuration definition in SAS Environment Manager. The default value for this option is **`False`**. For additional information, see "sas.microanalyticservice.system: supplementalProperties" in *SAS Micro Analytic Service: Programming and Administration Guide*.

Executing Content That Has Been Published to SAS Cloud Analytic Services Destinations

To execute content that has been published to SAS Cloud Analytic Services (CAS), use the CAS Model Publishing and Scoring action set. For example, the following code runs a model named Evaluate_Loans on the local CAS server.

```
/* Start a CAS session named _mmcas_. */
cas _mmcas_;

/* Create librefs for all existing caslibs so that they */
/* are visible in the SAS Studio Libraries tree.          */
caslib _all_ assign;

proc cas;
    /* Specify the session to use for the runModelLocal action. */
    session _mmcas_;

    /* Define the parameter list for the runModelLocal action. */
    /* Reload the destination model table ("targetCode")       */
    /* before you execute the decision.                        */
    destination_model_table = "targetCode";
    destination_model_lib = "public";

    destination_model = "Evaluate_Loans";

    dp_inputTable="hmeq_test";
    dp_inputCASLib="public";

    dp_outputTable="hmeq_test_dm";
    dp_outputCASLib="public";

    parmlist = {
        modelTable={
            name=destination_model_table,
            caslib=destination_model_lib
            },
        modelName=destination_model,
        inTable={
            name=dp_inputTable,
            caslib=dp_inputCASLib
        },
        outTable={
            name=dp_outputTable,
            caslib=dp_outputCASLib
        }
    };

    /* Load the modelPublishing action set. */
    loadactionset "modelPublishing";

    /* Run the model locally on the CAS server. */
    action runModelLocal submit result=r  status=rc / parmlist;
run;
quit;
```

If SAS Data Studio is licensed at your site, you can submit this code in SAS Data Studio. To open SAS Data Studio, click ≡ and select **Prepare Data**. For more information, see *SAS Data Studio: User's Guide*.

You can view additional examples of using this CAS action set to execute published content by viewing the test results that are generated by publishing validation tests. On the Test Results page for a rule set, click **Code** to display the code that was generated by SAS Intelligent Decisioning. For information about running publishing validation tests and viewing the results, see "Validate a Published Rule Set" on page 73.

For more information about CAS and the Model Publishing and Scoring action set, see the following documentation:

- *Getting Started with CASL Programming*
- *SAS Cloud Analytic Services: CASL Reference*
- “Model Publishing and Scoring Action Set” in *SAS Visual Analytics: Programming Guide*
- *SAS Cloud Analytic Services: User’s Guide*

Content Executed by Published Rule Sets

When you execute a published rule set, the version of the content that is executed depends on the publishing destination.

Destination Type	**Content That Is Executed by Published Rule Sets**
SAS Micro Analytic Service	Global variables are not locked. If a new version of a global variable is activated, the newly activated version is used by the published module. Lookup tables are embedded in decisions if the lookupStaticBinding configuration option is turned on. If this option is off, lookup tables are not embedded, and published modules use newly activated versions of lookup tables. For more information, see “sas.businessrules.lookupStaticBinding” in *SAS Intelligent Decisioning: Administrator’s Guide*.
SAS Cloud Analytic Services	The current values of global variables are included in generated code if the inlineGlobalVariableValues configuration option is turned on, and published modules do not use newly activated versions of global variables. If this option is turned off, the generated code uses a SAS format to retrieve the current value of the global variable when the published module is run. For more information, see “sas.businessrules.inlineGlobalVariableValues” in *SAS Intelligent Decisioning: Administrator’s Guide*. Lookup tables are locked if the lookupStaticBinding configuration option is turned on. If this option is off, lookup tables are not locked, and published modules use newly activated versions of lookup tables. For more information, see “sas.businessrules.lookupStaticBinding” in *SAS Intelligent Decisioning: Administrator’s Guide*.
Teradata, Hadoop, and Container Destinations	All of the published rule set’s content, including custom functions, lookup tables, and global variables is included inline in the published module. Updates to the objects used in the rule set are not used by the published module.

Importing and Exporting Rule Sets

You can import rule sets from and export rule sets to comma-delimited (CSV) files. The format of the CSV file is the same format that is used by the %DCM_IMPORT_RULESET macro. For more information, see "Format of Rule Set CSV Input File" in *SAS Intelligent Decisioning: Macro Guide*.

IMPORTANT The import and export features in SAS Intelligent Decisioning are for modifying an existing rule set. If you do not need to edit content as a CSV file, use either the SAS Viya Command-Line (sas-viya) or SAS Environment Manager to import and export content. The SAS Viya CLI and SAS Environment Manager enable you to import new objects. For information about the CLI, see *SAS Viya Platform: Content Migration from SAS Viya 4*. For information about using SAS Environment Manager, see "Content Page" in *SAS Environment Manager: User's Guide* and "Import Page" in *SAS Environment Manager: User's Guide*.

Import a Rule Set

IMPORTANT If you import rules from a CSV file into a rule set that already contains rules, the existing rules are replaced with the rules that are defined in the CSV file. To avoid replacing existing rules, create a new empty rule set, and then import the contents of the CSV file into the empty rule set. To append rules to a rule set, you can export the rule set, add content to the CSV file, and then re-import the CSV file.

1. Open the rule set into which you want to import new rules.
2. Click **Import**. If the rule set already contains rules, asks you if you want to replace the existing rules.

   TIP If you do not have permission to update an object, SAS Intelligent Decisioning displays `(Read-Only)` in the title bar next to the object name, and the buttons for modifying and saving the object are disabled.

3. Click 🗀, and select the CSV file that you want to import.
4. Select the encoding of the CSV file, and click **Import**.

Export a Rule Set

IMPORTANT Do not modify the file structure or the header row in the exported CSV file. You can modify the data values.

To export a rule set, open the rule set and click **Export**. The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

3

Working with Treatments and Treatment Groups

About Treatments and Decisions **84**

Example: A Decision That Includes a Treatment Group **85**

About Attributes, Eligibility Rules, and Effective Dates **88**
- Attributes and Attribute Aliases 88
- Eligibility Rules 89
- Effective Dates 89

Arbitrating Treatments **89**

About Channels **90**

Define a Treatment **90**

Define a Treatment Group **91**

Activating a Treatment Group **93**
- What Does Activating a Treatment Group Do? 93
- Controlling Where Treatment Groups Are Activated and How They Are Used 94
- Activate a Treatment Group 94

Managing Attributes and Aliases **95**
- About Attributes and Aliases 95
- Define Custom Attributes 95
- Add Attributes from a Different Treatment 96
- Duplicate Attributes 96
- Delete Attributes from a Treatment 96
- View Attribute Aliases for a Treatment Group 97
- Remove an Attribute Alias from a Treatment Group 97
- Determine Which Treatments Use an Attribute 97

Delete Treatments from a Treatment Group **97**

Managing Eligibility Rules for a Treatment **97**
- Remove Eligibility Rule Set from a Treatment 98
- Change the Eligibility Rule Set for a Treatment 98

Copy a Treatment or Treatment Group URL **98**

Compare Treatment or Treatment Group Content 99

Compare Treatment Group Code 101

Generate PDF Documentation for a Treatment Group 101

Managing Treatments and Treatment Groups 102
Duplicate Treatments or Treatment Groups 102
Delete Treatments or Treatment Groups 103
Rename a Treatment or Treatment Group 103
Move Treatments or Treatment Groups to a Different Folder 103

Managing Versions of Treatments and Treatment Groups 104
Set the Displayed Version 104
Create a New Version 104
Copy the Content of a Version 105
Delete a Version 106

Determine Which Objects Use a Treatment or Treatment Group 106

Checking Out and Committing Treatment or Treatment Group Versions 107
About Checking Out and Committing Versions 107
Checking Out a Treatment Group from within a Decision 108
Checking Out and Committing Treatments from within a Treatment Group 108
Concurrently Checking Out and Committing a Treatment or Treatment Group Versions 109
Check Out and Commit a Treatment or Treatment Group Version 110
Determine Who Has a Version Checked Out 110
Opening the Original Object 111
Undoing a Check Out 111

Managing Version Tags for Treatment and Treatment Groups 111
Add a Version Tag 111
Remove a Version Tag 112
Modify a Version Tag 112

About Treatments and Decisions

In SAS Intelligent Decisioning, you use treatments to define offers that you might want to present to subjects that contact your company as a result of an inbound marketing campaign.

In an inbound marketing campaign, a subject initiates contact with a company through a specific channel such as email, a website, or a phone call. A subject can be an individual customer, a company account, a household, or other entity. Depending on the channel, an application or website captures the subject's behavior or a human representative enters the subject's information into a form or a customer service application.

The subject usually has an identification number (a subject ID). The customer service application or other calling application uses the subject ID to send a request to SAS Intelligent Decisioning that invokes a decision. The decision typically performs the following tasks:

1 Uses the subject ID to issue a web service call or to query the company database in order to retrieve additional customer data, such as name, address, and household income, if such information exists.

2 Determines the set of offers for which the subject qualifies.

3 Arbitrates the set of offers to determine which ones the subject is most likely to respond to.

4 Generates a response tracking code and subject contact history data. The subject contact history data typically includes the set of offers from step 3.

5 Returns to the calling application the response tracking code and the set of offers for which the subject qualifies.

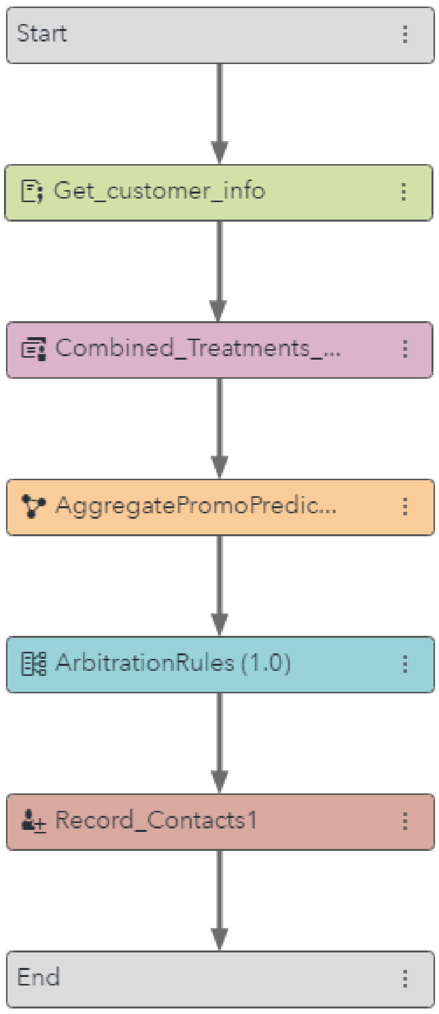
At this point, the calling application or a customer service representative can present the offers to the customer. The calling application can use the response tracking code to update the subject contact history data. For example, it can record which treatments are presented to the customer and the subject's response to the treatments.

"Example: A Decision That Includes a Treatment Group" on page 85 shows how to accomplish the tasks listed above by using a decision in SAS Intelligent Decisioning.

Example: A Decision That Includes a Treatment Group

Figure 3.1 shows a basic decision that defines offers, arbitrates these offers, and updates the subject contact history. This example contains only one treatment group and one record contacts node, but decisions can include multiple treatment groups and record contacts nodes.

Figure 3.1 Example Decision That Uses Treatments



This decision flow has the following elements:

1 The data query node named Get_customer_Info queries the company database and retrieves additional information about the subject based on the subject ID. For example, if the treatment is an offer to upgrade the subject's cell phone, the SQL query could retrieve the model number for the subject's current phone, if that information is available. For more information, see "Data Query Files" on page 159.

   Nodes that retrieve additional information about the subject are typical but not required.

2 The treatment group name Combined_Treatments_Final defines a set of treatments (offers) and includes rules that define who is eligible to receive the treatments. The decision uses the information about the subject and the eligibility rules to determine the treatments for which the subject qualifies.

   When you add a treatment group to a decision, SAS Intelligent Decisioning defines an output variable of type data grid. After the decision executes, this data grid contains a row for each treatment for which the subject qualifies.

Note: See "About Decision Variables and Mapping" on page 247 for information about when variables are created and how they are mapped.

You can include as many treatment groups in a decision as necessary. You can merge multiple output data grids into a single data grid if needed.

For more information, see the following topics:

- "About Attributes, Eligibility Rules, and Effective Dates" on page 88
- "About Channels" on page 90
- *SAS Intelligent Decisioning: Using Data Grids*

3 The model AggregatePromoPrediction and the rule set ArbitrationRules arbitrate the treatments. For example, the subject might qualify for several treatments, but you might want to offer them only the two treatments that they are most likely to accept. You might also want to return only the treatments that are appropriate for the channel.

- The model calculates a probability for each treatment (each row in the data grid) that indicates how likely the subject is to respond to the treatment.
- The rule set uses the DATAGRID_TOPN function to sort the treatment data grid based on the values of the probability column. The function returns the top two treatments to which the subject is most likely to respond. See "DATAGRID_TOPN Function" in *SAS Intelligent Decisioning: Using Data Grids* for more information.

You can use rule sets, scoring models, attribute values, or other methods to arbitrate treatments. For more information, see "Arbitrating Treatments" on page 89.

4 The record contacts node records information that you want to track, such as the treatments that are returned to the calling application and whether the record is used in generating aggregate reports for the channel. You can also specify variable values to track.

In real-time destinations (SAS Micro Analytic Service destinations), this node writes a record to the subject contact history. In all other destinations (SAS Cloud Analytic Services [CAS], Teradata, and Apache Hadoop), this node creates an output variable that contains the information that you specify that you want to track.

Record contacts nodes also create a response tracking code that the calling application can use to add additional information to the subject contact history.

TIP You can use record contacts nodes to track variable values even if you are not using treatments.

For more information, see the following topics:

- "Adding Record Contacts Nodes" on page 233
- Subject Contacts REST API

About Attributes, Eligibility Rules, and Effective Dates

A treatment defines the details of an offer that can be sent to a subject, who is eligible for the offer, and the dates during which the offer is available to be sent to a subject. A treatment is a set of attributes, eligibility rules, and effective dates.

Attributes and Attribute Aliases

Attributes are name-value pairs that define the specific details of the offer that you present to a subject. For example, for a treatment that gives subjects a 20% discount, you could define an attribute named DISCOUNT that has the value 20. To give subjects a special deal on a specific cell phone model, you can define an attribute named MODEL whose value is the specific model name. You could use treatment attributes to set values such as profitability, risk, cost, priority, or order. You can use the values of these attributes to arbitrate the treatments. For more information, see “Arbitrating Treatments” on page 89.

An attribute’s value can be dynamic or fixed.

Fixed
: You specify the value or set of values for fixed attributes when you define the attribute. You cannot customize the values in treatment groups, and the decision cannot set them at run time.

Dynamic
: You can set the value or set of values for a dynamic attribute when you add a treatment group to a decision, or the decision can set them at run-time. When you define a dynamic attribute, you can specify default values for the attribute. Within each treatment group in which the attribute appears, you can set static values that are used only within that specific treatment group, or you can specify that the attribute’s values are set by the decision at run time.

When you add a treatment group to a decision, SAS Intelligent Decisioning does the following:

- creates an output decision variable of type data grid that contains a column for each attribute in the treatment group.
- creates an input decision variable for each dynamic attribute whose value is set by the decision.

SAS Intelligent Decisioning automatically maps the treatment group’s attributes to the decision variables that have the same name and data type. For more information, see “Mapping Variables within a Decision” on page 247.

You can assign aliases to attributes. An alias is an alternative name that you can assign to a treatment attribute after you add the treatment to a treatment group. Aliases are useful when attributes in different treatments within the group represent the same data but are defined with different names. For example, treatment A could define an attribute named DISCOUNT, and treatment B could define an attribute named DISC. Assuming that these two attributes represent the same value, you can assign an alias to one or both of the attributes. An attribute can have only one alias.

Eligibility Rules

Eligibility rules define who is eligible to receive the offer defined by the treatment. For example, your campaign can target people who already have a specific credit card and who are at least 30 years old, or people who own a home but who do not have a home equity line of credit. You define eligibility rules in the eligibility rule set. An eligibility rule set is a filtering rule set. For more information, see "About Rules and Rule Set Types" on page 13.

Effective Dates

The effective dates for a treatment are the start and end dates when the treatment is active, that is, when it can be returned to a subject. Responses that are defined by a treatment are not returned to subjects outside of the effective dates. Effective dates are not required. A treatment that does not have effective dates is considered to always be active.

Note: The effective dates are always based on the time zone of the server where the decision is executed. The time zone for SAS Viya 4 servers is set to Coordinated Universal Time (UTC).

Arbitrating Treatments

You can determine which treatments a subject is most likely to respond to by using one or more of the following methods:

- use filtering rule sets to select only certain records for processing. Only the records whose conditions evaluate to true are processed by the remaining objects in the decision. For more information, see "About Rules and Rule Set Types" on page 13.

  Note: Filtering rule sets can be included directly in a decision, but they are also used as the eligibility rule set in treatments.

- use models to score treatments. For example, you might have a model that calculates propensity scores for individual treatments.
- use data grid functions to sort or subset the treatments according to the values of their attributes. For example, you could have attributes for value, profitability, or risk. For more information, see "Data Grid Functions" in *SAS Intelligent Decisioning: Using Data Grids*.

- use DS2 code files to create custom code for arbitrating the treatments. Chapter 6, “Using Custom Code Files,” on page 151.

About Channels

Channels are routes by which your company and a subject are in contact. Typical channels are email, phone call, and website. A default set of channels is defined in the Treatment Channels lookup table. Your administrator can customize this lookup table for your enterprise. You can use this lookup table in rules to narrow the list of offers to ones that are appropriate for the channel. See “Predefined Lookup Tables” on page 116 for more information.

You can create assignment rules to determine the channel through which your company and a subject are in contact if the channel was not specified in the original request.

Define a Treatment

Create a New Treatment and Define Attributes

1 Click , and then click **New Treatment**. The New Treatment window appears.

2 Enter a name for the treatment.

3 (Optional) Enter a description for the treatment. Descriptions are limited to 1000 characters.

> **TIP** You can edit the description later on the **Properties** tab.

4 Click , and select the folder where you want to save the treatment.

5 Click **Save** to save the treatment. SAS Intelligent Decisioning opens the new treatment and displays the **Attributes** tab.

> **TIP** Objects that are saved in a folder for which the check-out and commit feature is enabled, such as the `Decision Repository` folder, must be checked out before they can be edited.

6 (Optional) If the treatment is in a folder for which the check-out and commit feature is enabled, click the **Versions** tab and check out the latest version of the treatment. For more information, see “Check Out and Commit a Treatment or Treatment Group Version” on page 110.

7 (Optional) Define treatment attributes. A treatment can have zero or more attributes. For more information about attributes, see "About Attributes, Eligibility Rules, and Effective Dates" on page 88.

You can add new custom attributes, add attributes from other treatments, or duplicate attributes within the same treatment. For more information, see the following topics:

- "Define Custom Attributes" on page 95
- "Add Attributes from a Different Treatment" on page 96
- "Duplicate Attributes" on page 96

Specify Eligibility Rules

You specify eligibility rules in a filtering rule set. See "About Rules and Rule Set Types" on page 13 for more information.

8 On the **Eligibility Rule Set** tab, click **Add Rule Set**. The Select Rule Set window appears.

9 Select the filtering rule set that you want to use as the eligibility rule set for the treatment, and click **OK**.

10 At the top of the **Eligibility Rule Set** tab, select the version of the eligibility rule set that you want to use.

TIP You can view or edit an eligibility rule set from the **Eligibility Rule Set** tab by clicking **Open**.

(Optional) Specify Effective Dates

11 On the **Properties** tab, select the **Start date** and **End date** for the period in which the treatment can be sent to a subject. If you do not specify effective dates, the treatment is considered as always active. Specify the time based on the Coordinated Universal Time (UTC) time zone.

Note: The effective dates are always based on the time zone of the server where the decision is executed. See "Effective Dates" on page 89.

Define a Treatment Group

Create a New Treatment Group

1 Click ▤, and then click **New Treatment Group**. The New Treatment Group window appears.

2 Enter a name for the treatment group.

3 (Optional) Enter a description for the treatment group. Descriptions are limited to 1000 characters.

TIP You can edit the description later on the **Properties** tab.

4 Click 🗀, and select the folder where you want to save the treatment group.

5 Click **Save** to save the treatment group. SAS Intelligent Decisioning opens the new treatment group and displays the **Treatments** tab.

TIP Objects that are saved in a folder for which the check-out and commit feature is enabled, such as the `Decision Repository` folder, must be checked out before they can be edited.

6 (Optional) If the treatment group is in a folder for which the check-out and commit feature is enabled, click the **Versions** tab and check out the latest version of the treatment group. For more information, see "Check Out and Commit a Treatment or Treatment Group Version" on page 110.

Add Treatments

7 On the **Treatments** tab, click **Add Treatments**. The Add Treatments window appears.

8 Select the treatments that you want to add to the treatment group, and click **OK**.

TIP After you add a treatment to a treatment group, the variables in the eligibility rule set are added to the list of eligibility variables for the treatment group. You can view the eligibility variables for a treatment group on the **Eligibility Variables** tab.

9 (Optional) Select the version of each treatment that you want to use in the treatment group.

TIP You can change the version of a treatment at any time on the **Treatments** tab.

Customize Dynamic Treatment Attributes

For information about dynamic and fixed attributes, see "Define Custom Attributes" on page 95.

10 On the **Treatments** tab, click **Set Attributes**. The Set Attributes window appears. By default, dynamic attributes are automatically selected, and their values are set at run-time by the decision flow.

11 For each dynamic variable, verify that its setting is correct for the current treatment group.

- If you want the attribute's value to be set by the decision flow, leave the attribute selected.
- If you want to enter a static value or list of values for the attribute, clear the check box, and then enter the value or values that you want to be used in the current treatment group. To delete a value from a list, click the **X** for that value. To enter new values, press Enter after each new value.

Add Attribute Aliases

For information about aliases, see "Attributes and Attribute Aliases" on page 88.

12 Click the **Attributes** tab.

13 Click for the attribute to which you want to assign an alias. The Choose an Alias window appears and displays the list of existing attribute names that you can associate with the selected attribute.

- To assign an existing attribute name as an alias, select the attribute name in the list and click **OK**.
- To enter a new alias, click **New Alias**, enter the new alias name, and click **Save**. When you create a new alias, SAS Intelligent Decisioning creates a new output variable for the treatment group.

TIP In the Set Attributes window, you can view the alias that is assigned to an attribute by clicking .

Activating a Treatment Group

What Does Activating a Treatment Group Do?

Activating a version of a treatment group makes that version available for use by published decisions. Published decisions can use only activated versions.

In SAS Micro Analytic Service destinations, the sas.decisions.masInlineTreatmentGroup configuration option on page 94 controls what happens when you activate a treatment group. If this configuration option is set to On, the treatment group is converted into executable code in the form of a SAS Micro Analytic Service module. This module enables decisions that are published to SAS Micro Analytic Service destinations to use the most recently activated version of a treatment group. If this option is set to Off, a static version of the treatment group is included inline in the generated code for decisions.

CAS, Teradata, Hadoop, and container destinations do not support SAS Micro Analytic Service modules. Decisions that are published to these destinations always include a static version of each treatment group.

You must activate a version of any treatment group that is used in a decision. When you activate a version of a treatment group, that version is locked and cannot be edited. Each treatment group can have only one active version.

The active version of a treatment group is used in the following ways:

- when you run a test or scenario test prior to publishing an object.
- to generate a static copy of the treatment group when you publish a decision if you publish the decision to CAS, Teradata, Hadoop, or container destinations. Static copies are always generated for these destinations.

- by published decisions in SAS Micro Analytic Service destinations if the configuration sas.decisions.masInlineTreatmentGroup property is set to Off. See "Controlling Where Global Variables Are Activated and How They Are Used" on page 149 for more information.

Controlling Where Treatment Groups Are Activated and How They Are Used

Static versions of treatment groups are automatically embedded in content that is published to CAS, Teradata, Hadoop, and container destinations. Your administrator can use the following configuration options to control how treatment groups are used in SAS Micro Analytic Service destinations:

sas.treatmentdefinitions.activation.destinations = *destinations*
: specifies the SAS Micro Analytic Service destinations in which treatment groups are activated. Only one SAS Micro Analytic Service destination is supported. By default, that destination is named `maslocal`.

sas.decisions.masInlineTreatmentGroup = On | Off
: specifies whether the code for treatment groups is included inline in generated code when you publish a decision to SAS Micro Analytic Service destinations.

  When this property is set to On, a static copy of each treatment group is included in the code that is generated for SAS Micro Analytic Service destinations. Changes to the treatment groups do not affect published objects. When this property is set to Off, the generated code uses a SAS Micro Analytic Service module to retrieve the active version of the treatment group when the decision is run.

Activate a Treatment Group

Activating a version of a treatment group version locks that version. Locked groups cannot be unlocked.

You must activate a treatment group version before you can publish an object that uses that version. By default, when you activate a treatment group, it is activated in all of the publishing destinations that are configured at your site. However, you adminstrator might specify that groups are activated only in specific SAS Micro Analytic Service destinations. For more information, see "Controlling Where Treatment Groups Are Activated and How They Are Used" on page 94.

To activate a treatment group:

1 Open the treatment group that you want to activate.

> **TIP** If you do not have permission to update an object, SAS Intelligent Decisioning displays **`(Read-Only)`** in the title bar next to the object name, and the buttons for modifying and saving the object are disabled.

2 On the **Versions** tab, select the version that you want to activate, and click **Activate**.

Managing Attributes and Aliases

About Attributes and Aliases

For additional information about attributes and aliases, see the following topics:

- "Attributes and Attribute Aliases" on page 88
- "Add Attribute Aliases" on page 93
- "Customize Dynamic Treatment Attributes" on page 92

Define Custom Attributes

1 On the **Attributes** tab of a treatment, click **Add Attribute** and select **Custom attribute**. The Add Attribute window appears.

> **TIP** If you do not have permission to update an object, SAS Intelligent Decisioning displays `(Read-Only)` in the title bar next to the object name, and the buttons for modifying and saving the object are disabled.

2 Complete these steps for each attribute:

   a Enter a name for the attribute.

   b (Optional) Enter a description for the attribute.

   c Select the data type for the attribute.

   d Select **Fixed** or **Dynamic**, depending on how the attribute's value will be set when it is included in a treatment group.

   Dynamic
   The values of dynamic attributes can be set by the decision at run-time. When you define a dynamic attribute, you can specify a default value or a list of default values for the attribute. Within each treatment group in which the attribute appears, you can specify that the attribute's value is set by the decision at run time, or you can set a static value that is used only within that specific treatment group.

   Fixed
   You define the value or list of values for fixed attributes when you define the attribute. The values cannot be customized in treatment groups.

  e. Specify the value or the list of values for the attribute. For dynamic attributes, these values are used as the default values if the attribute's values are not customized in treatment groups.

     For attributes that are character, integer, or decimal attributes, you can specify a single value or a list of values. Select the **List of values** check box to specify a list of possible values for the attribute. In the **Value** field, press Enter after each specific value.

     For attributes of other data types, you can specify only a single value.

  f. Click **Add** to add the attribute to the table.

3. Click **OK** to save the attributes.

Add Attributes from a Different Treatment

1. On the **Attributes** tab of a treatment, click **Add Attribute**, and select **Treatment**. The Choose an Item window appears.
2. Select the treatment from which you want to copy an attribute, and click **OK**. The Add Attributes window appears.
3. Click +» to select all of the attributes, or select one or more individual attributes and click +›.
4. Click **Add** to add the attributes, or click **Add and replace** to replace existing attributes that have the same name.

Duplicate Attributes

1. On the **Attributes** tab of a treatment, select the attribute that you want to duplicate.
2. Click ⋮ , and select **Duplicate**. The Duplicate Attributes window appears.
3. Enter a new name for the duplicate attribute, and click **Duplicate**.

Delete Attributes from a Treatment

On the **Attributes** tab of a treatment, select the attribute that you want to delete, click ⋮ , and select **Delete**.

TIP If you do not have permission to update an object, SAS Intelligent Decisioning displays `(Read-Only)` in the title bar next to the object name, and the buttons for modifying and saving the object are disabled.

View Attribute Aliases for a Treatment Group

You can view attribute aliases in two ways:

- Click the **Attribute Aliases** tab in a treatment group.
- On the **Treatments** tab, click **Set Attributes**, and then click ☰ for a specific attribute.

Remove an Attribute Alias from a Treatment Group

On the **Attribute Aliases** tab of a treatment group, select the attributes whose aliases you want to remove, click ⋮ , and select **Remove alias**.

Determine Which Treatments Use an Attribute

On the **Attribute Aliases** tab of a treatment group, select the attribute for which you want treatment information, click ⋮ , and select **Used by**. SAS Intelligent Decisioning opens the Used By Treatments window, which lists the treatments that use the attribute.

Delete Treatments from a Treatment Group

On the **Treatments** tab of the treatment group, select the treatments that you want to remove from the group, click ⋮ , and select **Delete**.

Managing Eligibility Rules for a Treatment

For additional information about eligibility rules, see "About Attributes, Eligibility Rules, and Effective Dates" on page 88 and "Specify Eligibility Rules" on page 91.

Remove Eligibility Rule Set from a Treatment

On the **Eligibility Rule Set** tab of a treatment, click **Remove**.

Change the Eligibility Rule Set for a Treatment

1 On the **Eligibility Rule Set** tab of a treatment, click **Replace**. The Select Rule Set window appears.

2 Select the filtering rule set that you want to use as the eligibility rule set for the treatment, and click **OK**.

3 At the top of the **Eligibility Rule Set** tab, select the version of the eligibility rule set that you want to use.

   TIP You can view or edit an eligibility rule set on the **Eligibility Rule Set** tab by clicking **Open**.

Copy a Treatment or Treatment Group URL

To create a link for external documentation that automatically opens a treatment or treatment group in SAS Intelligent Decisioning:

1 Open the treatment or treatment group.

2 Click ⋮, and select either **Copy treatment URL** or **Copy treatment group URL**. The Copy URL window appears, and the URL is automatically selected.

3 Click **Copy**, and then click **Close**.

   Paste the link into your documentation.

Compare Treatment or Treatment Group Content

You can compare the contents of two different treatments or treatment groups, or you can compare the contents of two different versions of the same treatment or treatment group.

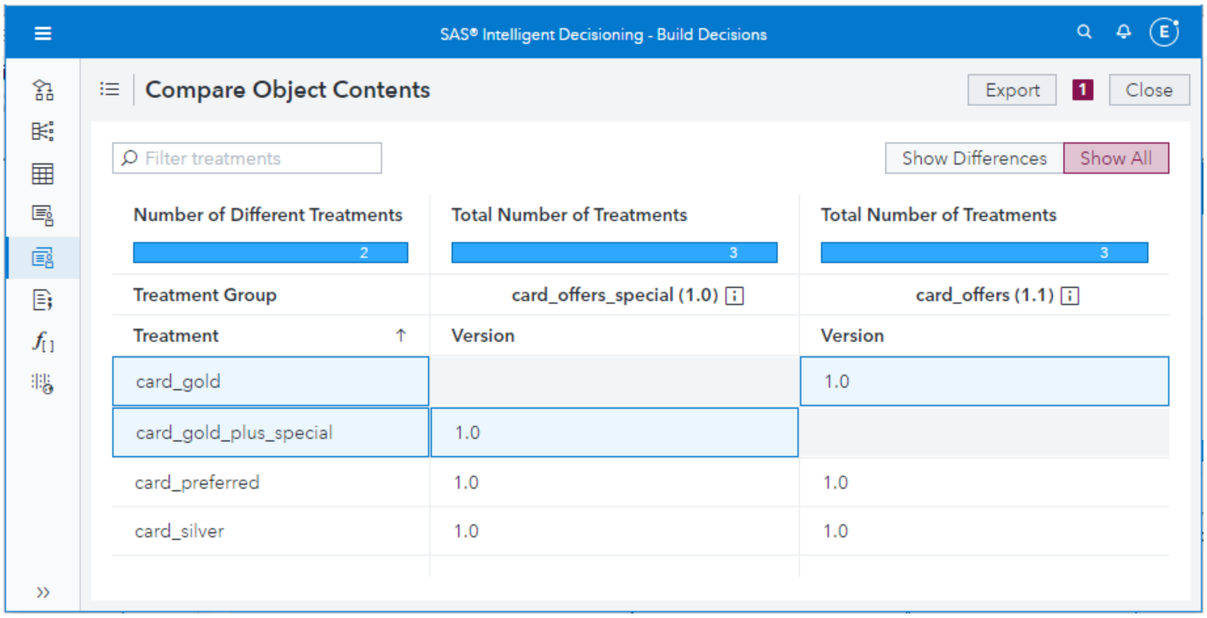
1 Select the objects that you want to compare.

- To compare the contents of two different treatments or treatment groups, select the treatments or treatment groups in the category view, click ⋮ , and select **Compare object contents**.
- To compare the contents of two versions of the same treatment or treatment group, open the treatment or treatment group, click ⋮ on the **Versions** tab, and select **Compare object contents**.

The Select Versions window appears.

2 Select the versions that you want to compare, and click **Compare**. SAS Intelligent Decisioning opens the two objects side-by-side in the Compare Object Contents window.

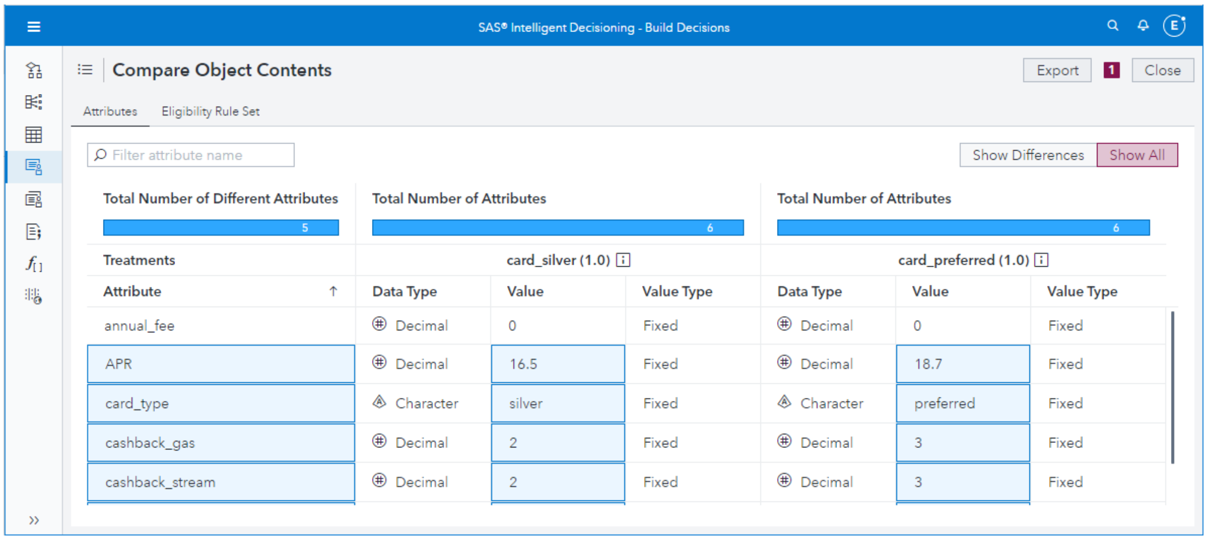
For treatment groups, the Compare Object Contents window displays the name and version number of the treatments that differ in each object. To display all of the treatments in each group, click **Show All**. For example, in the following figure, both treatment groups contain the card_preferred treatment and the card_silver treatment. The card_gold treatment appears only in the card_offers group, and the card_gold_plus_special treatment appears only in the card_offers_special treatment group.



Number of Different Treatments	Total Number of Treatments	Total Number of Treatments
2	3	3
Treatment Group	card_offers_special (1.0)	card_offers (1.1)
Treatment	Version	Version
card_gold		1.0
card_gold_plus_special	1.0	
card_preferred	1.0	1.0
card_silver	1.0	1.0

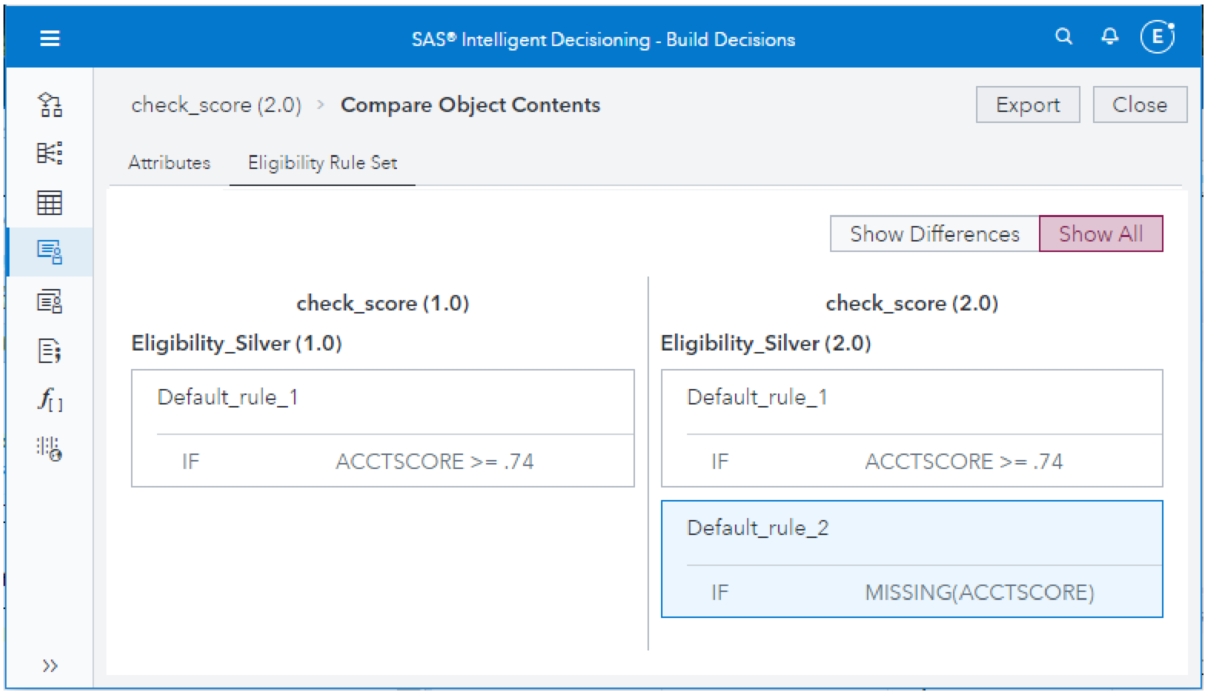
TIP Click [i] beside a treatment name or treatment group name to display its location.

For treatments, the **Attributes** tab displays the attribute names and values that are different between the two objects. For example, in the following figure, the two treatments contain the same attributes, but the values of all of the attributes are different except for the annual_fee attribute.



Total Number of Different Attributes: 5	Total Number of Attributes: 6			Total Number of Attributes: 6		
Treatments	card_silver (1.0)			card_preferred (1.0)		
Attribute	Data Type	Value	Value Type	Data Type	Value	Value Type
annual_fee	Decimal	0	Fixed	Decimal	0	Fixed
APR	Decimal	16.5	Fixed	Decimal	18.7	Fixed
card_type	Character	silver	Fixed	Character	preferred	Fixed
cashback_gas	Decimal	2	Fixed	Decimal	3	Fixed
cashback_stream	Decimal	2	Fixed	Decimal	3	Fixed

3 (Optional) Click the **Eligibility Rule Set** tab to display the differences in the eligibility rule sets. For example, in the following figure, the two version of the check_score treatment use different versions of the same eligibility rule set. Version 2.0 of the rule set contains an additional rule.


SAS® Intelligent Decisioning - Build Decisions
check_score (2.0) > Compare Object Contents
Export
Close
Attributes
Eligibility Rule Set
Show Differences
Show All

check_score (1.0)	check_score (2.0)
Eligibility_Silver (1.0)	Eligibility_Silver (2.0)
Default_rule_1: IF ACCTSCORE >= .74	Default_rule_1: IF ACCTSCORE >= .74
	Default_rule_2: IF MISSING(ACCTSCORE)

4 (Optional) Click **Export** to export the results of the comparison to a PDF file. The Export Comparison Results window appears.

5. (Optional) Select the information that you want to export, and click **Export**. If you want the PDF file to display only the differences between the two objects, select **Show differences**. To display all of the objects' information, select **Show all**.

   The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Compare Treatment Group Code

You can compare the generated code of two different treatment groups, or you can compare the generated code of two different versions of the same treatment group.

1. Select the objects that you want to compare.
   - To compare the generated code of two different treatment groups, select the groups in the category view, click ⋮, and select **Compare code**.
   - To compare the generated code of two versions of the same treatment group, open the treatment group, click ⋮ on the **Versions** tab, and select **Compare code**.

   The Select Versions window appears.
2. Select the versions that you want to compare, and click **Compare**. SAS Intelligent Decisioning opens the two objects side-by-side in the Compare Code window and highlights the differences.
3. (Optional) Click **Export** to export the results of the comparison to a PDF file. The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Generate PDF Documentation for a Treatment Group

You can generate detailed documentation for a treatment group as a PDF document. The PDF includes the treatment group properties, and details about each treatment that is included in the group. Treatment details include treatment properties, the name of the eligibility rule set, and details about each of the treatment's attributes.

1. Open the treatment group.
2. Click ⋮, and select **Create document**. The Create Document window appears.

TIP If this option is disabled, there might be unsaved changes. Click 🖫.

3 (Optional) Enter a name for the document if you do not want to use the default name.

4 Click **Create**. SAS Intelligent Decisioning creates the PDF. The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Managing Treatments and Treatment Groups

Duplicate Treatments or Treatment Groups

Note: You cannot duplicate a treatment or treatment group if it is open.

To duplicate a single treatment or treatment group:

1 In the Treatments or Treatment Groups category view, select the treatment or treatment group that you want to duplicate.

2 Click ⋮ and select **Duplicate**. The Duplicate Treatments or Duplicate Treatment Groups window appears.

3 Enter a new name for the duplicate treatment or treatment group.

4 (Optional) Enter a description for the treatment or treatment group.

5 Select the version of the treatment or treatment group that you want to duplicate.

6 Click 🗀 and select the location where you want to save the duplicate treatment or treatment group.

7 Click **Duplicate**.

To duplicate multiple treatments or treatment groups:

1 In the Treatments or Treatment Groups category view, select the treatments or treatment groups that you want to duplicate.

2 Click ⋮ and select **Duplicate**. SAS Intelligent Decisioning duplicates the latest version of the treatments or treatment groups and appends `_Copy` to the names of the duplicate copies. If needed, a number is also appended to the names of the duplicate copies.

Note: When you duplicate a treatment or treatment group, SAS Intelligent Decisioning creates a relationship between the original treatment or treatment group and the duplicate object. If either object is changed, and you later copy the contents of one object into the other, SAS Intelligent Decisioning displays a warning message. Verify that you want to replace the current contents of the treatment or treatment group before you paste the new content.

Delete Treatments or Treatment Groups

Note: You cannot delete a treatment or treatment group if it is open.

In the Treatments or Treatment Groups category view, select the treatments or treatment groups that you want to delete, click ⋮, and select **Delete**. SAS Intelligent Decisioning moves the deleted items to the recycle bin. For more information, see "Manage Folders and Folder Content" on page 9.

Rename a Treatment or Treatment Group

Note: You cannot rename a treatment or treatment group if it is open.

1 In the Treatments or Treatment Groups category view, select the treatment or treatment group that you want to rename.

2 Click ⋮, and select **Rename**.

3 Enter a new name for the treatment or treatment group, and click **Rename**.

Move Treatments or Treatment Groups to a Different Folder

1 In the Treatments or Treatment Groups category view, select the treatments or treatment groups that you want to move.

2 Click ⋮ and select **Move**. The Choose a Location window appears.

3 Select the location to which you want to move the treatments or treatment groups, and click **OK**.

Managing Versions of Treatments and Treatment Groups

Set the Displayed Version

The displayed version is the version whose information is displayed on the other tabs, such as the **Properties** and **Attributes** tabs. On the **Versions** tab, a ✓ in the **Displayed Version** column indicates the displayed version. To change the displayed version, click the version number for the version that you want to view. The displayed version is shown in the title bar.

Create a New Version

Note: For objects that are stored in locations for which the check-out and commit feature is enabled, you cannot manually create a new version. The only way to create a new version is to check out an existing version and commit a new version. For information, see "Check Out and Commit a Treatment or Treatment Group Version" on page 110.

Note: The current version of an object is the version with the highest version number. When you create a new version, SAS Intelligent Decisioning locks the current version before it creates the new version.

IMPORTANT You cannot unlock a locked version. You cannot save changes to a version that is locked. If you modify a version that is locked and click 🖫, SAS Intelligent Decisioning asks you if you want to replace the current unlocked version with your edited version.

To manually create a new version:

1 On the **Versions** tab, click the version number for the existing version that you want to use as the basis for the new version.

2 Click **New Version**. The New Version window appears.

3 Select the version type: **Minor** or **Major**. Version numbers follow the format *major.minor*. If you select **Major**, the number to the left of the period is incremented. If you select **Minor**, the number to the right of the period is incremented.

4 (Optional) For each version tag that you want to add, either enter a custom tag or begin typing and select a tag from the list of previously entered tags. Press Enter after each tag.

Note: A tag is limited to 100 characters.

> **TIP** All tags that have been previously entered for any object are saved in the list that is displayed when you start typing. You cannot delete tags from this list.

5 (Optional) Enter information about the new version in the **Notes** field.

> **TIP** You can edit these notes at any time on the **Versions** tab.

6 Click **Save**.

Copy the Content of a Version

You can copy the content of an object's version in the category view or on the **Version** tab for the object.

1 In the category view, complete these steps:

a Select the treatment or treatment group whose contents you want to copy.

b Click ⋮ , and select **Copy version**. The Copy Version window appears.

c Select the version whose contents you want to copy.

Alternatively, on the **Versions** tab of the treatment or treatment group whose contents you want to copy:

a Select the version whose contents you want to copy.

b Click ⋮ , and select **Copy version**. The Copy Version window appears.

2 Click 🗀, and select the target treatment or treatment group into which you want to paste the contents of the version. You can paste version contents only into an object of the same type.

When you paste the contents, SAS Intelligent Decisioning creates a new version of the target treatment or treatment group. The target object contains only the pasted content.

3 Select whether you want to create a new major or minor version.

4 (Optional) Modify the notes that will be associated with new version.

5 (Optional) Add tags that will be associated with the new version. Tags that are associated with a source object version are not automatically added to the new version. See "Add a Version Tag" on page 111

6 Click **Paste Version**, and then click **Yes**.

Note: When you copy the contents of a source object into a target object, SAS Intelligent Decisioning creates a relationship between the two objects. If the source object is modified after you copy its contents, and you later copy the contents of the target object back into the source object, SAS Intelligent Decisioning displays a warning message. Verify that you want to replace the current contents of the source object before you paste the new content.

Delete a Version

IMPORTANT When you delete a specific version, that version is deleted permanently. It is not moved into the recycle bin, and it cannot be restored.

Note: In order to be able to delete a specific version of an object, you must have permission to delete the object itself. Also, the configuration option sas.treatmentdefinitions.deleteVersions must be turned on.

On the **Versions** tab, select the version that you want to delete, click ⋮, and select **Delete**.

You cannot delete the current version.

Determine Which Objects Use a Treatment or Treatment Group

To list the objects that use a specific treatment or treatment group:

1 On the **Treatments** or **Treatment groups** category page, select the check box for the treatment or treatment group, click ⋮, and select **View used by report**. The All Objects that Use the Selected Item window appears. This window lists all objects that use any version of the selected treatment or treatment group.

2 (Optional) Select a specific version of the treatment or treatment group. SAS Intelligent Decisioning narrows the list to include only the objects that use the selected version of the treatment or treatment group.

Note: The **View used by report** option is also available from within an open treatment or treatment group.

In the report, you can use the **Filter** field to filter the list of objects based on the object names. Click on an object name to open the object. Click ⓘ next to an object name to display the date on which the object was last modified and the ID of the user who modified the object. For decisions and code files that have been checked out, SAS Intelligent Decisioning also displays the IDs of the users that have checked out the objects. For decisions, SAS Intelligent Decisioning includes the decision’s workflow status.

Click **Export** to export the Used By report to a PDF file. In the PDF file, you can click on an object name to open the object in a new browser tab.

Checking Out and Committing Treatment or Treatment Group Versions

About Checking Out and Committing Versions

Your administrator can enable the check-out and commit feature for treatments and treatment groups that are in any folder by specifying the folder in the sas.treatmentdefinitions.checkout.checkoutEnabledFolderPaths configuration option. Enabling this feature for a folder does not automatically modify the permissions for the folder or for the objects in it. You can still modify a treatment or a treatment group in the folder without checking it out, but you are expected to check out the latest version before you edit it. However, your administrator might also set permissions that require you to check out treatments and treatment groups in these folders before you can edit them. For more information, see “sas.treatmentdefinitions.checkout” in *SAS Intelligent Decisioning: Administrator’s Guide* and “Set Permissions for Check-Out Folders” in *SAS Intelligent Decisioning: Administrator’s Guide*.

By default, SAS Intelligent Decisioning defines a folder where you can store treatments and treatment groups that must be checked out before they can be edited. This folder is the `Decision Repository` folder, and it is the default value for the sas.treatmentdefinitions.checkout.checkoutEnabledFolderPaths configuration option. The default permissions for this folder require that non-administrative users check out a version and commit their changes to the checked-out version. Users who do not have administrative permissions cannot edit the treatments or treatment groups in `Decision Repository` without first checking them out.

If a version can be or must be checked out before it is modified, the **Check Out** button appears at the top of the **Versions** tab for that object. You can check out any version of an object. You can check out only one copy of a version at a time.

TIP If the sas.treatmentdefinitions.checkout.allowConcurrentCheckout option is turned off, and a user has checked out a treatment version or treatment group version, the **Check Out** button for that treatment or treatment group is disabled for all other users. For more information, see “Concurrently Checking Out and Committing a Treatment or Treatment Group Versions” on page 109.

When you check out a version, SAS Intelligent Decisioning writes a working copy of the version into your `My Folder` folder and opens the working copy. SAS Intelligent Decisioning adds "`(Checked Out)`" to the name that is displayed at the top of the window.

While you have a version checked out, the Treatments or Treatment Group category view shows two treatments or treatment groups with the same name, but the folders listed in the **Location** column differ for each object. The original version is in the location specified by the sas.treatmentdefinitions.checkout.checkoutEnabledFolderPaths configuration option, and the checked-out copy is in your `My Folder` folder.

TIP If an object that you have checked out does not appear in the category view, click ↻ to refresh the category view.

Note: When a decision that uses a treatment group is deployed to a production environment, the decision always uses the active version of a treatment group. When you edit a decision that uses a treatment group that you have checked out, the properties panel for the treatment group shows the location of the treatment group in the original folder. The panel does not list the location of the treatment group that is checked out because that copy is not the active version.

A **Commit** button appears at the top of the **Versions** tab for the checked-out version. When you are finished editing the checked-out version, you must commit your changes in order for other users to be able to see them. When you commit your changes, SAS Intelligent Decisioning creates a new version with your changes.

If the parent object is deleted before you commit your changes, you will not be able to commit your changes.

You cannot activate the checked-out version that is in `My Folder`. To activate a version with your changes, you must commit your changes, and activate the committed version.

Checking Out a Treatment Group from within a Decision

See "Checking Out and Committing Objects from within A Decision" on page 274 for information.

Checking Out and Committing Treatments from within a Treatment Group

You can check out a treatment from within a treatment group if the following conditions are true:

- The treatment group is checked out.

- The treatment is stored in a folder for which the check-out feature is enabled.

To check out or commit an individual treatment from within a checked-out treatment group,select the check box for the treatment on the **Treatments** tab, click ⋮ , and select **Check out** or **Commit**. SAS Intelligent Decisioning displays an asterisk next to the name of any treatment that is checked out from within the treatment group.

When you check out a treatment, the treatment group is modified to use the checked-out copy of the treatment, and SAS Intelligent Decisioning enables the **Commit** option in the menu.. When you commit a treatment group, all checked-out treatments that are used in the treatment group are also committed.

To cancel the check out of a treatment, select the check box for the treatment on the **Treatments** tab, click ⋮ , and select **Cancel check out**.

Concurrently Checking Out and Committing a Treatment or Treatment Group Versions

The ability for multiple users to check out the same treatment or treatment group at the same time is controlled by the sas.treatmentdefinitions.checkout.allowConcurrentCheckout configuration option. This option is turned on by default.

When this option is turned on, different users can check out the same version of the same object at the same time. Because the objects that are checked out are saved in each user's `My Folder` location, the default permissions allow individual users to see only the copies that they have checked out.

When this option is turned off and a user has checked out an object, the **Check Out** button for that object is disabled for all other users.

If multiple users check out the same version of the same object at the same time, each user's changes are preserved in a new version when they commit their changes. One user's changes do not overwrite another user's changes.

IMPORTANT If two users attempt to commit changes to the same object simultaneously, the first user's attempt will succeed but the second user might see an error message that the commit has failed. If the second user subsequently commits their changes, the **Modified By** column on the **Versions** tab for both the version committed by the first user and the version committed by the second user displays the user ID of the second user.

Check Out and Commit a Treatment or Treatment Group Version

TIP If a checked-out treatment group includes treatments that are also checked out, committing the treatment group also commits the treatments.

1 On the **Versions** tab, click **Check Out**.

SAS Intelligent Decisioning updates the **Properties** tab to indicate that the version is checked out.

2 Modify the checked-out version as needed, and save it.

TIP To discard the changes and delete the checked-out version from `My Folder`, you can commit the object without saving it first. However, committing the object without saving creates a new version of the object whose contents match the contents of the previous version. For information on undoing a check out, see “Undoing a Check Out” on page 111.

3 On the **Versions** tab, click **Commit**. The Commit Treatment Version or Commit Treatment Group Version window appears.

4 Select the version type: **Minor** or **Major**. Version numbers follow the format *major.minor*. If you select **Major**, the number to the left of the period is incremented, and the minor number is reset to zero. If you select **Minor**, the number to the right of the period is incremented.

5 (Optional) In the **Version tags** field, enter any version tags that you want to associate with the new version. Press Enter after each tag. Tags are limited to 100 characters each and cannot contain the following characters: $ ' " # & ? ( ) \ /

6 (Optional) Enter information about the new version in the **Notes** field.

7 Click **Commit**. SAS Intelligent Decisioning creates a new version with your changes, and deletes the working copy from your **`My Folder`** folder.

Determine Who Has a Version Checked Out

If the current version of an object is checked out, the IDs of the users that checked it out and the timestamps when each user checked it out appear in the **Checked out by** field on the **Properties** tab for the original object. You can also display this information by clicking ⧉ beside the version number on the **Versions** tab.

Opening the Original Object

When you check out an object, SAS Intelligent Decisioning adds the field **Original object link** to the **Properties** tab for the checked-out object. This field contains a link to the original object that was checked out. You can use this link to verify that you have checked out the correct version and to compare the original content with the modified content in the checked-out version.

Undoing a Check Out

How you undo the checkout of an object depends on how the object was checked out.

If both an object and a decision that uses the object are checked out at the same time, or if you checked out the object from within the decision, click ⋮ on the object's node in the decision diagram, and select **Cancel checkout**.

You can discard a checked-out version and any changes that you made by deleting the working copy of the version from your `My Folder` folder if the following conditions are true:

- You have not checked out a decision that uses the object.
- The object was not checked out at the same time as a decision that uses the object, or the object was not checked out from within the decision after the decision was checked out.

The deleted version is moved to the recycle bin. See "Delete Treatments or Treatment Groups" on page 103.

Managing Version Tags for Treatment and Treatment Groups

Add a Version Tag

Version tags enable you to better organize and group your content. Version tags are associated with specific versions of an object and not with the entire object. You can add the same tag to any version of any object. To add a tag to a treatment or treatment group version:

1 On the **Versions** tab, position your cursor in the **Version Tags** column for the version that you want to tag. If the version is not tagged, + appears. If the version has at least one tag, ✎ appears.

2 Click + to open the New Version Tags window, or click ✎ to open the Edit Version Tags window.

3 For each tag that you want to add, either enter a custom tag or begin typing and select a tag from the list of previously entered tags. Press Enter after each tag. Tags are limited to 100 characters each and cannot contain the following characters: $ ' " # & ? ( ) \ /

> TIP All tags that have been previously entered for any object are saved in the list that is displayed when you start typing. You cannot delete tags from this list.

> TIP To filter the version list based on a tag, right-click on the **Version Tags** column heading, enter a filter string in the **Filter** box, and press Enter. To clear the filter, delete the string from the **Filter** box, and press Enter.

4 Click **Close** to close the window.

Remove a Version Tag

1 On the **Versions** tab, position your cursor in the **Version Tags** column for the version whose tag you want to remove, and click ✎.

2 Click × beside the tag that you want to remove.

3 Click **OK** to close the window. The tag remains in the list of previously entered tags that is displayed when you add a tag, but the tag is no longer associated with version.

Modify a Version Tag

You cannot modify a version tag that already exists. To change the content of an existing tag, delete the tag as described in "Remove a Version Tag" on page 112, and then add the tag again as described in "Add a Version Tag" on page 111.

4

Working with Lookup Tables and Functions

About Lookup Tables and Functions 114

About Labels for Lookup Keys and Values 115

Predefined Lookup Tables 116

Create a New Lookup Table 116

Importing and Exporting Lookup Tables 117
- About Importing, Exporting, and Activating Lookup Tables 117
- Format of the Lookup CSV Import File 117
- Import Entries or Refresh Lookup Tables from CSV Files 117
- Export a Lookup Table to a CSV File 117

Add Lookup Table Entries 119

Edit Lookup Table Entries 120

Delete Lookup Table Entries 121

Edit Lookup Key or Value Labels 121

Copy a Lookup Table URL 121

Compare Lookup Table Content 121

Generate PDF Documentation for a Lookup Table 123

Managing Lookup Tables 123
- Duplicating Lookup Tables 123
- Delete Lookup Tables 123
- Rename a Lookup Table 123
- Move Lookup Tables 123

Managing Versions of Lookup Tables 125
- Set the Displayed Version 125
- Create a New Version 125
- Copy the Content of a Version 125
- Delete a Version 125

Determine Which Objects Use a Lookup Table 128

Checking Out and Committing Lookup Table Versions **128**
About Checking Out and Committing Versions 128
Checking Out and Committing a Lookup Table from within a Rule Set 128
Committing a Lookup Table Together with a Rule Set 128
Concurrently Checking Out and Committing Lookup Table Versions 128
Check Out and Commit a Lookup Table Version 128
Determine Who Has a Version Checked Out 128
Opening the Original Object 128
Undoing a Check Out 128

Managing Version Tags for Lookup Tables **133**
Add a Version Tag 133
Remove a Version Tag 133
Modify a Version Tag 133

Activating a Lookup Table **134**
What Does Activating a Lookup Table Do? 134
Controlling Where Lookup Tables Are Activated And How They Are Used 134
Activate a Lookup Table 134

Dictionary **137**
LOOKUP Function 137
LOOKUPVALUE Function 138

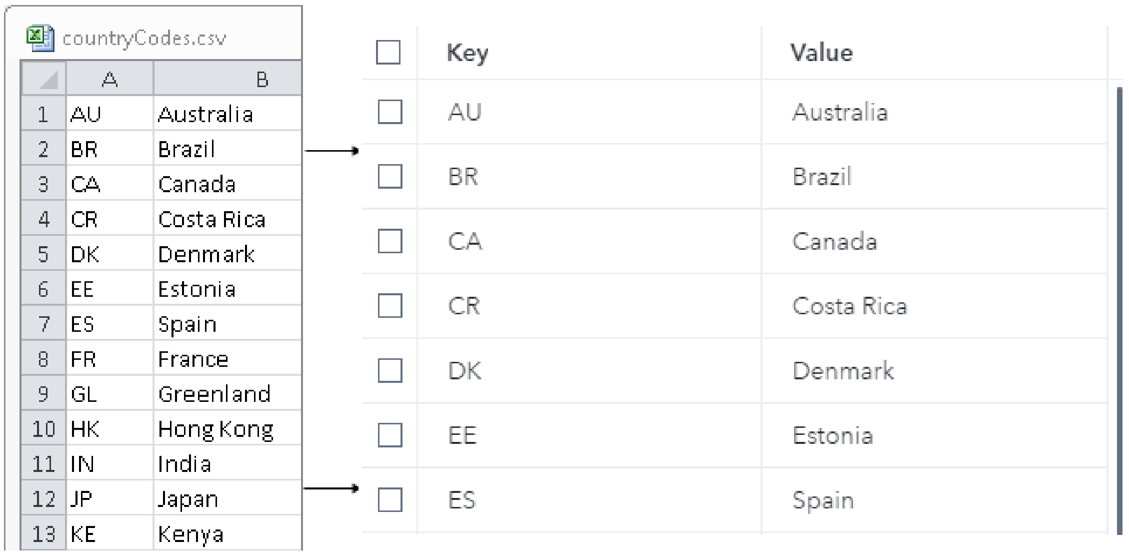
About Lookup Tables and Functions

SAS Intelligent Decisioning provides the ability to import lookup tables and reference them from rules. Lookup tables are tables of key-value pairs. For example, you can use a lookup table to retrieve a part name based on the code number of the part or to retrieve the full name for a country based on its abbreviation.

You can import lookup data from comma-separated-values (CSV) files, such as those created by spreadsheet applications, into lookup tables in SAS Intelligent Decisioning. You can re-import updated CSV files as needed to refresh the lookup tables.

Note: SAS Intelligent Decisioning does not support CSV files that contain signature lines.

Figure 4.1 CSV File Imported Into SAS Intelligent Decisioning



countryCodes.csv

	A	B
1	AU	Australia
2	BR	Brazil
3	CA	Canada
4	CR	Costa Rica
5	DK	Denmark
6	EE	Estonia
7	ES	Spain
8	FR	France
9	GL	Greenland
10	HK	Hong Kong
11	IN	India
12	JP	Japan
13	KE	Kenya

☐	Key	Value
☐	AU	Australia
☐	BR	Brazil
☐	CA	Canada
☐	CR	Costa Rica
☐	DK	Denmark
☐	EE	Estonia
☐	ES	Spain

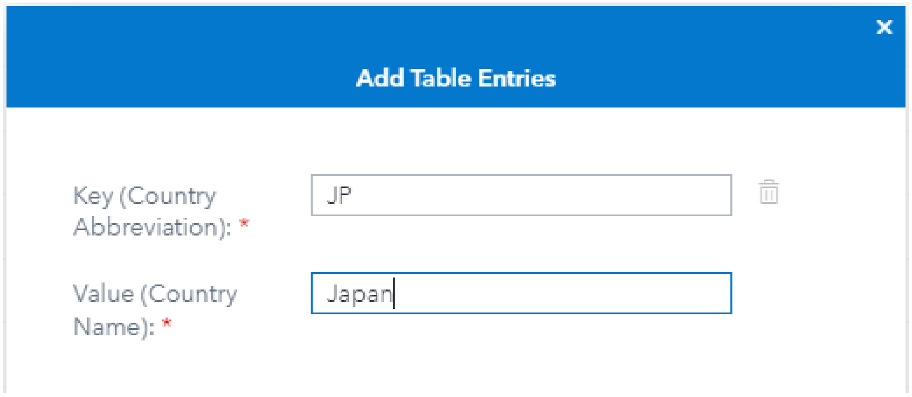
In a lookup table, each *lookup key* is associated with a *lookup value*. Lookup keys must be unique within each lookup table.

SAS Intelligent Decisioning provides two functions, LOOKUP and LOOKUPVALUE, that enable you to determine whether a lookup key exists in a lookup table and to retrieve a lookup value from a lookup table.

About Labels for Lookup Keys and Values

You can specify labels to use for lookup keys and lookup values. For example, for the lookup table shown in Figure 4.2 on page 115, you could specify `Country Abbreviation` for the key and `Country Name` for the value. You could specify the names of variables that are used to store the keys or values as the labels.

The labels that you enter are added to the **Properties** tab for the lookup table. They are also included in windows that contain fields where you enter keys or values. For example, if you add labels to the lookup table shown in Figure 4.2 on page 115 and open the Add Table Entries window, the labels are displayed in parentheses after the existing **Key** and **Value** field labels.



Key (Country Abbreviation): * JP

Value (Country Name): * Japan

Predefined Lookup Tables

SAS Intelligent Decisioning defines two lookup tables for you:

Treatment Channels
: defines several channels for use with treatments. Channels are the route by which your company and a subject are in contact. The predefined channels include ATM, Agent, Call Center, Mail, Phone, and Web.

Subject Level
: defines subject levels for use with treatments. A subject level is the type of account that is associated with a subject. For example, a subject might be an individual customer, a company account, or a household. The predefined levels are Account, Customer, and Household.

Your administrator can modify these tables as described in "Edit Lookup Table Entries" on page 120.

Create a New Lookup Table

1 Click ▦ to navigate to the Lookup Tables category view.

2 Click **New Lookup Table**. The New Lookup Table window appears.

3 Enter a name for the new lookup table. Lookup table names are limited to 250 characters. Lookup table names are case insensitive and must be unique within the database.

4 (Optional) Enter a label to use for the lookup key. Labels are limited to 100 characters. For more information, see "About Labels for Lookup Keys and Values" on page 115.

5 (Optional) Enter a label to use for the value. Labels are limited to 100 characters. For more information, see "About Labels for Lookup Keys and Values" on page 115.

6 (Optional) Enter a description for the new lookup table. Descriptions are limited to 1000 characters.

> **TIP** You can edit the description later on the **Properties** tab.

7 Click 🗀, and select the folder where you want to create the new lookup table.

8 Click **Save**. The application opens the new lookup table and displays the **Lookup Table** tab.

> **TIP** Objects that are saved in a folder for which the check-out and commit feature is enabled, such as the `Decision Repository` folder, must be checked out before they can be edited.

9 (Optional) If the lookup table is in a folder for which the check-out and commit feature is enabled, click the **Versions** tab and check out the latest version of the lookup table. For more information, see "Check Out and Commit a Lookup Table Version" on page 131.

10 Add entries to the new table either by importing a CSV file or by adding entries manually. See "Import Entries or Refresh Lookup Tables from CSV Files" on page 118 and "Add Lookup Table Entries" on page 119 for more information.

Importing and Exporting Lookup Tables

About Importing, Exporting, and Activating Lookup Tables

You can import lookup tables from and export lookup tables to comma-delimited (CSV) files. You can export an existing lookup table, edit the CSV file, and import the updated CSV file. You can export a lookup table from one environment and import it into another environment, or you can import and export lookup tables within the same environment.

> **IMPORTANT** The import and export features in SAS Intelligent Decisioning are for modifying an existing lookup table. If you do not need to edit the CSV file, use either the SAS Viya Command-Line (sas-viya)SAS administrative CLI (sas-admin) or SAS Environment Manager to transfer lookup tables between SAS Viya environments. The recommended approach is to use the transfer plug-in to the sas-viyasas-admin CLI. The CLI preserves the ID and version history of the objects being transferred. For lookup tables, the CLI also preserves information about which rule sets use the lookup table. For information about the CLI, see *SAS Viya Platform: Content Migration from*

SAS Viya 4. For information about using SAS Environment Manager, see "Content Page" in *SAS Environment Manager: User's Guide* and "Import Page" in *SAS Environment Manager: User's Guide*.

TIP Whether a lookup table that is active in the source environment is automatically activated in the target environment is determined by the setting of the activation.activateLookupOnImport configuration option. This option affects tables that are imported by using the SAS Viya Command-Line (sas-viya)SAS administrative CLI (sas-admin) and by using SAS Environment Manager. For more information, see "Reference Data Service Properties" in *SAS Intelligent Decisioning: Administrator's Guide*.

Format of the Lookup CSV Import File

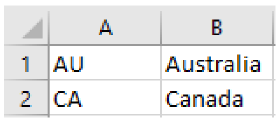
Note: The format of the CSV file depends on whether you are importing a CSV file by using the **Import** button in the user interface or by using the %DCM_IMPORT_LOOKUP macro. For information about the format that the macro uses, see "Format of the Lookup CSV Input File for the Macro" in *SAS Intelligent Decisioning: Macro Guide*.

Do not include a header row in the CSV file. Each row of the CSV input file identifies a key-value pair. To create a blank column in the CSV file, specify two comma separators without any content between them.

The following example specifies the keys **AU** and **CA** and associates them with the values **Australia** and **Canada**, respectively.

```
AU,Australia
CA,Canada
```

In a spreadsheet application, this lookup table appears as shown in the following figure.



	A	B
1	AU	Australia
2	CA	Canada

Import Entries or Refresh Lookup Tables from CSV Files

Note: It is recommended that a single lookup table contains no more than 10,000 entries. If you are importing a very large lookup table, you might need to increase the JVM heap size for the businessRulesReferenceData service. For more information, see "jvm.java_option_xmx" in *SAS Intelligent Decisioning: Administrator's Guide*.

Note: Lookup key names and values that contain two hash signs (##) are not supported.

You can import entries into an empty table, and you can refresh an existing lookup table by re-importing the same table.

1 Open the lookup table to which you want to import entries.

   TIP If you do not have permission to update an object, SAS Intelligent Decisioning displays `(Read-Only)` in the title bar next to the object name, and the buttons for modifying and saving the object are disabled.

2 Click **Import**. The Import Lookup Table window appears.

3 Click 🗀, and select the CSV file that contains the lookup table entries.

4 Select the encoding for the lookup table, and click **Import**.

Export a Lookup Table to a CSV File

IMPORTANT Do not modify the file structure or the header row in the exported CSV file. You can modify the data values.

To export a lookup table, open the lookup table and click **Export**. The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Add Lookup Table Entries

Note: It is recommended that a single lookup table contains no more than 10,000 entries.

TIP You cannot add new entries to a lookup table version that has been activated. In order to edit the table, you must create a new version.

1 Open the lookup table to which you want to add entries.

> **TIP** If you do not have permission to update an object, SAS Intelligent Decisioning displays `(Read-Only)` in the title bar next to the object name, and the buttons for modifying and saving the object are disabled.

2 Click **New Entries** if the lookup table is empty, or click + if the lookup table already contains entries. The Add Table Entries window appears.

3 Enter the lookup key name and value for the new entry. Key names and lookup values are each limited to 100 characters. Key names must be unique within the same lookup table.

Note: Lookup key names and values that contain two hash signs (##) are not supported.

To add additional entries, click **Add an entry**, and enter the new key name and value.

4 Click **Save** to save the new entries and close the Add Table Entries window.

Edit Lookup Table Entries

> **TIP** You cannot edit entries in a lookup table version that has been activated. In order to edit the table, you must create a new version.

1 Open the lookup table.

> **TIP** If you do not have permission to update an object, SAS Intelligent Decisioning displays `(Read-Only)` in the title bar next to the object name, and the buttons for modifying and saving the object are disabled.

2 Select the entries that you want to edit, and click ✎. The Edit Table Entries window appears.

3 Edit the exiting entries, and click **Save**.

Delete Lookup Table Entries

Open the lookup table, select the entries that you want to delete, and click 🗑.

Edit Lookup Key or Value Labels

1 In the Lookups category view, open the lookup table.

2 Click ✎ beside the **Key** or **Value** column heading in the lookup table. The Rename window appears.

3 Enter the label that you want to use for the key or value, and click **OK**.

Copy a Lookup Table URL

To create a link for external documentation that automatically opens a lookup table in SAS Intelligent Decisioning:

1 Open the lookup table.

2 Click ⋮, and select **Copy lookup table URL**. The Copy URL window appears, and the URL is automatically selected.

3 Click **Copy**, and then click **Close**.

Paste the link into your documentation.

Compare Lookup Table Content

You can compare the contents of two different lookup tables, or you can compare the contents of two different versions of the same lookup table.

1 Select the objects that you want to compare.

- To compare the contents of two different lookup tables, select the lookup tables in the category view, click ⋮, and select **Compare object contents**.

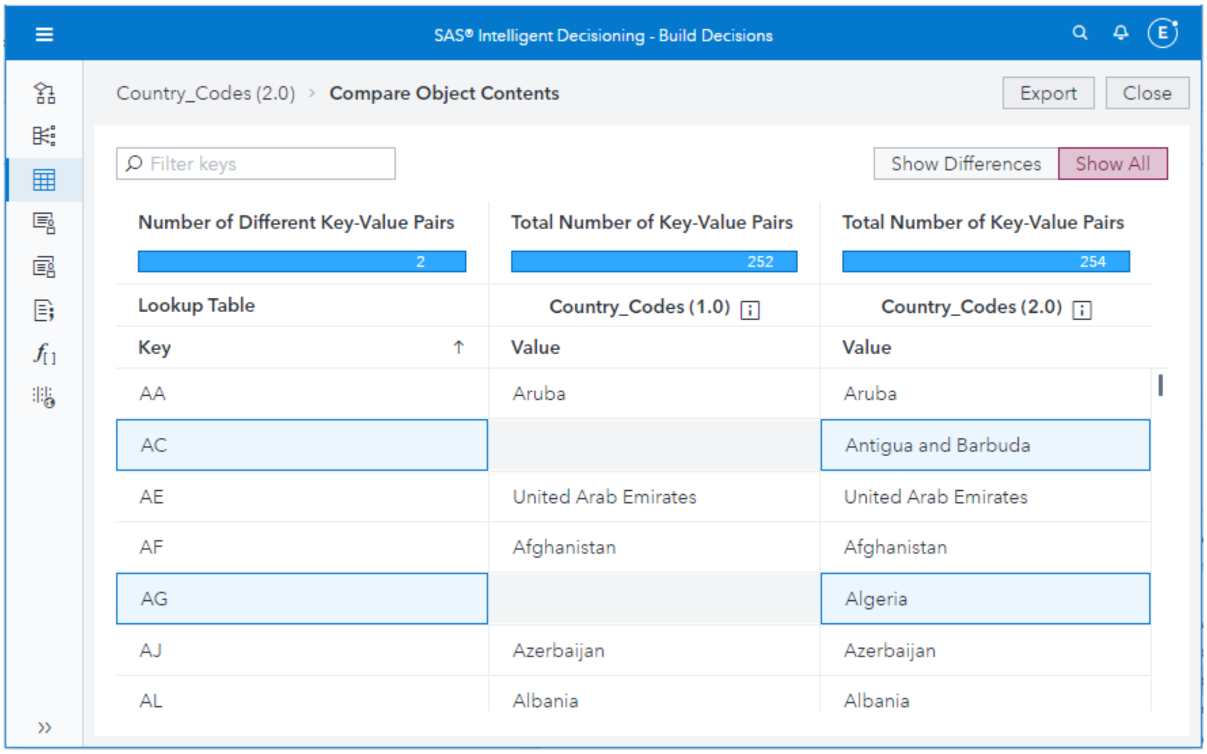
- To compare the contents of two versions of the same lookup table, open the lookup table, click ⋮ on the **Versions** tab, and select **Compare object contents**.

The Select Versions window appears.

2 Select the versions that you want to compare, and click **Compare**. SAS Intelligent Decisioning opens the two objects side-by-side in the Compare Object Contents window.

By default, the Compare Object Contents window displays the names and values in each entry in both objects and highlights the differences. To display all of the entries in each object, click **Show All**. For example, in the following figure, only version 2.0 of the Country_Codes lookup table contains entries for Algeria, and Antigua and Barbuda.

TIP Click ⓘ beside a table name to display its location.


SAS® Intelligent Decisioning - Build Decisions
Country_Codes (2.0) > Compare Object Contents
Export
Close
Filter keys
Show Differences
Show All

Number of Different Key-Value Pairs	Total Number of Key-Value Pairs	Total Number of Key-Value Pairs
2	252	254

Lookup Table	Country_Codes (1.0)	Country_Codes (2.0)
Key	**Value**	**Value**
AA	Aruba	Aruba
AC		Antigua and Barbuda
AE	United Arab Emirates	United Arab Emirates
AF	Afghanistan	Afghanistan
AG		Algeria
AJ	Azerbaijan	Azerbaijan
AL	Albania	Albania

3 (Optional) Click **Export** to export the results of the comparison to a PDF file. The Export Comparison Results window appears.

4 (Optional) Select the information that you want to export, and click **Export**. If you want the PDF file to display only the differences between the two objects, select **Show differences**. To display all of the objects' information, select **Show all**.

The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Generate PDF Documentation for a Lookup Table

You can generate detailed documentation for a lookup table as a PDF document. The PDF includes the lookup table properties and a table of the current key-value pairs that are contained in the lookup table.

1 Open the lookup table.

2 Click ⋮, and select **Create document**. The Create Document window appears.

> **TIP** If this option is disabled, there might be unsaved changes. Click 🖫.

3 (Optional) Enter a name for the document if you do not want to use the default name.

4 Click **Create**. SAS Intelligent Decisioning creates the PDF. The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Managing Lookup Tables

Duplicating Lookup Tables

About Duplicating Lookup Tables in Folders for Which the Check-Out and Commit Feature Is Enabled

When you duplicate a lookup table, SAS Intelligent Decisioning first creates an empty lookup table, and then updates the lookup table to add the individual lookup entries. The default permissions for the Decision Repository enable users to create new objects, but the permissions do not include the ability to update the objects. In order to duplicate an object that is in the Decision Repository, you must check out the object.

If your site has created additional folders for which the check-out and commit feature is enabled, and if those folders have been assigned the same permissions as those assigned to the Decision Repository, you must also check out objects in those folders before you can duplicate them.

Duplicate Lookup Tables

Note: You can duplicate the predefined Treatment Channels or Subject Level lookup tables, but the duplicates will not be used by SAS Intelligent Decisioning to determine channels or subject levels. You cannot delete the duplicates by using the SAS Intelligent Decisioning user interface because these tables are not stored in a folder. You must use the REST API to delete the duplicate tables.

To duplicate a single lookup table:

1 Select the table that you want to duplicate, click ⋮, and select **Duplicate**.

2 Enter a new name for the duplicate lookup table. Names are limited to 250 characters. Lookup table names are case insensitive and must be unique within the database.

3 (Optional) Enter a description for the duplicate table. Descriptions are limited to 1000 characters.

4 Select the version of the lookup table that you want to duplicate.

5 Click 🗀 and select the location where you want to save the duplicate lookup table.

6 Click **Duplicate**.

To duplicate multiple lookup tables, select the tables that you want to duplicate, click ⋮, and select **Duplicate**. SAS Intelligent Decisioning duplicates the latest version of the lookup tables and appends an `-Copy` to the names of the duplicate copies. If needed, a number is also appended to the duplicate table names.

Note: When you duplicate a lookup table, SAS Intelligent Decisioning creates a relationship between the original lookup table and the duplicate lookup table. If either object is changed, and you later copy the contents of one object into the other, SAS Intelligent Decisioning displays a warning message. Verify that you want to replace the current contents of the lookup table before you paste the new content.

Delete Lookup Tables

Select the tables that you want to delete, click ⋮, and select **Delete**. SAS Intelligent Decisioning moves the lookup table to the recycle bin. For more information, see "Manage Folders and Folder Content" on page 9.

Rename a Lookup Table

1 Select the table that you want to rename, click ⋮, and select **Rename**. The Rename window appears.

2 Enter a new name for the table, and click **Rename**.

Move Lookup Tables

Note: You cannot move the predefined Treatment Channels or Subject Level lookup tables.

1 Select the tables that you want to move, click ⋮, and select **Move**. The Choose a Location window appears.

2 Select the folder where you want to move the tables, and click **OK**.

Managing Versions of Lookup Tables

Set the Displayed Version

The displayed version is the version whose information is displayed on the **Properties** and **Lookup Table** tabs. On the **Versions** tab, a ✓ in the **Displayed Version** column indicates the displayed version. To change the displayed version, click the version number for the version that you want to view. The displayed version is shown in the title bar.

Create a New Version

Note: If you activate the only unlocked version of a lookup table, SAS Intelligent Decisioning automatically creates a new unlocked version. When you manually create a new version of a lookup table, previous versions are not locked. The only way to lock a version of a lookup table is to activate it. For more information, see "Activate a Lookup Table" on page 137.

Note: For objects that are stored in locations for which the check-out and commit feature is enabled, you cannot manually create a new version. The only way to create a new version is to check out an existing version and commit a new version. For information, see "Check Out and Commit a Lookup Table Version" on page 131.

IMPORTANT You cannot unlock a locked version. You cannot save changes to a version that is locked. If you modify a version that is locked and click 💾, SAS Intelligent Decisioning asks you if you want to replace the current unlocked version with your edited version.

To manually create a new version:

1 On the **Versions** tab, click the version number for the existing version that you want to use as the basis for the new version.

2 Click **New Version**. The New Version window appears.

3 Select the version type: **Minor** or **Major**. Version numbers follow the format *major.minor*. If you select **Major**, the number to the left of the period is incremented. If you select **Minor**, the number to the right of the period is incremented.

4 (Optional) For each version tag that you want to add, either enter a custom tag or begin typing and select a tag from the list of previously entered tags. Press Enter after each tag.

   Note: A tag is limited to 100 characters.

   TIP All tags that have been previously entered for any object are saved in the list that is displayed when you start typing. You cannot delete tags from this list.

5 (Optional) Enter information about the new version in the **Notes** field.

   TIP You can edit these notes at any time on the **Versions** tab.

6 Click **Save**.

Copy the Content of a Version

You can copy the content of an object's version in the category view or on the **Version** tab for the object.

1 In the category view, complete these steps:

   a Select the lookup table whose contents you want to copy.

   b Click ⋮ , and select **Copy version**. The Copy Version window appears.

   c Select the version whose contents you want to copy.

   Alternatively, on the **Versions** tab of the lookup table whose contents you want to copy:

   a Select the version whose contents you want to copy.

   b Click ⋮ , and select **Copy version**. The Copy Version window appears.

2 Click 🗀, and select the target lookup table into which you want to paste the contents of the version.

   When you paste the contents, SAS Intelligent Decisioning creates a new version of the target lookup table. The target object contains only the pasted content.

3 Select whether you want to create a new major or minor version.

4 (Optional) Modify the notes that will be associated with new version.

5 (Optional) Add tags that will be associated with the new version. Tags that are associated with a source object version are not automatically added to the new version. See "Add a Version Tag" on page 133

6 Click **Paste Version**, and then click **Yes**.

Note: When you copy the contents of a source object into a target object, SAS Intelligent Decisioning creates a relationship between the two objects. If the source object is modified after you copy its contents, and you later copy the contents of the target object back into the source object, SAS Intelligent Decisioning displays a warning message. Verify that you want to replace the current contents of the source object before you paste the new content.

Delete a Version

IMPORTANT When you delete a specific version, that version is deleted permanently. It is not moved into the recycle bin, and it cannot be restored.

Note: In order to be able to delete a specific version of an object, you must have permission to delete the object itself. Also, the configuration option sas.referencedata.deleteVersions must be turned on.

On the **Versions** tab, select the version that you want to delete, click ⋮, and select **Delete**.

You cannot delete the current version.

Determine Which Objects Use a Lookup Table

On the **Lookup tables** category page, select the check box for the lookup table, click ⋮, and select **View used by report**. The All Objects that Use the Selected Item window appears. This window lists all objects that use any version of the selected lookup table.

Note: The **View used by report** option is also available from within an open lookup table.

In the report, you can use the **Filter** field to filter the list of objects based on the object names. Click on an object name to open the object. Click ⓘ next to an object name to display the date on which the object was last modified and the ID of the user who modified the object. For decisions and code files that have been checked out, SAS Intelligent Decisioning also displays the IDs of the users that have checked out the objects. For decisions, SAS Intelligent Decisioning includes the decision’s workflow status.

Click **Export** to export the Used By report to a PDF file. In the PDF file, you can click on an object name to open the object in a new browser tab.

Checking Out and Committing Lookup Table Versions

About Checking Out and Committing Versions

Your administrator can enable the check-out and commit feature for lookup tables that are in any folder by specifying the folder in the sas.referencedata.checkout.checkoutEnabledFolderPaths configuration option. Enabling this feature for a folder does not automatically modify the permissions for the folder or for the objects in it. You can still modify a lookup table in the folder without checking it out, but you are expected to check out the latest version before you edit it. However, your administrator might also set permissions that require you to check out lookup tables in these folders before you can edit them. For more information, see "sas.referencedata.checkout" in *SAS Intelligent Decisioning: Administrator's Guide* and "Set Permissions for Check-Out Folders" in *SAS Intelligent Decisioning: Administrator's Guide*.

By default, SAS Intelligent Decisioning defines a folder where you can store lookup tables that must be checked out before they can be edited. This folder is the `Decision Repository` folder, and it is the default value for the sas.referencedata.checkout.checkoutEnabledFolderPaths configuration option. The default permissions for this folder require that non-administrative users check out a version and commit their changes to the checked-out version. Users who do not have administrative permissions cannot edit the lookup tables in `Decision Repository` without first checking them out.

If a version can be or must be checked out before it is modified, the **Check Out** button appears at the top of the **Versions** tab for that object. You can check out any version of an object. You can check out only one copy of a version at a time.

TIP If the sas.referencedata.checkout.allowConcurrentCheckout option is turned off, and a user has checked out a lookup table version, the **Check Out** button for that lookup table is disabled for all other users. For more information, see "Concurrently Checking Out and Committing Lookup Table Versions" on page 131.

When you check out a version, SAS Intelligent Decisioning writes a working copy of the version into your `My Folder` folder and opens the working copy. SAS Intelligent Decisioning adds "`(Checked Out)`" to the name that is displayed at the top of the window.

While you have a version checked out, the Lookup Tables category view shows two lookup tables with the same name, but the folders listed in the **Location** column differ for each lookup table. The original version is in the location specified by the sas.referencedata.checkout.checkoutEnabledFolderPaths configuration option, and the checked-out copy is in your `My Folder` folder.

TIP If an object that you have checked out does not appear in the category view, click ↺ to refresh the category view.

A **Commit** button appears at the top of the **Versions** tab for the checked-out version. When you are finished editing the checked-out version, you must commit your changes in order for other users to be able to see them. When you commit your changes, SAS Intelligent Decisioning creates a new version with your changes.

If the parent object is deleted before you commit your changes, you will not be able to commit your changes.

You cannot activate the checked-out version that is in `My Folder`. To activate a version with your changes, you must commit your changes, and activate the committed version.

Checking Out and Committing a Lookup Table from within a Rule Set

You can check out a lookup table from within a rule set if the following conditions are true:

- The rule set is checked out.
- The lookup table is stored in a folder for which the check-out feature is enabled.
- The rule set does not have any unsaved changes. If the rule set contains any unsaved changes, the **Check out lookup** option is disabled.

To check out or commit a lookup table from within a checked-out rule set, right-click on the LOOKUP or LOOKUPVALUE expression, and select **Check out lookup** or **Commit lookup**. SAS Intelligent Decisioning displays as asterisk in front of the name of any lookup table that is checked out.

When you check out a lookup table from within a rule set, the checked-out rule set is modified to use the checked-out copy of the lookup table, and SAS Intelligent Decisioning enables the **Commit lookup** option in the menu. When you commit the rule set, all of the checked-out lookup tables that are used in the rule set are also committed.

The newly committed versions of the lookup tables are not automatically activated when they are committed. You must activate the newly committed versions of the lookup tables. For more instructions, see “Activate a Lookup Table” on page 137.

Rule sets that are not checked out always use the activated versions of lookup tables. If you do not activate the newly committed versions of the lookup tables, all rule sets that use the lookup tables will continue to use the previously activated versions of each table.

To cancel the check out of a lookup table, right-click on the LOOKUP or LOOKUPVALUE expression, and select **Cancel check out**.

Committing a Lookup Table Together with a Rule Set

If you check out a lookup table from within a rule set and then commit the rule set, the lookup table is also committed. For more information, see “Checking Out and Committing a Lookup Table from within a Rule Set” on page 130.

Alternatively, you can check out a rule set that uses a lookup table, open the lookup table in the **Lookup tables** category view and check it out, then change the reference in the rule set to use the checked-out version of the lookup table. The lookup table is committed when you commit the rule set.

The newly committed version of the lookup table is not automatically activated when they are committed. You must activate the newly committed version of the lookup table. For more instructions, see “Activate a Lookup Table” on page 137.

Concurrently Checking Out and Committing Lookup Table Versions

The ability for multiple users to check out the same lookup table at the same time is controlled by the sas.referencedata.checkout.allowConcurrentCheckout configuration option. This option is turned on by default.

When this option is turned on, different users can check out the same version of the same object at the same time. Because the objects that are checked out are saved in each user's `My Folder` location, the default permissions allow individual users to see only the copies that they have checked out.

When this option is turned off and a user has checked out an object, the **Check Out** button for that object is disabled for all other users.

If multiple users check out the same version of the same object at the same time, each user's changes are preserved in a new version when they commit their changes. One user's changes do not overwrite another user's changes.

> **IMPORTANT** If two users attempt to commit changes to the same object simultaneously, the first user's attempt will succeed but the second user might see an error message that the commit has failed. If the second user subsequently commits their changes, the **Modified By** column on the **Versions** tab for both the version committed by the first user and the version committed by the second user displays the user ID of the second user.

Check Out and Commit a Lookup Table Version

1 On the **Versions** tab, click **Check Out**.

   SAS Intelligent Decisioning updates the **Properties** tab to indicate that the version is checked out.

2 Modify the checked-out version as needed, and save it.

   > **TIP** To discard the changes and delete the checked-out version from `My Folder`, you can commit the object without saving it first. However, committing the object without saving creates a new version of the object whose contents match the contents of the previous version. For information on undoing a check out, see "Undoing a Check Out" on page 132.

3 On the **Versions** tab, click **Commit**. The Commit Lookup Table Version window appears.

4 Select the version type: **Minor** or **Major**. Version numbers follow the format *major.minor*. If you select **Major**, the number to the left of the period is incremented, and the minor number is reset to zero. If you select **Minor**, the number to the right of the period is incremented.

5 (Optional) In the **Version tags** field, enter any version tags that you want to associate with the new version. Press Enter after each tag. Tags are limited to 100 characters each and cannot contain the following characters: $ ' " # & ? ( ) \ /

6 (Optional) Enter information about the new version in the **Notes** field.

7 Click **Commit**. SAS Intelligent Decisioning creates a new version with your changes, and deletes the working copy from your `My Folder` folder.

Determine Who Has a Version Checked Out

If the current version of an object is checked out, the IDs of the users that checked it out and the timestamps when each user checked it out appear in the **Checked out by** field on the **Properties** tab for the original object. You can also display this information by clicking ↗ beside the version number on the **Versions** tab.

Opening the Original Object

When you check out an object, SAS Intelligent Decisioning adds the field **Original object link** to the **Properties** tab for the checked-out object. This field contains a link to the original object that was checked out. You can use this link to verify that you have checked out the correct version and to compare the original content with the modified content in the checked-out version.

Undoing a Check Out

If both an object and a decision that uses the object are checked out at the same time, or if you checked out the object from within the decision, click ⋮ on the object's node in the decision diagram, and select **Cancel checkout**.

You can discard a checked-out version and any changes that you made by deleting the working copy of the version from your `My Folder` folder if the following conditions are true:

- You have not checked out a decision that uses the object.
- The object was not checked out at the same time as a decision that uses the object, or the object was not checked out from within the decision after the decision was checked out.

The deleted version is moved to the recycle bin. See "Delete Lookup Tables" on page 125.

Managing Version Tags for Lookup Tables

Add a Version Tag

Version tags enable you to better organize and group your content. Version tags are associated with specific versions of an object and not with the entire object. You can add the same tag to any version of any object. To add a tag to a lookup version:

1. On the **Versions** tab, position your cursor in the **Version Tags** column for the version that you want to tag. If the version is not tagged, + appears. If the version has at least one tag, ✎ appears.
2. Click + to open the New Version Tags window, or click ✎ to open the Edit Version Tags window.
3. For each tag that you want to add, either enter a custom tag or begin typing and select a tag from the list of previously entered tags. Press Enter after each tag. Tags are limited to 100 characters each and cannot contain the following characters: $ ' " # & ? ( ) \ /

   TIP All tags that have been previously entered for any object are saved in the list that is displayed when you start typing. You cannot delete tags from this list.

   TIP To filter the version list based on a tag, right-click on the **Version Tags** column heading, enter a filter string in the **Filter** box, and press Enter. To clear the filter, delete the string from the **Filter** box, and press Enter.

4. Click **Close** to close the window.

Remove a Version Tag

1. On the **Versions** tab, position your cursor in the **Version Tags** column for the version whose tag you want to remove, and click ✎.
2. Click × beside the tag that you want to remove.

3 Click **OK** to close the window. The tag remains in the list of previously entered tags that is displayed when you add a tag, but the tag is no longer associated with version.

Modify a Version Tag

You cannot modify a version tag that already exists. To change the content of an existing tag, delete the tag as described in "Remove a Version Tag" on page 133, and then add the tag again as described in "Add a Version Tag" on page 133.

Activating a Lookup Table

What Does Activating a Lookup Table Do?

Activating a lookup table makes the table available for use by published objects. By default, when you activate a lookup table, the lookup table is converted into executable code in the form of a SAS Cloud Analytic Services (CAS) format. If your site has a SAS Micro Analytic Service publishing destination, then a SAS Micro Analytic Service module is also created for the table. The CAS format is used when you create and run tests and scenario tests (pre-publishing tests) for the rule sets and decisions that use the lookup tables. The CAS format and SAS Micro Analytic Service module also enable objects that are published to those destination types to use the most recently activated version of a lookup table instead of a static copy.

However, your administrator can set configuration options to specify that SAS Intelligent Decisioning includes lookup tables inline in generated code instead of using formats and modules for CAS and SAS Micro Analytic Service destinations. For more information, see "Controlling Where Lookup Tables Are Activated And How They Are Used" on page 135.

Teradata, Hadoop, and container destinations do not support the format or module that is created when you activate a lookup table. Objects that are published to these destinations must include a static version of each lookup table.

You must activate a version of any lookup table that is used in a rule set or decision. When you activate a version of a lookup table, that version is locked and cannot be edited. Each lookup table can have only one active version.

The active version of a lookup table is used in the following ways:

- when you run a test or scenario test prior to publishing an object.
- to generate a static copy of the table when you publish an object if a static copy is needed, such as when you publish the object to Teradata, Hadoop, or container destinations. In order to generate the static copy for Teradata or Hadoop destinations, the lookupStaticBinding configuration property must be set to On. Static copies are automatically generated for container destinations.

- by published objects in CAS and SAS Micro Analytic Service destinations if the lookupStaticBinding configuration property is set to Off.

Controlling Where Lookup Tables Are Activated And How They Are Used

Your administrator can use the following configuration options to control where lookup tables are activated and how they are used:

sas.referencedata.activation.destinations = *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. For more information, see "sas.referencedata.activation.destinations" in *SAS Intelligent Decisioning: Administrator's Guide*.

sas.referencedata.publish.lookupDisableMasPublish = On | Off
: controls whether lookup tables are activated in your SAS Micro Analytic Service publishing destination. When this option is set to Off and you activate a lookup table, it is activated in this destination. If this option is set to On, then lookup tables are not activated in this destination, and your administrator must set the sas.businessrules.lookupStaticBinding option to On in order to embed static copies of lookup tables in generated code and ensure that published content executes correctly. For more information, see "sas.referencedata.publish.lookupDisableMasPublish" in *SAS Intelligent Decisioning: Administrator's Guide*.

sas.businessrules.lookupStaticBinding = On | Off
: controls whether a static copy of the active version of a lookup table is embedded in the generated code for an object when that object is published. When this option is set to On, a static copy of the active version of any lookup table that is used in an object is embedded in the generated code when those objects are published. When this option is set to Off, a published object's use of lookup tables depends on the destination to which they were published.

  - Objects that are published to SAS Cloud Analytic Services (CAS) and SAS Micro Analytic Service destinations use the most recently activated version of lookup tables.
  - Objects that are published to Teradata and Hadoop destinations cannot use lookup tables.

  Note: Static versions of lookup tables are automatically embedded in content published to container destinations. This option is ignored for container destinations.

  For more information, see "sas.businessrules.lookupStaticBinding" in *SAS Intelligent Decisioning: Administrator's Guide*.

TIP To publish very large lookup tables to CAS destinations, set both of the following options to On:

- sas.referencedata.publish.lookupDisableMasPublish
- sas.businessrules.lookupStaticBinding

Table 4.1 Interaction of Lookup Table Configuration Settings

sas.referencedata.publish.lookupDisableMasPublish	sas.businessrules.lookupStaticBinding	SAS Micro Analytic Service	CAS	Teradata and Hadoop	Container Destinations
Off	Off	Published objects use the most recently activated versions of lookup tables.	Published objects use the most recently activated versions of lookup tables.	Published objects cannot use lookup tables.	Published objects use the embedded versions of lookup tables.
Off	On	Published objects use the embedded versions of lookup tables.	Published objects use the embedded versions of lookup tables.	Published objects use the embedded versions of lookup tables.	Published objects use the embedded versions of lookup tables.
On	Off	Do not use lookup tables. These option settings might produce incorrect results if a lookup table was previously activated in a SAS Micro Analytic Service destination.	Published objects use the most recently activated versions of lookup tables.	Published objects cannot use lookup tables.	Published objects use the embedded versions of lookup tables.
On	On	Published objects use the embedded versions of lookup tables.	Published objects use the embedded versions of lookup tables.	Published objects use the embedded versions of lookup tables.	Published objects use the embedded versions of lookup tables.

Activate a Lookup Table

Activating a lookup table locks it. If you activate the only unlocked version of a lookup table, SAS Intelligent Decisioning automatically creates a new unlocked version.

You cannot activate a lookup table that has been checked out. You must commit the changes to the lookup table before you can activate it.

To activate the latest version of a lookup table that is not in a folder for which the check-out feature has been enabled, open the lookup table, and click **Activate** at the top of the window.

To activate a specific version of the lookup table, or to activate a lookup table that is in a folder for which the check-out feature is enabled, you must activate the lookup table on the **Versions** tab:

1 Open the lookup table that you want to activate.

> TIP If you do not have permission to update an object, SAS Intelligent Decisioning displays `(Read-Only)` in the title bar next to the object name, and the buttons for modifying and saving the object are disabled.

2 On the **Versions** tab, select the version that you want to activate, and click **Set Version** to set it as the displayed version. See "Set the Displayed Version" on page 125 for more information.

3 Click **Activate**.

Dictionary

LOOKUP Function

Determines whether a lookup key exists in a lookup table.

Restrictions:	You can specify the LOOKUP function only in condition expressions. If an expression contains the LOOKUP function, then the expression cannot contain anything else.
Returned data type:	Boolean

Syntax

LOOKUP ('*lookup_table_name*', *variable_or_value*)

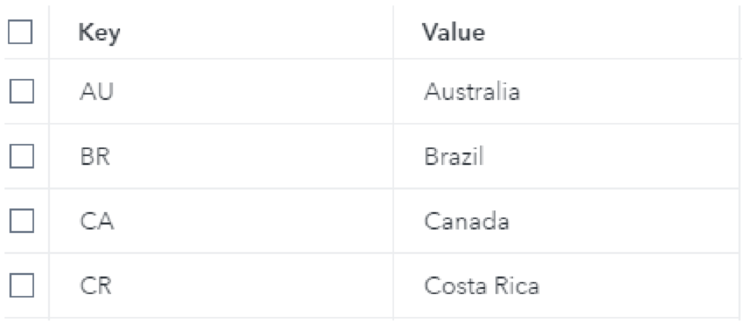
Required Arguments

lookup_table_name
: specifies the name of the lookup table that you want to search.

variable_or_value
: specifies either the literal key value or a variable that contains a lookup key value.

Example

Suppose you have a Country_Codes lookup table that uses two-letter abbreviations for countries as the lookup key and country names as the lookup values.



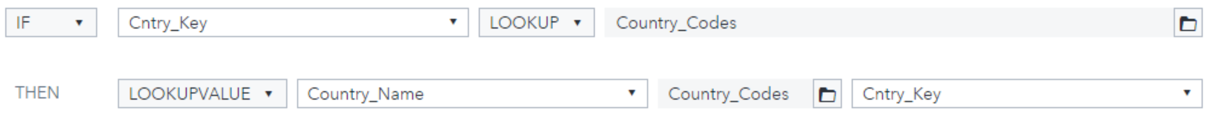
☐	Key	Value
☐	AU	Australia
☐	BR	Brazil
☐	CA	Canada
☐	CR	Costa Rica

To verify that the value of the variable Cntry_Key exists as a lookup key in the table Country_Codes, you can use the following expression:

```
LOOKUP('Country_Codes',Cntry_Key)
```

If the value of Cntry_Key exists as a lookup key, the LOOKUP function returns the value `True`.

In the following rule, if the key specified by the variable Cntry_Key exists in the lookup table Country_Codes, then the value that is associated with that key is assigned to the variable Country_Name.



IF Cntry_Key LOOKUP Country_Codes
THEN LOOKUPVALUE Country_Name Country_Codes Cntry_Key

LOOKUPVALUE Function

Retrieves a lookup value from a lookup table.

Restrictions: You can specify the LOOKUPVALUE function only in action expressions.

If an expression contains the LOOKUPVALUE function, then the expression cannot contain anything else.

Returned data type: CHARACTER, INTEGER, DECIMAL, DATE, DATETIME, BOOLEAN

Tip: Lookup tables are stored as character data. However, you can assign the results of the LOOKUPVALUE function to any of the returned data types. The LOOKUPVALUE function converts the results to match the type of the variable.

Syntax

LOOKUPVALUE ('*lookup_table_name*', *variable_or_value*)

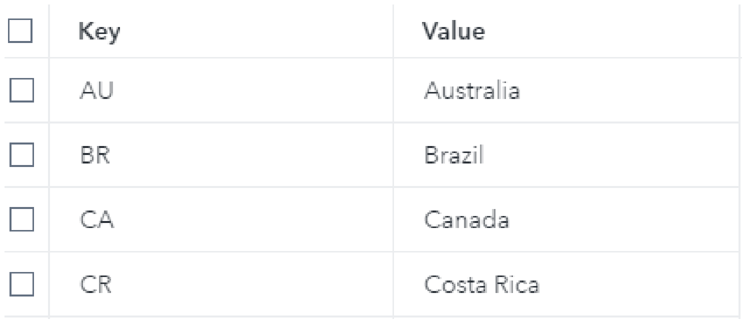
Required Arguments

lookup_table_name
specifies the name of the lookup table that you want to search.

variable_or_value
specifies either the literal key value or a variable that contains the lookup key value.

Example

Suppose you have a Country_Codes lookup table that uses two-letter abbreviations for countries as the lookup key and country names as the lookup values. The Country_Codes lookup table contains the lookup key `CA`, and the lookup value that corresponds to that key is `Canada`.

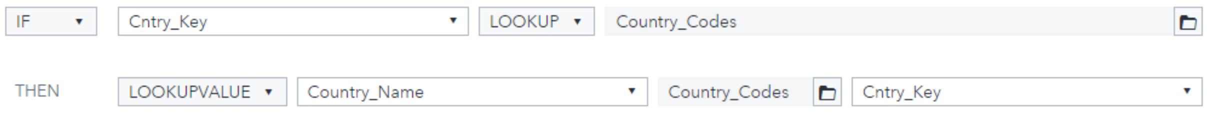


☐	Key	Value
☐	AU	Australia
☐	BR	Brazil
☐	CA	Canada
☐	CR	Costa Rica

If the Cntry_Key variable in the current input record contains the value `CA`, you can use the following expression to retrieve the lookup value that is associated with that key from the table Country_Codes:

```
LOOKUPVALUE('Country_Codes',Cntry_Key)
```

In the following rule, if the key specified by the variable Cntry_Key exists in the lookup table Country_Codes, then the value that is associated with that key is assigned to the variable Country_Name.



IF Cntry_Key LOOKUP Country_Codes

THEN LOOKUPVALUE Country_Name Country_Codes Cntry_Key

5

Managing Global Variables

Using Global Variables **141**

Create a New Global Variable **142**

Managing Global Variables **143**
Duplicate a Global Variable 143
Remove a Global Variable from the List 143
Edit a Global Variable 144

Managing Versions of Global Variables **144**
Set the Displayed Version 144
Create a New Version 144
Delete a Version 145

Determine Which Objects Use a Global Variable **146**

Managing Version Tags for Global Variables **146**
Add a Version Tag 146
Remove a Version Tag 147
Modify a Version Tag 147

Manage the Global Variable Recycle Bin **147**

Activating a Global Variable **148**
What Does Activating a Global Variable Do? 148
Controlling Where Global Variables Are Activated and How They Are Used 149
Activate a Global Variable 149

Using Global Variables

You can use global variables to define common variables for use in multiple rule sets and decisions. For example, you might want to define global variables for benchmark interest rates, current exchange rates, or inventory levels for specific products. By using global variables, you can set these values in one place and refer to the variables in multiple objects.

TIP You cannot use global variables in models in SAS Model Manager. You can pass the value of a global variable into a model by mapping a model input parameter to a global variable. For more information, see "Mapping Variables within a Decision" on page 247.

After you create and activate a global variable, you can add it to the **Global Variables** tab of a rule set or a decision. You can use global variables in conditional expressions, but you cannot modify the value of a global variable in a rule set or in a decision.

When you add a rule set that uses a global variable to a decision, SAS Intelligent Decisioning creates a decision variable with the same name as the global variable. When you test the decision, SAS Intelligent Decisioning assigns the current value of the global variable to the decision variable. When you run a published decision, the value that is assigned to the decision variable depends on the destination type. For information, see "Content Executed by Published Decisions" on page 309.

When you add a common rule set to an assignment rule set and the common rule set uses global variables, you must manually add the global variables to the assignment rule set.

Note: The decision variable is not a global variable, but it has the same name and value as the global variable. For more information about decision variables, see "About Decision Variables and Mapping" on page 247.

For more information, see "Add Global Variables to a Rule Set" on page 19 and "Add Global Variables to a Decision" on page 225.

Create a New Global Variable

1 Click to navigate to the Global Variables category view.

2 Click **New Global Variable**. SAS Intelligent Decisioning displays the New Global Variable window.

3 Enter a name for the variable.

4 (Optional) Enter a description for the new global variable. Descriptions are limited to 1000 characters.

TIP You can edit the description later on the **Properties** tab.

5 Select the variable's data type.

6 Enter a value for the variable. To enter a value for a date or datetime variable, click or , and select the value in the date or datetime picker.

7 Click **Save** to create the variable.

Managing Global Variables

Duplicate a Global Variable

To duplicate a single global variable:

1. Select the variable that you want to duplicate, click ⋮, and select **Duplicate**.
2. Enter a new name for the duplicate global variable. Names are limited to 250 characters. Global variable names are case insensitive and must be unique within the database.
3. (Optional) Enter a description for the duplicate variable. Descriptions are limited to 1000 characters.
4. Select the version of the global variable that you want to duplicate.
5. Click **Duplicate**.

To duplicate multiple global variables, select the variables that you want to duplicate, click ⋮, and select **Duplicate**. SAS Intelligent Decisioning duplicates the latest version of the global variables and appends `-Copy` to the names of the duplicate copies. If needed, a number is also appended to the duplicate variable names.

Remove a Global Variable from the List

You cannot publish an object that uses a global variable that has been removed from the list.

In the Global Variables category view, select the variables that you want to remove, click ⋮, and select **Delete**. SAS Intelligent Decisioning moves the deleted global variables to the global variables recycle bin. See “Manage the Global Variable Recycle Bin” on page 147 for more information.

TIP A global variable that has been removed from the list might be referenced in objects that have already been published. To remove references to global variables from a published object, you must republish the object.

Edit a Global Variable

You cannot edit a global variable version that has been activated, and you cannot rename a global variable.

To edit the description or value of the most current version of a global variable, click on the variable in the Global Variables category view.

Managing Versions of Global Variables

Set the Displayed Version

The displayed version is the version whose information is displayed on the **Properties** tab. On the **Versions** tab, a ✓ in the **Displayed Version** column indicates the displayed version. To change the displayed version, click the version number for the version that you want to view. The displayed version is shown in the title bar.

Create a New Version

Note: For objects that are stored in locations for which the check-out and commit feature is enabled, you cannot manually create a new version. The only way to create a new version is to check out an existing version and commit a new version. For information, see "Check Out and Commit a Lookup Table Version" on page 131.

Note: The current version of an object is the version with the highest version number. When you create a new version, SAS Intelligent Decisioning locks the current version before it creates the new version.

IMPORTANT You cannot unlock a locked version. You cannot save changes to a version that is locked. If you modify a version that is locked and click 🖫, SAS Intelligent Decisioning asks you if you want to replace the current unlocked version with your edited version.

To manually create a new version:

1 On the **Versions** tab, click the version number for the existing version that you want to use as the basis for the new version.

2 Click **New Version**. The New Version window appears.

3 Select the version type: **Minor** or **Major**. Version numbers follow the format *major.minor*. If you select **Major**, the number to the left of the period is incremented. If you select **Minor**, the number to the right of the period is incremented.

4 (Optional) For each version tag that you want to add, either enter a custom tag or begin typing and select a tag from the list of previously entered tags. Press Enter after each tag.

Note: A tag is limited to 100 characters.

TIP All tags that have been previously entered for any object are saved in the list that is displayed when you start typing. You cannot delete tags from this list.

5 (Optional) Enter information about the new version in the **Notes** field.

TIP You can edit these notes at any time on the **Versions** tab.

6 Click **Save**.

Delete a Version

IMPORTANT When you delete a specific version, that version is deleted permanently. It is not moved into the recycle bin, and it cannot be restored.

Note: In order to be able to delete a specific version of an object, you must have permission to delete the object itself.

On the **Versions** tab, select the version that you want to delete, click ⋮, and select **Delete**.

You cannot delete the current version.

Determine Which Objects Use a Global Variable

On the **Global variables** category page, select the check box for the global variable, click ⋮, and select **View used by report**. The All Objects that Use the Selected Item window appears. This window lists all objects that use any version of the selected global variable.

In the report, you can use the **Filter** field to filter the list of objects based on the object names. Click on an object name to open the object. Click ⓘ next to an object name to display the date on which the object was last modified and the ID of the user who modified the object. For decisions and code files that have been checked out, SAS Intelligent Decisioning also displays the IDs of the users that have checked out the objects. For decisions, SAS Intelligent Decisioning includes the decision’s workflow status.

Click **Export** to export the Used By report to a PDF file. In the PDF file, you can click on an object name to open the object in a new browser tab.

Managing Version Tags for Global Variables

Add a Version Tag

Version tags enable you to better organize and group your content. Version tags are associated with specific versions of an object and not with the entire object. You can add the same tag to any version of any object. To add a tag to a global variable version:

1 On the **Versions** tab, position your cursor in the **Version Tags** column for the version that you want to tag. If the version is not tagged, + appears. If the version has at least one tag, ✎ appears.

2 Click + to open the New Version Tags window, or click ✎ to open the Edit Version Tags window.

3 For each tag that you want to add, either enter a custom tag or begin typing and select a tag from the list of previously entered tags. Press Enter after each tag. Tags are limited to 100 characters each and cannot contain the following characters: $ ' " # & ? ( ) \ /

TIP All tags that have been previously entered for any object are saved in the list that is displayed when you start typing. You cannot delete tags from this list.

TIP To filter the version list based on a tag, right-click on the **Version Tags** column heading, enter a filter string in the **Filter** box, and press Enter. To clear the filter, delete the string from the **Filter** box, and press Enter.

4 Click **Close** to close the window.

Remove a Version Tag

1 On the **Versions** tab, position your cursor in the **Version Tags** column for the version whose tag you want to remove, and click ✎.

2 Click ✕ beside the tag that you want to remove.

3 Click **OK** to close the window. The tag remains in the list of previously entered tags that is displayed when you add a tag, but the tag is no longer associated with version.

Modify a Version Tag

You cannot modify a version tag that already exists. To change the content of an existing tag, delete the tag as described in "Remove a Version Tag" on page 147, and then add the tag again as described in "Add a Version Tag" on page 146.

Manage the Global Variable Recycle Bin

To open the recycle bin in the Global Variables category view, click **Open** and select **Recycle bin**.

To restore an item from the recycle bin, right-click on the item in the recycle bin, and select **Restore**.

To permanently delete an item, right-click on the item in the recycle bin, and select **Delete**. The item is deleted from the recycle bin and cannot be restored.

Activating a Global Variable

What Does Activating a Global Variable Do?

Activating a global variable makes that variable available for use by published objects. Published objects can use only activated versions. By default, when you activate a global variable, the value of the global variable is included inline in code that is generated when you publish rule sets and decisions. However, this behavior is controlled by the inlineGlobalVariables configuration options that your administrator can set.

If these configuration options are set to Off, then when you activate a global variable, the global variable is converted into executable code in the form of a SAS Cloud Analytic Services (CAS) format. If your site has a SAS Micro Analytic Service publishing destination, then a SAS Micro Analytic Service module is also created for the variable. The CAS format and SAS Micro Analytic Service module enable objects that are published to those destination types to use the most recently activated version of a global variable instead of using a static copy.

The active version of a global variable is the version that is used when you run basic tests, scenario tests, and publishing validation tests for a rule set or decision that uses the global variable.

Teradata, Hadoop, and container destinations do not support the format or module that is created when you activate a global variable. Objects that are published to these destinations always include a static version of each global variable.

Note: In multi-tenant environments, each tenant has its own global variables that are unique to that tenant.

You must activate a version of any global variable that is used in a rule set or in a decision. When you activate a version of a global variable, that version is locked and cannot be edited. Each global variable can have only one active version.

The active version of a global variable is used in the following ways:

- to generate a static copy of the variable when you publish an object if a static copy is needed, such as when you publish the object to Teradata, Hadoop, or container destinations. Static copies are always generated for these destinations.
- by published objects in CAS and SAS Micro Analytic Service destinations if the inlineGlobalVariableValues configuration properties are set to Off. See "Controlling Where Global Variables Are Activated and How They Are Used" on page 149 for more information.

Controlling Where Global Variables Are Activated and How They Are Used

Your administrator can use the following configuration options to control how global variables are used:

sas.businessrules.inlineGlobalVariableValues = On | Off
: specifies whether the current values of global variables are included inline in generated code when you publish a rule set to CAS or SAS Micro Analytic Service destinations.

sas.decisions.inlineGlobalVariableValues = On | Off
: specifies whether the current values of global variables are included inline in generated code when you publish a decision to CAS or SAS Micro Analytic Service destinations.

When these properties are set to On, the current value of each global variable is included in the code that is generated for CAS and SAS Micro Analytic Service destinations. Changes to the values of the variables do not affect published objects. When these properties are 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 or decision is run.

Note: Static versions of global variables are automatically embedded in content that is published to Teradata, Hadoop, and container destinations. These options are ignored for those destinations.

Activate a Global Variable

Activating a version of a global variable locks that version. Locked variables cannot be unlocked.

You must activate a global variable before you can publish an object that uses the variable. When you activate a global variable, it is activated in all of the publishing destinations that are configured at your site.

Published rule sets and decisions use activated versions of global variables. The version varies according to the destination type. For more information, see “Content Executed by Published Decisions” on page 309 and “Content Executed by Published Rule Sets” on page 79.

1 Open the global variable that you want to activate.

2 On the **Versions** tab, select the version that you want to activate, and click **Activate**.

3 Click **Save** to close the Edit Global Variable window.

6 Using Custom Code Files

Introduction to Custom Code Files 152

DS2 Code Files 153
Rules for Creating DS2 Code Files 153
Example: Custom DS2 Code File 154
Testing DS2 Code Files in SAS Studio 155

Python Code Files 157
About Publishing Destinations and Python Distributions 157
Rules For Developing Python Code Files 157
Required Dependencies For Python Files in SAS Container Runtime Destinations 158

Data Query Files 159
Using Data Query Files 159
Query Output Types and Editors 160
Developing SQL Code 160
Decision Variables for Data Query Files 161
Using Macro Variables in SAS Studio 162
Handling Table and Column Names in Data Query Files 163

Custom Context Files 163
Using Custom Context Files 163
Adding an INIT Method to a Custom Context File 164
Associating a Custom Context File with a Decision 164
Testing or Validating a Rule Set, DS2 File, or Custom Function 165
Referencing a Function That Is Defined in a Context File 165
Referencing a Variable That Is Defined in a Context File 166

Create a New Code File 167

Managing the Variables in a DS2 or Python Code File 168
About Variables 168
Syncing Variables and Code 170
Add Variables from a Data Table 171
Add Variables from a Rule Set, Decision, or Code File 172
Create Custom VariablesCreate Custom Variables 172
Duplicate a Variable 173
Importing and Exporting Variables 174
Delete Variables 175

Edit Variable Properties 175
Edit Metadata for Data Grid Variables 175

Validate SQL, DS2, or Context Files **175**

Copy a Code File URL **176**

Compare Generated Code for Code Files **176**

Managing Code Files **177**
Duplicate Code Files 177
Delete Code Files 178
Rename a Code File 178
Move Code Files to a Different Folder 178

Managing Versions of Code Files **178**
Set the Displayed Version 178
Create a New Version 179
Copy the Content of a Version 180
Delete a Version 180
Upgrade Decisions to Use a New Version of a Code File 181

Determine Which Objects Use a Code File **181**

Checking Out and Committing Code File Versions **182**
About Checking Out and Committing Versions 182
Checking Out a Code File from within a Decision 183
Concurrently Checking Out and Committing Code File Versions 184
Check Out and Commit a Custom Code File Version 184
Determine Who Has a Version Checked Out 185
Opening the Original Object 185
Undoing a Check Out 185

Managing Version Tags for Code Files **186**
Add a Version Tag 186
Remove a Version Tag 186
Modify a Version Tag 187

Testing Code Files **187**
Ways to Test a Code File 187
Test a Code File 188
Test a Scenario 190
Working with Test Output Data 194
Status Icons for Tests 194

Manage Comments for a Code File Test **195**

Introduction to Custom Code Files

You can define custom code files to do things that are not possible in rules, models, or treatments. For example, you can define a code file that makes HTTP calls to REST APIs, interacts with a database, manipulates files in the file system, or performs custom data transformations. You can use custom context files to define functions that you can use in rule sets, in custom DS2 code files, and in custom functions that are defined in the Custom Functions category.

IMPORTANT You are responsible for ensuring that only users who are authorized to do so can install Python libraries, develop, and test code nodes that use SQL or Python, and execute decisions that use custom code nodes.

IMPORTANT All changes that you make to a code file affect all unpublished decisions that use that file. For more information, see "Content That Is Used by Tests and Scenarios for Decisions" on page 303 and "Content Executed by Published Decisions" on page 309.

Note: Not all of the code that you can write in a custom code file will work in all publishing destinations or in decision tests.

For information about adding a custom code file to a decision, see "Adding Objects to a Decision" on page 230.

DS2 Code Files

Rules for Creating DS2 Code Files

When you are developing your DS2 package, follow these rules:

- Do not change the package name in the PACKAGE statement:

  ```
  package "${PACKAGE_NAME}" /inline;
  ```

  This line must appear exactly as shown and must begin in column one of the code file. The token is replaced with a package name that SAS Intelligent Decisioning uses to maintain the relationship between the code file and the decisions that use it.

- If your custom DS2 code file defines multiple packages, the packages that are used by the `${PACKAGE_NAME}` package must be defined before the `${PACKAGE_NAME}` package.

- Do not specify any DS2 options (DS2_OPTIONS statement) in your package code.

- Custom DS2 code files in SAS Intelligent Decisioning support only three data types: double, varchar, and package datagrid. These data type names are case sensitive. List input-only parameters first. List input-output and output-only parameters next, using the in_out modifier. Do not specify a length for input-output and output-only parameters. The length for these parameters is derived from the variables that are passed into the method.

  For example, the following lines define the variables LOAN, REASON, and ASSETS.

  ```
  package datagrid "assets"
  ```

```
double "loan"
IN_OUT varchar "reason"
```

IMPORTANT Before you validate a DS2 code file or custom context file that uses a data grid package, verify that the declaration statements for the package specify `datagrid` instead of `dcm_datagrid`. For example:

```
in_out package datagrid variable;

dcl package datagrid variable;
```

Note: Packages are input-output data types.

- Your package must define an EXECUTE method.
- The variables that you use as parameters for the EXECUTE method in your code must be mapped to input variables in your decision.
- Do not enter comments within the signature of the method parameters in your code file. The comment characters cause the signature to be parsed incorrectly.
- To query a database, use a custom data query file instead of using a DS2 SQLSTMT package in a DS2 code file. For information, see "Data Query Files" on page 159.
- Custom DS2 code files are executed by using the runDS2 CAS action. This action supports varying-length binary variables, but it does not support binary variables.

For information about developing DS2 packages, processing data grids, and the SAS APIs, see the following:

- *SAS DS2 Programmer's Guide*
- *SAS DS2 Language Reference*
- *SAS Intelligent Decisioning: Using Data Grids*
- "Reserved Words in the DS2 Language" in *SAS DS2 Programmer's Guide*
- http://developer.sas.com
- "DS2 Programming for SAS Micro Analytic Service" in *SAS Micro Analytic Service: Programming and Administration Guide*
- "Best Practices for DS2 Programming in SAS Intelligent Decisioning" in *SAS Micro Analytic Service: Programming and Administration Guide*

Example: Custom DS2 Code File

The following DS2 package sends a request to the external API http://helloacm.com/api/fortune. This API returns a character string that contains escaped characters. The GETFORTUNE method uses the DS2 function TRANSTRN to modify these characters.

```
package "${PACKAGE_NAME}" /inline;
   dcl package http fortune_pkg();
   dcl varchar(1048576) character set utf8 http_response;
   dcl int rc;
   dcl int getMethodDefined;

   method execute(in_out varchar "fortune");
      if missing(getMethodDefined) then do; /* establish GET method */
         fortune_pkg.createGetMethod('http://helloacm.com/api/fortune');
         "getMethodDefined" = 1;
      end;

      "fortune" = 'Gloom and Doom';  /* default fortune */
      fortune_pkg.executeMethod();
      fortune_pkg.getResponseBodyAsString(http_response, rc);
      if (rc = 0) then do; /* clean up escaped characters in the response */
         "fortune" = transtrn(http_response,'\n',' ');
         "fortune" = transtrn("fortune",'\t',' ');
         "fortune" = transtrn("fortune",'\"','"');
      end;
   end;
endpackage;
```

Testing DS2 Code Files in SAS Studio

Note: If your custom code uses data grid variables, see “Working with Data Grids in SAS Studio” in *SAS Intelligent Decisioning: Using Data Grids*.

If your DS2 custom code file uses functions that are defined in a custom context file, insert the packages that are defined in the custom context file before the DS2 code file package.

Replace the placeholder name **`${PACKAGE_NAME}`** with a valid DS2 package name.

To test your package in a separate DS2 invocation from where the calling program is running, replace the `inline` modifier with `overwrite=yes`.

```
proc ds2;
   package "testCustomCode" /overwrite=yes;
      method execute(double l, double w, double h, double d,
                     in_out double vol, in_out double wgt);

         vol = l * w * h;
         wgt = vol * d;
      end;
   endpackage;
run;
quit;
```

In the following example, the package is compiled in the same PROC DS2 invocation as the DS2 data program that instantiates the package, so the code specifies the `/inline` package modifier.

```
/* Create test data. */
data work.testdata;
   length material $13;
   long=40; wide=20; high=10; density=0.098; material = 'aluminum'; output;
   long=20; wide=10; high=4;  density=0.284; material = 'iron';     output;
run;

proc ds2;

   /* Replace the placeholder name with a package name. */
   package "testCustomCode" /inline;
      method execute(double l,
                     double w,
                     double h,
                     double d,
              in_out double vol,
              in_out double wgt);

         vol = l * w * h;
         wgt = vol * d;
      end;
   endpackage;

/* Use a DS2 data program to execute the custom code. */
   data _null_;
      dcl package testCustomCode myCustomCode(); /* Instantiate the package. */
      dcl double volume;
      dcl double weight;

      method run();
         /* Read in the variables long, wide, high, and density. */
         set work.testdata(drop=(material));
         volume = .;
         weight = .;
         myCustomCode.execute(long, wide, high, density, volume, weight);
         put _all_;
      end;
   enddata;
run;
quit;
```

Python Code Files

About Publishing Destinations and Python Distributions

IMPORTANT You can publish decisions that contain Python code files only to SAS Micro Analytic Service destinations, SAS Cloud Analytic Services (CAS) destinations, and container destinations.

When you publish a decision that contains a Python code file, SAS Intelligent Decisioning generates a private DS2 PyMAS package. The package is assigned a random name. Your Python program is encapsulated inside a DS2 EXECUTE method. When the decision is executed, the DS2 process sends the Python program to a Python process to be executed.

In SAS Micro Analytic Service and CAS destinations, an executing decision uses the Python distribution that is installed at your site. In container destinations, executing decisions use the Python base image that is included in the published container. For information about the Python base image that is included in published containers, see "About a SAS Container Runtime Image" in *SAS Container Runtime: Programming and Administration Guide*. For information about the standard libraries that are distributed with specific versions of Python, see https://docs.python.org.

Rules For Developing Python Code Files

When you are developing your Python code, follow these rules:

- Your Python code must define an EXECUTE function. This function is the only public function allowed.
- An `Output:` docstring is required. This string must immediately follow the Python EXECUTE function declaration, and it must be indented within the EXECUTE function definition. In the docstring, list all of the output variables produced by the program. For example, if your program has two output variables named `prediction` and `probability`, your docstring would appear as follows:

  ```
  '''Output: prediction, probability'''
  ```

- If your Python code uses packages that are not built-in packages, list these packages in the `DependentPackages:` docstring. For more information, see "Required Dependencies For Python Files in SAS Container Runtime Destinations" on page 158.
- Optional input arguments to the Python EXECUTE function are not supported.

- You must specify the data types for the input and output variables for the EXECUTE function in your Python code on the **Variables** tab of the decision. The DS2 package code uses this information to resolve the data types for the variables. These variables are specified in the signature of the DS2 EXECUTE method. If you add a Python code file to a decision and you have not specified the variables on the **Variables** tab, SAS Intelligent Decisioning might display the following message: `Some objects in the decision define variables for which no corresponding decision variables have been created.` For more information, see "About Decision Variables and Mapping" on page 247.
- The Python EXECUTE function must return standard Python data types. For more information, see "Return Values " in *SAS Micro Analytic Service: Programming and Administration Guide*.
- You can import modules and define other functions and classes that are used by the Python EXECUTE function.
- Test your Python code by using a Python interpreter that is outside of SAS Intelligent Decisioning before you incorporate your code into a decision.

IMPORTANT After you test your code outside of SAS Intelligent Decisioning, incorporate it into a decision and test the decision. In order to run a test for a decision that contains a Python code file, you or an administrator must configure support for Python. See "Configuring Support for Python Code Files" in *SAS Intelligent Decisioning: Administrator's Guide* for more information.

For more information about using Python with DS2, see the following:

- "Python Support in SAS Micro Analytic Service" in *SAS Micro Analytic Service: Programming and Administration Guide*
- "Executing Python Modules in DS2 Modules" in *SAS Micro Analytic Service: Programming and Administration Guide*

TIP If you encounter errors that are unrelated to syntax or logic, check with your administrator to verify that Python is configured correctly. For additional information, see "Configuring Support for Python Code Files" in *SAS Intelligent Decisioning: Administrator's Guide*.

Required Dependencies For Python Files in SAS Container Runtime Destinations

If you plan to publish a decision that includes a Python code file to a SAS Container Runtime destination, and if your Python file uses packages that are not built-in packages, list these packages in the `DependentPackages:` docstring. List all of the package names on a single line. Use the official package names. Separate the package names with commas. For example:

```
'''DependentPackages: sklearn, requests, dateutil'''
```

When you publish the decision, SAS Intelligent Decisioning passes the name and `pip3` install command for each package to the model publish service. The model publish service uses this information to install the packages in the container that is published to the destination.

For information about the Python base image that is included in published containers, see "About a SAS Container Runtime Image" in *SAS Container Runtime: Programming and Administration Guide*.

Note: Do not specify package aliases or the names of built-in packages in this docstring. If you include the name of a built-in package or of a package alias, SAS Intelligent Decisioning writes the following message to the log when you publish the decision:

```
ERROR: No matching distribution found for name
```

Verify whether a package is built in to the Python distribution that will be used by the executing decision before you add a package name to this docstring. Verify that the docstring is correct by running a publishing validation test for your decision before you deploy your decision in a production environment. For information about the standard libraries that are distributed with specific versions of Python, see https://docs.python.org.

Data Query Files

Using Data Query Files

A query enables you to extract data from one or more tables according to criteria that you specify. When you create the query, you can choose which editor you want to use to create the file. Your choice of editor affects whether the data query can return scalar variables. For more information, see "Query Output Types and Editors" on page 160.

IMPORTANT You can publish decisions that contain data query files to SAS Micro Analytic Service destinations or to container destinations. For SAS Micro Analytic Service destinations, you can query only the data sources that are supported by SAS Micro Analytic Service. For more information, see "Data Sources Supported for Use with SAS Micro Analytic Service" in *SAS Micro Analytic Service: Programming and Administration Guide*.

In container destinations, Oracle is the only database that can be queried by using data query files.

In order to test decisions that contain data query files, an administrator must configure support for SQL query files. See "Configuring Support for Data Query Files" in *SAS Intelligent Decisioning: Administrator's Guide*.

IMPORTANT Tests that are running in SAS Intelligent Decisioning might encounter a significant performance impact. Pre-publish testing requires SAS Intelligent Decisioning to convert the SQL code to use HTTP protocol and instantiate a MAS module for each call to the SQL package. These actions incur a significant performance impact and create a limitation on the size of the results table. When you test a decision that uses a data query node, use an input data set that is as small as possible.

Query Output Types and Editors

When you create a new data query file, you can choose to create the file in the SQL editor or in SAS Studio.

Files that are created in the SQL editor can return either a data grid or a single row of scalar variables. You can select the output type on the **Properties** tab of the data query file.

Files that are created in SAS Studio are created as stand-alone queries. These files return a data grid. Do not change the output type for these files. For more information, see "Understanding the Differences between a Stand-Alone Query and a Flow Query" in *SAS Studio: User's Guide*.

For information about data grids, see *SAS Intelligent Decisioning: Using Data Grids*.

Developing SQL Code

When you are developing your SQL code, follow these rules:

- Data query nodes support SELECT, INSERT, UPDATE, and DELETE statements. They do not support any data definition language (DDL) statements such as ALTER or DROP that alter the structure of the table.
- Data query nodes support only the following data types: decimal, string, date, datetime, and integer. You must specify a length for string variables. You cannot specify a length for other variable types. Specify input variables with a question mark (?). Use the AS keyword to specify the input and output variables in the decision as aliases for the database column names. Enclose the decision variable specifications in braces ({}). For example, in the following SELECT statement, DEBTRATIO, CAUSE, and BADLOAN are decision variables. The variable BADLOAN is an input variable in the decision. This statement retrieves the values of the DEBTINC, REASON, and BADLOAN columns from the database and assigns their values to the decision variables.

  ```
  SELECT debtinc AS {:debtRatio:decimal},
         reason AS {:cause:string:8}
    FROM hmeq_test WHERE bad = {?:badloan:decimal}
  ```

  The following INSERT statement adds a row that contains the columns NAME and AMT to the table MYTABLE:

  ```
  INSERT INTO mytable (name, amt)
         values ({?:name:string:1000},{?:amt:integer})
  ```

Note: You can read data from native SQL date and datetime variables. However, do not rely on the accuracy of data that contains fractional seconds.

- Do not use an asterisk (`*`) to select database columns. SAS Intelligent Decisioning does not have any metadata about the table, so it cannot determine what columns are in the table.
- When you create a new data query file, SAS Intelligent Decisioning adds the comment `/* include sqlReturnInfo */` to the file. This comment enables SAS Intelligent Decisioning to define metadata for the query's variables. Do not delete this line if you want SAS Intelligent Decisioning to create all of the metadata. For more information, see "Decision Variables for Data Query Files" on page 161.
- When you create data query files in SAS Studio, you must specify the **PROC FEDSQL** output option on the **Output Options** tab in SAS Studio. For more information, see "Generating a FedSQL Query" in *SAS Studio: User's Guide*.

For additional information about developing SQL code, see the following topics:

- For information about the data sources that are supported by SAS Micro Analytic Service, see "Data Sources Supported for Use with SAS Micro Analytic Service" in *SAS Micro Analytic Service: Programming and Administration Guide*.
- For syntax information about SQL statements, see "FedSQL Statements" in *SAS FedSQL Language Reference*.
- For the list of reserved words in FedSQL, see "FedSQL Reserved Words" in *SAS Viya Platform: FedSQL Programming for SAS Cloud Analytic Services*.
- For information about the functions that are supported in SQL code, see "FedSQL Functions" in *SAS FedSQL Language Reference*.
- For information about using SAS Studio to create queries, see "Creating a Stand-Alone Query" in *SAS Studio: User's Guide* and "Using Macro Variables in SAS Studio" on page 162.

Decision Variables for Data Query Files

When you create a data query file, SAS Intelligent Decisioning automatically adds the comment `/* include sqlReturnInfo */`. When this comment is present and you add the data query file to a decision, SAS Intelligent Decisioning defines the metadata for decision variables for the return code, row count, and query output. If this comment is not present, SAS Intelligent Decisioning defines metadata for only the query output. You can use the **Add missing variables** option to add these variables to the decision and map these decision variables to the query's variables. You can set the **Create variables automatically in decisions** setting to automatically add these variables to the decision. For more information, see "About Decision Variables and Mapping" on page 247.

The variables are mapped as shown in Table 6.1. When the decision is run, the variables are created only if the query selects data.

Table 6.1 Variables for Query Files

Query File Variable	Decision Variable	Description
*file_name*_returnCode_out	returnCode	The return code that is generated by the query file.
*file_name*_rowCount_out	rowCount	The number of rows returned by the query.
*file_name*_out	dgo	The data grid variable that contains the output of the data query. The lengths of character columns in the data grid are determined by the lengths of the character variables that are specified in the SELECT statement. **Note:** This variable is created only if the query selects data.

If the comment is present but the query does not select data,SAS Intelligent Decisioning creates only the return code variables.

If you remove the comment and the query selects data, only the output variables are created.

If you remove the comment and the query does not select data, no variables are created.

Using Macro Variables in SAS Studio

When you author a data query file in SAS Studio, you can pass input data to the decision by using macro variables. When you add the data query file to a decision, SAS Intelligent Decisioning creates decision variables that have the same name as the macro variables. When the decision executes, the values of the decision variables are set to the same values as the macro variables.

To define the macro variable in SAS Studio, open a **SAS Program** tab and submit the %LET statement. For example, the following statement defines the macro variable &myName:

```
%let myname = 'Arinya'
```

To use the macro variable in the **Query** tab, reference it with an ampersand followed by the macro variable name:

```
SELECT name FROM table WHERE name = &myname
```

When you create a filter expression, set the appropriate options so that SAS Studio recognizes the macro variable. The settings depend on whether the variable is character or numeric.

- If the macro variable is numeric, check the **Allow macros** check box.
- If the macro variable is character, check the **Match Case** check box, and clear the **Quote Strings** check box.

You can use custom SQL code for your data query instead of using a quick filter. However, SAS Intelligent Decisioning cannot determine the variable's data type when it is creating the decision variable. SAS Intelligent Decisioning assumes that the variable is a numeric unless its name ends with `char` or `string`. The `char` and `string` suffixes are not case sensitive.

For more information, see "Filtering Data" in *SAS Studio: User's Guide* and "Macro Variables Defined by Users" in *SAS Macro Language: Reference*.

Handling Table and Column Names in Data Query Files

FedSQL performs case folding to make all table and column names uppercase if they are not enclosed in quotation marks. This case folding might cause issues with databases that perform case folding on unquoted references to make them lowercase. To avoid issues with running your query code in both SAS Studio and in SAS Intelligent Decisioning, do the following

- In SAS Studio, if you refer to a table column by using a table alias in a JOIN, WHERE, or HAVING clause, do not enclose the table alias in quotation marks. Alternatively, you can refer to the column by using only the column name or by using the full table name instead of the alias.
- When you define a libref in SAS Studio to connect to your PostgreSQL database, set the PRESERVE_COL_NAMES and PERSERVE_TAB_NAMES options on the LIBNAME statement to Yes. For additional information, see "PRESERVE_COL_NAMES= LIBNAME Statement Option" in *SAS/ACCESS for Relational Databases: Reference* and "PRESERVE_TAB_NAMES= LIBNAME Statement Option" in *SAS/ACCESS for Relational Databases: Reference*.
- Set the SAS Intelligent Decisioning configuration option quoteStudioQueryIdentifiers to On. For additional information, see "sas.decisions.codefiles.quoteStudioQueryIdentifiers" in *SAS Intelligent Decisioning: Administrator's Guide*.

Custom Context Files

Using Custom Context Files

A custom context file is a custom DS2 package file in which you can define context functions that you can use in rule sets, in DS2 code files, and in custom functions that are defined in the Custom Functions category. When SAS Intelligent Decisioning generates the code for a decision, each context function is defined only once in the generated code.

You can test or validate rule sets, DS2 code files, and custom functions that use context functions before you publish them. In a production environment, you can reference the functions that are defined in a custom context file only from decisions. Rule sets that are published independently (outside of a decision) cannot use context functions.

At run time, the custom context can hold information in internal variables about the record that is being processed. This information can be shared with other nodes by calling functions that are defined in the context file.

Custom context files that are specified in nested decisions are ignored. You must specify the custom context file in the parent decision. A nested decision cannot use a custom context file that is different from the custom context file that is used by the parent decision. The parent decision’s context file is used to resolve all references to context functions.

To create a custom context file, select **Custom context** for the file type in Step 4 of “Create a New Code File” on page 167.

Note: In container destinations, Oracle is the only database that can be queried by using custom context files that contain SQL queries.

Adding an INIT Method to a Custom Context File

You can add an INIT method to your custom context files. By adding an INIT method, you can initialize variables before the first row of data is processed by the decision. The INIT method is executed the first time the custom context file is executed for the decision. The method is executed only once per thread. For example, the following code creates a custom context file with an INIT method that initializes a hash:

```
package CustomContext;
   dcl varchar(1024) ctxtKey;
   dcl varchar(1024000) ctxtValue;
   dcl package hash context;
   method init();
      context = _new_ hash([ctxtKey],[ctxtValue],0, '', '', '', '', 'MULTIDATA');
   end;
   ...
endpackage;
```

Associating a Custom Context File with a Decision

You can specify a custom context file for decisions that are published to SAS Micro Analytic Service destinations and specify a different context file for decisions that are published to other destinations. You do not need to specify a file for both destination types unless the content differs between the two files.

1 Open the decision, and click the **Properties** tab.

2 Specify a custom context file in one or both of the following fields:

- **Custom context** specifies the context file to use when the decision is published to SAS Cloud Analytic Services (CAS), Git destinations that are compatible with CAS, Teradata, Apache Hadoop, or container destinations. You must specify the custom context file in this field in order to run scoring tests, scenario tests, and publishing validation tests on the decision.
- **Custom context for SAS Micro Analytic Service** specifies the context file to use when the decision is published to SAS Micro Analytic Service destinations and Git destinations that are compatible with SAS Micro Analytic Service. Specify the context file in this field if you plan to publish the decision to one of these destinations.

3 Select the version of the custom context file.

Testing or Validating a Rule Set, DS2 File, or Custom Function

To test or validate rule sets, DS2 code files, or custom functions that use context functions, you must specify the custom context file in the **Test custom context** field on the **Properties** tab before you test or validate the object. For instructions, see "Validate SQL, DS2, or Context Files" on page 175 and "Testing Code Files" on page 187.

Referencing a Function That Is Defined in a Context File

When you add a custom context file to an object's properties, SAS Intelligent Decisioning defines a variable in the generated code for the object that is named `application`. This variable appears only in the generated code. It does not appear in the **Variables** tab for the object.

You must use the `application` variable to refer to functions that are defined in the custom context file. To call a custom context function from within a rule set, DS2 code file, or custom function, use the following format:

```
application.method_name(arguments);
```

For example, you could create a custom context file that defines a custom function named squareIt:

```
package CustomContext;
   method squareIt(in_out double i) returns double;
     dcl double squared;
     squared=i*i;
     return squared;
   end;
endpackage;
```

In a rule set, you could call the squareIt function as follows:

```
mySquare = application.squareIt(i)
```

In the Custom Functions category, you can define a custom function that calls the squareIt context function as follows:

```
method addSquares() returns double;
    dcl double tally;
    dcl double squared;
    dcl double i;
    tally = 0;
    do i= 1 to 5;
       squared = application.squareIt(i);
       tally + squared;
    end;
    return tally;
end;
```

Referencing a Variable That Is Defined in a Context File

When you add a custom context file to an object's properties, SAS Intelligent Decisioning defines a variable in the generated code for the object that is named `application`. This variable appears only in the generated code. It does not appear in the **Variables** tab for the object.

You must use the `application` variable to refer to variables that are defined in the custom context file. To refer to a custom context variable from within a rule set, DS2 code file, or custom function, use the following format:

`application.`*`variable_name`*

For example, you could create a custom context file that sets the value of a variable named `grade` based on the value of the variable `score`:

```
package CustomContext;
   dcl varchar(20) grade;
   method execute(in_out double score);
     if score >= 90 then grade='A';
     else if score >= 80 then grade='B';
     else if score >= 70 then grade='C';
     else if score >= 60 then grade='D';
     else grade='F';
   end;
endpackage;
```

In a rule set, you can execute the custom context file and set the value of the rule set variable `finalGrade` to the value of the custom context variable `grade` as follows:

```
application.execute(score);
finalGrade=application.grade;
```

Create a New Code File

1. Click 🗎 to navigate to the Code Files category view.
2. Click **New Code File**. The New Code File window appears.
3. Enter a name for the new code file.

   Note: When a data query file generates output, the file name can be up to 28 characters long. When the data query file is added to a decision, SAS Intelligent Decisioning creates an output-only decision variable. The name of the decision variable is the data query file name plus `_out`.

4. Select the code file type.
5. (Optional) For data query file types, select the editor with which you want to edit the file.

   TIP For data query files that you choose to edit in SAS Studio, SAS Intelligent Decisioning adds `.cqy.df` to the file name, and the file type is displayed as **Data Query**.

   For data query files that you choose to edit in the SQL editor, the file type is displayed as **SQL**. These files can return either a data grid or a single row of scalar variables. You can select the output type on the **Properties** tab.

   For more information, see "Query Output Types and Editors" on page 160.

6. (Optional) Enter a description for the file. Descriptions are limited to 1000 characters.

   TIP You can edit the description later on the **Properties** tab.

7. Click 🗀, and select the folder where you want to save the file.
8. Click **Save**. SAS Intelligent Decisioning opens the new code file.

   For data query files that you choose to edit with SAS Studio, SAS Intelligent Decisioning displays the **Properties** tab, and you can click **Open Data Query** to open the file in SAS Studio.

   For all other file types, SAS Intelligent Decisioning uses the appropriate code editor to open the file on the **Code** tab.

TIP Except for query files that you edit in SAS Studio, you must check out objects that are saved in a folder for which the check-out and commit feature is enabled before you can edit them. The check-out and commit feature is always enabled for the `Decision Repository` folder. See "Checking Out and Committing Decision Versions" on page 272 for more information.

9 (Optional) If the code file is in a folder for which the check-out and commit feature is enabled, click the **Versions** tab and check out the latest version of the code file. For more information, see "Check Out and Commit a Custom Code File Version" on page 184.

TIP By default, the autocomplete feature is turned on for SAS functions and custom functions. You can turn off this feature or change what is displayed in the autocomplete list by using the SAS Intelligent Decisioning settings. You can specify that SAS Intelligent Decisioning also displays variable names. For information, see "SAS Intelligent Decisioning Settings" on page 8.

In the autocmplete list, SAS Intelligent Decisioning displays all function names and variable names that match the text that you enter. Double-click an item to add it to the code field. Single-click an item to display information about the item, such as whether it is a function or a variable. For functions, the information includes the returned data type and category. For variables, the information includes the data type and direction (input, output, or both).

The autocomplete list does not include functions that are defined in a custom context file.

Managing the Variables in a DS2 or Python Code File

About Variables

The Properties of a Variable

Property	Description
Name	Variable names must start with a letter or an underscore (_), and they can contain only alphanumeric characters and the underscore.

Property	Description
	Variable names can be up to 32 characters long. They must be unique within a code file. **Note:** SAS Intelligent Decisioning does not support double-byte character set (DBCS) characters in variable names. **Note:** Do not use any of these operators or keywords as variable names: AND, OR, IN, NOT, LIKE, TRUE, or FALSE. Do not use _N_ or any DS2 reserved word as a variable name. See "Reserved Words in the DS2 Language" in *SAS DS2 Programmer's Guide* for information about reserved words in the DS2 language.
Data type	Code files in SAS Intelligent Decisioning support the following data types: Boolean, character, data grid, date, datetime, decimal, and integer. For Boolean variables, you can select `True` or `False` for the initial value. However, SAS Intelligent Decisioning represents Boolean values by using the numbers 1 and 0 in the code that it generates. When SAS Intelligent Decisioning is evaluating Boolean expressions, any non-zero number is considered True. When you are entering expressions, specify `1` for True and `0` for False. **Note:** For content that will be published to a SAS Micro Analytic Service destination, the values of date and datetime input variables must be numeric. They cannot be formatted values such as `18jul2019`. Also, integer variables are converted to decimal variables when the content is published.
Input and **Output**	A variable in a DS2 code file can be an input variable, an output variable, or both. A variable in a Python code file can be an input variable or an output variable, but not both. See "Input Variables and Output Variables" on page 170 for more information.
Length	The length for Boolean and numeric variable types is set automatically. For output-only data grid variables, the length is set to the value that you specify. For output-only character variables in decisions that are published to container destinations or to SAS Micro Analytic Service destinations, the length that you specify is ignored unless the sas.decisions.variable.length.honorOutputLengthInMAS configuration option is turned on. For more information,see "sas.decisions.variable.length" in *SAS Intelligent Decisioning: Administrator's Guide*. For character variables and data grid variables that are input-only or input-output variables, the variableLengthOverridden configuration property determines how the length is set. By default, this property is set to Off, and the length is set to the length in the input data. When the variableLengthOverridden property is set to On, the length of input-only and input-output character variables and data grid variables is set to the value that you specify. For more information, see "sas.decisions.variableLengthOverridden" in *SAS Intelligent Decisioning: Administrator's Guide*.

Property	Description
	The maximum length for character variables (outside of a data grid) and data grid variables is 10485760 characters. The maximum length for character variables within a data grid is 32767 characters.
Initial value	You can specify an initial value for all data types except data grids. Initial values are used only when you test a code file in SAS Intelligent Decisioning. **Note:** For content that will be published to a SAS Micro Analytic Service destination, the values of date and datetime input variables must be numeric. They cannot be formatted values such as `18jul2019`. Also, integer variables are converted to decimal variables when the content is published.
Description	Descriptions are limited to 256 characters.

Input Variables and Output Variables

For each variable in an SQL query, a DS2 method signature, or a Python function signature, you must specify whether the variable is an input variable, an output variable, or, in the case of DS2 files, both an input variable and an output variable.

- Input variables are variables that are present in the input table for a decision. When a decision that uses a code file is deployed in a production system, all input variables must be mapped to table columns in input data. When you test a code file in SAS Intelligent Decisioning, for each input variable, you can either map it to a table column or specify a constant as its input value. If you choose not to map a variable to either a table column or a static value, SAS Intelligent Decisioning displays a warning message. When you create or edit a variable, clear the **Input** check box for any variable that you do not want to be mapped to a column in an input table or for which you do not want to specify a value.
- Output variables are variables that are written to the output table that is created when a decision that uses a code file is run. When you create or edit a variable, clear the **Output** check box for any variable that you want to exclude from the output data.

When you create a new variable, it is created as an output-only variable by default.

Syncing Variables and Code

To propagate changes that you enter on the **Variables** tab to the code that is shown on the **Code** tab, click **Sync to Code** on the **Variables** tab. For Python code files, changes to input variables are propagated to the signature of the EXECUTE function, and changes to output variables are propagated both to the docString and to the RETURN statement. For DS2 files, changes to input-only, output-only, and input-and-output variables are propagated to the signature of the EXECUTE method. Both output-only and input-and-output variables are added with the `in_out` keyword.

To propagate changes that you enter on the **Code** tab to the **Variables** tab, save the code file or click **Sync Variables** on the **Code** tab. For Python code files, changes to the signature of the EXECUTE function are propagated to the **Variables** tab as changes to input variables. Changes to the docString and to the RETURN statement are propagated as changes to output variables. New variables are added as character variables, but you change the data type on the **Variables** tab. For DS2 files, variables that you add to the signature of the EXECUTE method with the `in_out` keyword are added to the **Variables** tab as output-only variables. Variables that are not specified with the `in_out` keyword are added to the **Variables** tab as input-only variables. You can specify that any variable is both an input and output variable by selecting both the **Input** and **Output** check box on the **Variables** tab.

If you make conflicting changes to both the **Code** tab and the **Variables** tab and then save the file, SAS Intelligent Decisioning prompts you to select which tab contains the changes that you want to retain. If you make conflicting changes on the two tabs and then click a sync button, only the changes on the tab on which you click the sync button are retained. Those changes are then propagated to the other tab.

Note: You cannot manage local variables on the **Variables** tab. Local variables are variables for which neither the **Input** nor the **Output** checkboxes are selected on the **Variables** tab. If local variables are listed on the **Variables** tab, these variables are removed from the tab when you save the code file or click a sync button.

Add Variables from a Data Table

1 On the **Variables** tab, click **Add variable** and select **Data table**. The Choose Data window appears, and the list of SAS Cloud Analytic Services (CAS) tables that are loaded into memory is displayed on the **Available** tab.

   If the table that you need does not appear in the list of available tables, try the following solutions:

   - If the table appears on the **Data Sources** tab, right-click on the table, and select **Load** to load the table into memory. If the table does not appear on the **Available** tab, click ↻.
   - If the table does not appear on the **Data Sources** tab, import the data. The process of importing the data loads it into memory. For information about importing data from different sources, see “Making Data Available to CAS” in *SAS Data Explorer: User’s Guide*.

2 Select the table from which you want to import variables, and click **OK**. The Add Variables window appears.

3 Select the variables that you want to import and click **+›**. To import all of the variables in the table, click **+»**.

4 Click **Add** to add the select variables, or click **Add and replace** to replace existing variables that have the same name.

5 On the **Variables** tab, select or clear the **Input** and **Output** check boxes as necessary. See "Input Variables and Output Variables" on page 170 for more information.

> **TIP** To filter the variable list, right-click on the **Variable** column heading, enter a filter string in the **Filter** box, and press Enter. To clear the filter, delete the string from the **Filter** box, and press Enter.

6 Save the code file or click **Sync to Code** to add the variables to the code that is on the **Code** tab. For more information, see "Syncing Variables and Code" on page 203.

Add Variables from a Rule Set, Decision, or Code File

1 On the **Variables** tab, click **Add variable**, and select **Rule set**, **Decision**, or **Code file**. The Choose an Item window appears.

2 Select the object from which you want to import variables, and click **OK**. The Add Variables window appears.

3 Select the variables that you want to import and click **+›**. To import all of the variables in the table, click **+»**.

4 Click **Add** to add the selected variables, or select **Add and replace** to replace existing variables that have the same name.

5 On the **Variables** tab, select or clear the **Input** and **Output** check boxes as necessary. See "Input Variables and Output Variables" on page 170 for more information.

6 Save the code file or click **Sync to Code** to add the variables to the code that is on the **Code** tab. For more information, see "Syncing Variables and Code" on page 203.

Create Custom VariablesCreate Custom Variables

Note: For information about data grid variables, see "Defining Data Grid Variables" in *SAS Intelligent Decisioning: Using Data Grids*.

To create custom variables on the **Variables** tab:

1 Click **Add variable** and select **Custom variable**. The Add Variables window appears.

2 Complete these steps for each variable that you want to add:

a. Enter the name of the new variable, and select the data type of the variable. See “The Properties of a Variable” on page 168 for additional information.

   > **TIP** To re-add a variable that was used in a locked version, you must specify the same data type that was used in the previous version.

b. (Optional) Click **Optional** to display the **Description**, **Initial value**, and **Length** fields.

c. (Optional) Enter a length, initial value, and description for the new variable. Whether you can specify an intial value or length for the variable depends on the variable's data type. See “The Properties of a Variable” on page 168 for additional information.

d. Click **Add**. SAS Intelligent Decisioning adds the new variable to the table of variables at the bottom of the window. By default, variables are added to the table as both input and output variables.

e. Verify that the **Input** and **Output** check boxes are selected correctly for each variable.

   - Clear the **Input** check box for any variable that you do not want to be mapped to a column in an input table or for which you do not want to specify a value.
   - Clear the **Output** check box for any variable that you want to exclude from the output data.
   - Clear both the **Input** and **Output** check boxes to create a temporary variable.

   > **TIP** Variables in DS2 files can be both input and output variables but Python variables cannot.

3 (Optional) Modify the variable properties in the table of variables.

4 Click **OK** to add the variables and close the Add Variables window.

5 Save the code file or click **Sync to Code** to add the variables to the code that is on the **Code** tab. For more information, see “Syncing Variables and Code” on page 203.

Duplicate a Variable

1 On the **Variables** tab, select the variable that you want to duplicate, click ⋮, and select **Duplicate**. The Duplicate Variable window appears.

2 Enter a new name for the duplicate variable.

3 (Optional) Enter a description for the variable.

4 Click **Duplicate**.

5 Save the code file or click **Sync to Code** to add the variables to the code that is on the **Code** tab. For more information, see "Syncing Variables and Code" on page 203.

Importing and Exporting Variables

Import Variables

You can import variables from either comma-delimited (CSV) files or from JavaScript Object Notation (JSON) files. These files must conform to formats described in Appendix 2, "Import File Formats," on page 325. These formats are the same formats that are created when you export variables.

1 Open the code file into which you want to import variables.

2 On the **Variables** tab, click **Import**, and select either **Comma-delimited (*.csv)** or **JSON (*.json)**. The Import File window appears.

3 Click **Browse** and select the file from which you want to import variables.

4 Specify the encoding of the CSV file.

5 Select **Add variables** to append the variables to the current list of variables, or select **Replace variables** to replace the current list of variables with the imported variables.

> TIP To add new columns to an existing data grid variable, select **Replace variables**.

6 Click **Import**, and then click 🖫 to import the variables and save the code file.

Export Variables

1 Open the code file from which you want to export variables.

2 On the **Variables** tab, select the variables that you want to export.

3 Click **Export**, and select the file type to which you want to export the variables. The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Delete Variables

1 On the **Variables** tab, select the check box for the variables that you want to delete, click ⋮ , and select **Delete**.

2 Click **Sync to Code** to delete the variables from the code that is on the **Code** tab.

Edit Variable Properties

On the **Variables** tab, click on the name of the variable that you want to edit. The Edit Variable window appears. Edit the properties as needed, and then click **OK**.

Note: When you rename a variable, click **Sync to Code** to change references to that variable within the same object change to use the new name.

See “The Properties of a Variable” on page 168 for additional information.

Edit Metadata for Data Grid Variables

For information, see “Editing Data Grid Variable Metadata” in *SAS Intelligent Decisioning: Using Data Grids*.

Validate SQL, DS2, or Context Files

To validate a code file of type SQL, DS2, or custom context, open the file and click **Validate**. For DS2 code files that use context functions, specify the custom context file in the **Test custom context** field on the **Properties** tab.

For DS2 code files and custom context files, SAS Intelligent Decisioning compiles the code.

IMPORTANT Before you validate a DS2 code file or custom context file that uses a data grid package, verify that the declaration statements for the package specify `datagrid` instead of `dcm_datagrid`. For example:

```
in_out package datagrid variable;
```

```
dcl package datagrid variable;
```

For SQL query files, SAS Intelligent Decisioning runs the query in order to verify the SAS Micro Analytic Service connection string, scans the SQL statements for syntax errors, and looks for references to non-existent tables and columns.

IMPORTANT Do not validate a query that uses dynamic parameters to alter data. In order to validate a query, SAS Intelligent Decisioning generates values to use for the query parameters and executes the query. These parameter values might not match the values that are passed to the query during normal execution and can produce unintended changes in the database. For example, data might be deleted that should not be deleted.

If your query does not use dynamic parameters but does alter the data, be aware that validating the query will alter the data.

Copy a Code File URL

To create a link for external documentation that automatically opens a code file in SAS Intelligent Decisioning:

1. Open the code file.
2. Click ⋮, and select **Copy code file URL**. The Copy URL window appears, and the URL is automatically selected.
3. Click **Copy**, and then click **Close**.

   Paste the link into your documentation.

Compare Generated Code for Code Files

You can compare the generated code of two different code files, or you can compare the generated code of two different versions of the same code file.

1. Select the objects that you want to compare.
   - To compare the generated code of two different code files, select the code files in the category view, click ⋮, and select **Compare code**.
   - To compare the generated code of two versions of the same code file, open the code file, click ⋮ on the **Versions** tab, and select **Compare code**.

   The Select Versions window appears.

2 Select the versions that you want to compare, and click **Compare**. SAS Intelligent Decisioning opens the two objects side-by-side in the Compare Code window and highlights the differences.

3 (Optional) Click **Export** to export the results of the comparison to a PDF file. The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Managing Code Files

Duplicate Code Files

Note: You cannot duplicate a code file if it is open.

To duplicate a single code file:

1 In the Code Files view, select the code file that you want to duplicate.

2 Click ⋮ and select **Duplicate**. The Duplicate Code Files window appears.

3 Enter a new name for the duplicate code file.

4 (Optional) Enter a description for the code file.

5 Click 🗀, and select the folder where you want to save the duplicate file.

6 Click **Duplicate**.

To duplicate multiple code files:

1 In the Code files view, select the code files that you want to duplicate.

2 Click ⋮ and select **Duplicate**. SAS Intelligent Decisioning duplicates the code files and appends `_Copy` to the names of the duplicate copies. If needed, a number is also appended to the names of the duplicate copies.

Note: When you duplicate a code file, SAS Intelligent Decisioning creates a relationship between the original code file and the duplicate code file. If either object is changed, and you later copy the contents of one object into the other, SAS Intelligent Decisioning displays a warning message. Verify that you want to replace the current contents of the code file before you paste the new content.

Delete Code Files

In the Code Files category view, select the code files that you want to delete, click ⋮, and select **Delete**. SAS Intelligent Decisioning moves the files to the recycle bin. For more information, see “Manage Folders and Folder Content” on page 9.

Rename a Code File

Note: You cannot rename a code file if it is open.

1 In the Code Files category view, select the code file that you want to rename.

2 Click ⋮ and select **Rename**. The Rename window appears.

3 Enter a new name for the code file, and click **Rename**.

Move Code Files to a Different Folder

1 In the Code Files category view, select the code files that you want to move.

2 Click ⋮ and select **Move**. The Choose a Location window appears.

3 Select the location to which you want to move the code files, and click **OK**.

Managing Versions of Code Files

Set the Displayed Version

The displayed version is the version whose information is displayed on the other tabs, such as the **Code**, **Properties**, and **Variables** tabs. On the **Versions** tab, a ✓ in the **Displayed Version** column indicates the displayed version. To change the displayed version, click the version number for the version that you want to view. The displayed version is shown in the title bar.

Create a New Version

Note: For objects that are stored in locations for which the check-out and commit feature is enabled, you cannot manually create a new version. The only way to create a new version is to check out an existing version and commit a new version. For information, see "Check Out and Commit a Custom Code File Version" on page 184.

Note: The current version of an object is the version with the highest version number. When you create a new version, SAS Intelligent Decisioning locks the current version before it creates the new version.

IMPORTANT You cannot unlock a locked version. You cannot save changes to a version that is locked. If you modify a version that is locked and click 🖫, SAS Intelligent Decisioning asks you if you want to replace the current unlocked version with your edited version.

To manually create a new version:

1. On the **Versions** tab, click the version number for the existing version that you want to use as the basis for the new version.
2. Click **New Version**. The New Version window appears.
3. Select the version type: **Minor** or **Major**. Version numbers follow the format *major.minor*. If you select **Major**, the number to the left of the period is incremented. If you select **Minor**, the number to the right of the period is incremented.
4. (Optional) For each version tag that you want to add, either enter a custom tag or begin typing and select a tag from the list of previously entered tags. Press Enter after each tag.

   Note: A tag is limited to 100 characters.

   TIP All tags that have been previously entered for any object are saved in the list that is displayed when you start typing. You cannot delete tags from this list.
5. (Optional) Enter information about the new version in the **Notes** field.

   TIP You can edit these notes at any time on the **Versions** tab.
6. Click **Save**.

Copy the Content of a Version

You can copy the content of an object's version in the category view or on the **Version** tab for the object.

1 In the category view, complete these steps:

   a Select the code file whose contents you want to copy.

   b Click ⋮ , and select **Copy version**. The Copy Version window appears.

   c Select the version whose contents you want to copy.

   Alternatively, on the **Versions** tab of the code file whose contents you want to copy:

   a Select the version whose contents you want to copy.

   b Click ⋮ , and select **Copy version**. The Copy Version window appears.

2 Click 🗀, and select the target code file into which you want to paste the contents of the version.

   When you paste the contents, SAS Intelligent Decisioning creates a new version of the target code file. The target object contains only the pasted content.

3 Select whether you want to create a new major or minor version.

4 (Optional) Modify the notes that will be associated with new version.

5 (Optional) Add tags that will be associated with the new version. Tags that are associated with a source object version are not automatically added to the new version. See "Add a Version Tag" on page 186

6 Click **Paste Version**, and then click **Yes**.

Note: When you copy the contents of a source object into a target object, SAS Intelligent Decisioning creates a relationship between the two objects. If the source object is modified after you copy its contents, and you later copy the contents of the target object back into the source object, SAS Intelligent Decisioning displays a warning message. Verify that you want to replace the current contents of the source object before you paste the new content.

Delete a Version

IMPORTANT When you delete a specific version, that version is deleted permanently. It is not moved into the recycle bin, and it cannot be restored.

Note: In order to be able to delete a specific version of an object, you must have permission to delete the object itself. Also, the configuration option sas.decisions.codeFiles.deleteVersions must be turned on.

On the **Versions** tab, select the version that you want to delete, click ⋮, and select **Delete**.

You cannot delete the current version.

Upgrade Decisions to Use a New Version of a Code File

If you create a new version of a code file that is already used in other decisions, you can upgrade the decisions to use the new version.

1 On the **Versions** tab for the code file, click ⋮, and select **Upgrade decisions**. The Upgrade Decisions window appears. This window lists all of the decisions that include the code file

2 In the **Version to upgrade to** field, select the version of the code file to which you want to upgrade the decisions.

3 Select **Automatically map variables** if you want SAS Intelligent Decisioning to automatically map new code file variables in the decisions where the code file is used.

   For information about how SAS Intelligent Decisioning maps variables, see "About Decision Variables and Mapping" on page 247.

4 Select the check boxes for the decisions that you want to upgrade, and click **Upgrade Decisions**.

TIP To update all of the objects that are used in a decision, see "Update Decisions to Use New Object Versions" on page 253.

Determine Which Objects Use a Code File

To list the objects that use a specific code file:

1 On the **Code files** category page, select the check box for the code file, click ⋮, and select **View used by report**. The All Objects that Use the Selected Item window appears. This window lists all objects that use any version of the selected code file.

2 (Optional) Select a specific version of the code file. SAS Intelligent Decisioning narrows the list to include only the objects that use the selected version of the code file.

Note: The **View used by report** option is also available from within an open code file.

In the report, you can use the **Filter** field to filter the list of objects based on the object names. Click on an object name to open the object. Click ⓘ next to an object name to display the date on which the object was last modified and the ID of the user who modified the object. For decisions and code files that have been checked out, SAS Intelligent Decisioning also displays the IDs of the users that have checked out the objects. For decisions, SAS Intelligent Decisioning includes the decision's workflow status.

Click **Export** to export the Used By report to a PDF file. In the PDF file, you can click on an object name to open the object in a new browser tab.

Checking Out and Committing Code File Versions

About Checking Out and Committing Versions

Your administrator can enable the check-out and commit feature for custom code files that are in any folder by specifying the folder in the sas.decisions.checkout.checkoutEnabledFolderPaths configuration option. Enabling this feature for a folder does not automatically modify the permissions for the folder or for the objects in it. You can still modify a code file in the folder without checking it out, but you are expected to check out the latest version before you edit it. However, your administrator might also set permissions that require you to check out code files in these folders before you can edit them. For more information, see "sas.decisions.checkout" in *SAS Intelligent Decisioning: Administrator's Guide* and "Set Permissions for Check-Out Folders" in *SAS Intelligent Decisioning: Administrator's Guide*.

By default, SAS Intelligent Decisioning defines a folder where you can store code files that must be checked out before they can be edited. This folder is the `Decision Repository` folder, and it is the default value for the sas.decisions.checkout.checkoutEnabledFolderPaths configuration option. The default permissions for this folder require that non-administrative users check out a version and commit their changes to the checked-out version. Users who do not have administrative permissions cannot edit the code files in `Decision Repository` without first checking them out.

IMPORTANT When you create a custom code file in a folder for which the check-out feature is enabled, you must check out the code file before you can edit it in the SAS Intelligent Decisioning user interface. However, custom code files are stored in the Files service. The files service grants the file creator full access to the file. The file creator can use the REST API to edit the file without checking it out. The file creator can do this even when the file is in a folder for which the check-out feature is enabled (including the `Decision Repository` folder). In order to prevent the file from being edited without being checked out, the user ID of the file creator must be different from any user ID that is used to edit the file. For example, an administrator can create an empty code file that other users can then edit if they check it out.

If a version can be or must be checked out before it is modified, the **Check Out** button appears at the top of the **Versions** tab for that object. You can check out any version of an object. You can check out only one copy of a version at a time.

TIP If the sas.decisions.checkout.allowConcurrentCheckout option is turned off, and a user has checked out a code file version, the **Check Out** button for that code file is disabled for all other users. For more information, see "Concurrently Checking Out and Committing Code File Versions" on page 184.

When you check out a version, SAS Intelligent Decisioning writes a working copy of the version into your `My Folder` folder and opens the working copy. SAS Intelligent Decisioning adds "`(Checked Out)`" to the name that is displayed at the top of the window.

While you have a version checked out, the Code Files category view shows two code files with the same name, but the folders listed in the **Location** column differ for each file. The original version is in the location specified by the sas.decisions.checkout.checkoutEnabledFolderPaths configuration option, and the checked-out copy is in your `My Folder` folder.

TIP If an object that you have checked out does not appear in the category view, click ↺ to refresh the category view.

A **Commit** button appears at the top of the **Versions** tab for the checked-out version. When you are finished editing the checked-out version, you must commit your changes in order for other users to be able to see them. When you commit your changes, SAS Intelligent Decisioning creates a new version with your changes.

If the parent object is deleted before you commit your changes, you will not be able to commit your changes.

Checking Out a Code File from within a Decision

See "Checking Out and Committing Objects from within A Decision" on page 274 for information.

Concurrently Checking Out and Committing Code File Versions

The ability for multiple users to check out the same code file at the same time is controlled by the sas.decisions.checkout.allowConcurrentCheckout configuration option. This option is turned on by default.

When this option is turned on, different users can check out the same version of the same object at the same time. Because the objects that are checked out are saved in each user's `My Folder` location, the default permissions allow individual users to see only the copies that they have checked out.

When this option is turned off and a user has checked out an object, the **Check Out** button for that object is disabled for all other users.

If multiple users check out the same version of the same object at the same time, each user's changes are preserved in a new version when they commit their changes. One user's changes do not overwrite another user's changes.

IMPORTANT If two users attempt to commit changes to the same object simultaneously, the first user's attempt will succeed but the second user might see an error message that the commit has failed. If the second user subsequently commits their changes, the **Modified By** column on the **Versions** tab for both the version committed by the first user and the version committed by the second user displays the user ID of the second user.

Check Out and Commit a Custom Code File Version

1 On the **Versions** tab, click **Check Out**.

   SAS Intelligent Decisioning updates the **Properties** tab to indicate that the version is checked out.

2 Modify the checked-out version as needed, and save it.

   TIP To discard the changes and delete the checked-out version from `My Folder`, you can commit the object without saving it first. However, committing the object without saving creates a new version of the object whose contents match the contents of the previous version. For information on undoing a check out, see "Undoing a Check Out" on page 185.

3 On the **Versions** tab, click **Commit**. The Commit Code File VersionCommit Code File Version window appears.

4 Select the version type: **Minor** or **Major**. Version numbers follow the format *major.minor*. If you select **Major**, the number to the left of the period is incremented, and the minor number is reset to zero. If you select **Minor**, the number to the right of the period is incremented.

5 (Optional) In the **Version tags** field, enter any version tags that you want to associate with the new version. Press Enter after each tag. Tags are limited to 100 characters each and cannot contain the following characters: $ ' " # & ? ( ) \ /

6 (Optional) Enter information about the new version in the **Notes** field.

7 Click **Commit**. SAS Intelligent Decisioning creates a new version with your changes, and deletes the working copy from your `My Folder` folder.

Determine Who Has a Version Checked Out

If the current version of an object is checked out, the IDs of the users that checked it out and the timestamps when each user checked it out appear in the **Checked out by** field on the **Properties** tab for the original object. You can also display this information by clicking ↗ beside the version number on the **Versions** tab.

Opening the Original Object

When you check out an object, SAS Intelligent Decisioning adds the field **Original object link** to the **Properties** tab for the checked-out object. This field contains a link to the original object that was checked out. You can use this link to verify that you have checked out the correct version and to compare the original content with the modified content in the checked-out version.

Undoing a Check Out

How you undo the checkout of an object depends on how the object was checked out.

If both an object and a decision that uses the object are checked out at the same time, or if you checked out the object from within the decision, click ⋮ on the object's node in the decision diagram, and select **Cancel checkout**.

You can discard a checked-out version and any changes that you made by deleting the working copy of the version from your `My Folder` folder if the following conditions are true:

- You have not checked out a decision that uses the object.
- The object was not checked out at the same time as a decision that uses the object, or the object was not checked out from within the decision after the decision was checked out.

The deleted version is moved to the recycle bin. See “Delete Code Files” on page 178.

Managing Version Tags for Code Files

Add a Version Tag

Version tags enable you to better organize and group your content. Version tags are associated with specific versions of an object and not with the entire object. You can add the same tag to any version of any object. To add a tag to a code file version:

1. On the **Versions** tab, position your cursor in the **Version Tags** column for the version that you want to tag. If the version is not tagged, + appears. If the version has at least one tag, ✎ appears.
2. Click + to open the New Version Tags window, or click ✎ to open the Edit Version Tags window.
3. For each tag that you want to add, either enter a custom tag or begin typing and select a tag from the list of previously entered tags. Press Enter after each tag. Tags are limited to 100 characters each and cannot contain the following characters: $ ' " # & ? ( ) \ /

   TIP All tags that have been previously entered for any object are saved in the list that is displayed when you start typing. You cannot delete tags from this list.

   TIP To filter the version list based on a tag, right-click on the **Version Tags** column heading, enter a filter string in the **Filter** box, and press Enter. To clear the filter, delete the string from the **Filter** box, and press Enter.

4. Click **Close** to close the window.

Remove a Version Tag

1. On the **Versions** tab, position your cursor in the **Version Tags** column for the version whose tag you want to remove, and click ✎.
2. Click × beside the tag that you want to remove.

3 Click **OK** to close the window. The tag remains in the list of previously entered tags that is displayed when you add a tag, but the tag is no longer associated with version.

Modify a Version Tag

You cannot modify a version tag that already exists. To change the content of an existing tag, delete the tag as described in "Remove a Version Tag" on page 186, and then add the tag again as described in "Add a Version Tag" on page 186.

Testing Code Files

Ways to Test a Code File

There are two types of tests for code files:

Basic test
: executes the code file in the SAS Cloud Analytic Services (CAS) destination using the input table that you specify. You can also specify a debugging variable. For more information, see "Create and Run a New Test" on page 188.

Scenario test
: enables you to enter specific input values and the output values that you expect the test to generate. A scenario test identifies differences between the output that you expect to see and the actual output that is generated when the test is run. You can also compare the test definitions and test results of different scenarios. Scenario tests are also run in CAS. For more information, see "Test a Scenario" on page 289.

Note: You can create and run basic tests and scenario tests for DS2, SQL, and Python custom code files, but you cannot run these tests directly on custom context files. You can test the functions that are defined in a custom context file by testing the rule set or DS2 file that uses the context functions. Validating a custom context file determines whether the file is syntactically correct and whether it includes declaration statements for the packages that are used in the file. For more information, see "Validate SQL, DS2, or Context Files" on page 175.

Test a Code File

Create and Run a New Test

Testing a code file is optional, but doing so is a best practice. Testing enables you to discover any problems before the code file is incorporated into a production system.

IMPORTANT If you are testing a DS2 or SQL code file that uses functions that are defined in a custom context file, verify that the context file is specified in the **Test custom context** field on the **Properties** before running the test.

1 On the **Scoring** tab, click the **Tests** tab.

2 Click **New Test**. The New Test window appears.

3 Enter a name for the test if you do not want to use the default name. The name cannot contain forward slashes (/) or curly braces ({}).

4 (Optional) Enter a description for the test. Descriptions are limited to 1000 characters.

5 (Optional) Click 🗀 for the **Location** field, and select the folder where you want to save the test definition and results.

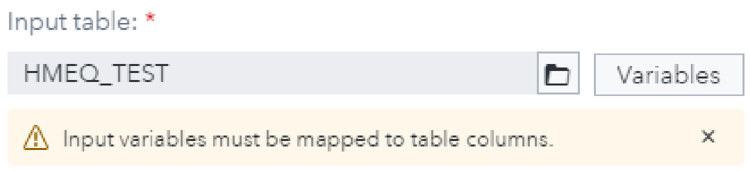
   TIP Selecting a location is optional, but it is highly recommended. Storing test definitions and test results in a folder simplifies the tasks of setting permissions and transferring the test files.

6 Click 🗀 for the **Input table** field, select the input table for the test, and click **OK**.

7 Verify or change the variable mappings. To run a full test, map all of the input variables to columns in the input table that you selected for the test. To run a partial test, you can map only the input variables that are needed for the test.

   SAS Intelligent Decisioning automatically maps the input variables in the code file to columns in the input table when the names and data types of the variables match those of the table columns.

   If any input variables are not mapped to input columns or to static values, the application displays a warning message. At run time, SAS Intelligent Decisioning assigns missing values to input variables that are not mapped.



HMEQ_TEST

Input variables must be mapped to table columns.

You can change the automatic variable mappings in the Variable Mappings window.

To change variable mappings:

a Click **Variables**. The Variable Mappings window appears.

b For each input variable, select the table column to which the variable should be mapped. If the input table contains more than 25 columns, click **More columns** to display additional column names. Alternatively, for Decimal, Integer, and Character variables, you can select **Use value** for the table column, and specify a literal value in the **Value** column. When you are entering literal values, remember these rules:

- Do not enclose character strings in quotation marks.
- To specify a missing value for character variables, select **Use value** and leave the **Value** column empty. When SAS Intelligent Decisioning generates code, it generates an empty string (`''`). For numeric values, enter a period (.).

Note: For content that will be published to a SAS Micro Analytic Service destination, the values of date and datetime input variables must be numeric. They cannot be formatted values such as `18jul2019`. Also, integer variables are converted to decimal variables when the content is published.

c Click **OK** to close the Variable Mappings window.

8 (Optional) Click **Advanced** to display the advanced options.

9 (Optional) Click 🗀 and select the library where you want to write the output of the test.

10 (Optional) Enter a name for the test results output table if you do not want to use the default name. The default name is `test-name_object-name_timestamp_output`.

11 (Optional) Select the version of the code file that you want to test.

12 (Optional) Select the variable that you want to serve as an input debug variable. You can specify an input-only variable or an input-output variable.

For more information, see “Debugging Code File Tests” on page 190.

13 (Optional) Select **Preserve unmapped columns in the output table** if you want columns that are not mapped to an output variable to be written to the output table.

14 Click **Run** to run the test. Alternatively, click **Save** to save the test definition without running it.

The status of the test is indicated by the icon in the **Status** column. For explanations of each icon, see “Status Icons for Tests” on page 194.

15 Click ▤ in the **Results** column to view the results of the test.

16 In the test results window, click **Test Results** in the navigation panel to display the URIs and other information for the test. Click **Output**, **Code**, or **Log** to display the output data set, the code that was generated by SAS Intelligent Decisioning, or the SAS log that was generated when the code was run.

TIP You can click on the values of character variables to display the entire value in a separate window. For data grid variables, you can choose to view the variable value in three different formats:

- Click the **Data Grid** tab to view the data grid value as a table.
- Click the **Formatted** tab to view the data grid as a formatted JSON character string.
- Click the **Plain** tab to view the data grid as an unformatted character string.

TIP On the **Log** page, you can click ⭳ to download the log file.

Debugging Code File Tests

When you create a test, you can specify a variable to use as a debugging variable in the **Input debug variable** field. You can specify an input-only variable or an input-output variable. For more information, see Step 12 of "Create and Run a New Test" on page 188.

When you specify an input debug variable, SAS Intelligent Decisioning automatically sets the maximum number of threads that can be allocated for the test to 1. Setting the thread count to 1 ensures that the variable's values are written to the log in the correct order and are not affected by different threads completing at different times.

To capture variable values for input-only or temporary variables after the code file logic has executed for a specific record, you can specify that the variable is an output variable, and then re-run the test. Before incorporating the code file into a production system, return the input and output settings for the variable to their previous settings. For more information, see "Input Variables and Output Variables" on page 170 and "Edit Variable Properties" on page 175.

Test a Scenario

Create and Run a Scenario Test

IMPORTANT If you are testing a DS2 or SQL code file that uses functions that are defined in a custom context file, verify that the context file is specified in the **Test custom context** field on the **Properties** before running the test.

1 On the **Scoring** tab, click the **Scenarios** tab.

2 Click **New Test**. The New Scenario Test window appears.

3 Enter a name for the test if you do not want to use the default name. The name cannot contain forward slashes (/) or curly braces ({}).

4 (Optional) Click for the **Test definition location** field, and select the folder where you want to save the test definition.

> **TIP** Selecting a test definition location is optional, but it is highly recommended. Storing test definitions in a folder simplifies the tasks of setting permissions and transferring the test files.

5 Click for the **Output table location** field, and select the folder where you want to save the test results.

6 (Optional) Select the version of the code file that you want to test.

7 (Optional) Enter a description for the test. Descriptions are limited to 1000 characters.

8 Enter the values that you want to use for each input variable. You do not have to enter values for every input variable. At run time, SAS Intelligent Decisioning uses missing values to input variables for which you do not specify a value.

Note: Values longer than 32767 characters for character variables that are either input-only or input-output will be truncated. You cannot enter input values for variables of type binary or of type varying-length binary.

To enter values for the columns in a data grid variable:

a Click in the **Value** field, and then click . The Edit Data Grid window appears.

b Click + to add the row, and then enter the values for each column.

Repeat this step for each row of values that you want to add to the data grid.

> **TIP** By default, data grid column names appear across the top of the data grid view, and row numbers appear down the left side. You can click to change the view of the data grid so that row numbers appear across the top and data grid column names appear down the left side.

c Click **OK** to save the data grid values and close the Edit Data Grid window.

9 (Optional) For each output variable, select the **Include** check box and enter the expected output value. The **Include** check box controls whether a variable's expected value is used to determine the status of a scenario test. If you select **Include** for a variable and the test does not return the expected value, the test status is set to Completed with warnings (⚠). If you do not select the check box, the application ignores the expected value of that variable when it determines the status of the test.

To enter expected values for the columns in a data grid variable, click in the **Expected Output** field, and follow the instructions in Step 8.

Note: A scenario test cannot verify issues with trailing spaces. For example, it cannot distinguish between a string that contains a single space `' '` and a string that contains three spaces `'   '`.

10 Click **Run** to run the test. Alternatively, click **Save** to save the test definition without running it.

The status of the test is indicated by the icon in the **Status** column. For explanations of each icon, see "Status Icons for Tests" on page 194.

11 Click ▤ in the **Results** column to view the results of the test.

12 On the Test Results page, click **Test Results** in the navigation panel to display the URIs and other information for the test. Click **Input** to display the values of input variables and dynamic attributes. Click **Output**, **Code**, or **Log** to display the output data set, the code that was generated by SAS Intelligent Decisioning, or the SAS log that was generated when the code was run.

On the **Output** page, click **Show All** to display the expected and actual values of all output variables. Click **Show Differences** to display only the variables whose expected values do not match the actual values that were returned by the test.

TIP You can click on the values of character variables to display the entire value in a separate window. For data grid variables, you can choose to view the variable value in three different formats:

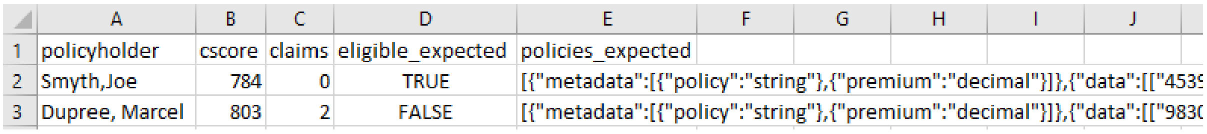
- Click the **Data Grid** tab to view the data grid value as a table.
- Click the **Formatted** tab to view the data grid as a formatted JSON character string.
- Click the **Plain** tab to view the data grid as an unformatted character string.

TIP On the **Output** page, click **Export** to export the output table as a comma-separated values (CSV) file. On the **Log** page, click ⤓ to download the log file.

Import Scenario Test Definitions

You can import scenario test values from a comma-delimited (CSV) file. Each line in the CSV file is imported as one scenario test. In the CSV file, add a column of values for each variable. In the header row, enter the names of the input variables and of the output variables with `_expected` appended to the name.

For example, suppose your scenario test has the input variables `policyholder`, `cscore`, and `claims`, and it has the output variables `eligible` and `policies`. An import file for this test might appear in a spreadsheet application as shown in the following figure.



	A	B	C	D	E	F	G	H	I	J
1	policyholder	cscore	claims	eligible_expected	policies_expected					
2	Smyth,Joe	784	0	TRUE	[{"metadata":[{"policy":"string"},{"premium":"decimal"}]},{"data":[["453					
3	Dupree, Marcel	803	2	FALSE	[{"metadata":[{"policy":"string"},{"premium":"decimal"}]},{"data":[["983					

The policies output variable in this example is a data grid variable. To enter values for data grid variables, use the JSON string format described in “Introduction to Data Grids” in *SAS Intelligent Decisioning: Using Data Grids*.

The format for date and datetime variables depends on your locale. Use the same format that is created by the date and datetime pickers when you click or to enter initial values for custom variables on the **Variables** tab.

To import scenario test definitions:

1 On the **Scoring** tab, click the **Scenarios** subtab, and then click **Import Scenarios**. The Import Scenarios window appears.

2 In the **Import from** field, click and select the CSV file that contains the scenario test values.

Note: The import file is limited to 10 MB.

3 Select or enter the encoding of the CSV file.

4 Enter a prefix for the scenario test definitions. SAS Intelligent Decisioning appends a number to this prefix for each test definition. The prefix can include double-byte characters and special characters, including single quotation marks.

5 (Optional) Click for the **Folder location** field, and select the folder where you want to save the test definitions.

6 Click for the **Output table location** field, and select the folder where you want to save the test results.

7 Click **Import**.

Compare Different Scenarios

You can display the scenario definitions or results of two or more tests side-by-side. On the **Scenarios** tab, select two or more tests, click , and select one of the following options:

Compare ⇨ Definitions
displays the input values and the expected output values that you entered for both tests. To edit the input values for a scenario test definition, click next to the test name under **Input Table**. To edit the expected output values for a test, click next to the test name under **Output Table**.

Compare ⇨ Results
displays the input values and the actual output values that were generated by the test. To display both the expected values and the actual values in the output table, select **Display expected values**. For each variable for which you selected **Include** in Step 9 on page 191, the application highlights the variable if the actual and expected values do not match.

Click **Export** to export the results comparison as a comma-separated values (CSV) file. The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Working with Test Output Data

After you run a test, you can work with the output table in other SAS applications to analyze the data, create and compare models, discover relationships hidden in the data, and generate reports based on the data.

Note: The actions available to you depend on the applications that are available at your site.

On the Test Results page, select the **Output** table in the navigation pane, click **Actions**, and select one of the following options:

Explore Lineage
opens SAS Lineage Viewer. SAS Lineage Viewer enables you to better understand the relationships between objects in your SAS Viya applications. These objects include data, transformation processes, reports, and visualizations. For more information, see *SAS Lineage: User's Guide*.

Explore and Visualize Data
opens the output table in SAS Visual Analytics. SAS Visual Analytics enables you to create, test, and compare models based on the patterns discovered during exploration of the data. You can export the model before or after performing model comparison for use with other SAS products or to put the model into production. SAS Visual Analytics supports a range of visualization, discovery, and reporting features. For more information, see *Welcome to SAS Visual Analytics*.

Prepare Data
opens the output table in SAS Data Studio. SAS Data Studio enables you to perform data transforms such as joining tables, appending data to a table, transposing columns, creating calculated columns, and so on. For more information, see *SAS Data Studio: User's Guide*.

Status Icons for Tests

Icon	Status
○	The test is not ready to run. The test definition is not complete, or it might contain errors.

Icon	Status
◎	The test has been defined and can be run. Some input variables have not been mapped or have not been assigned a value, so the test might execute only a subset of the code in the custom file.
◉	The test is defined correctly and is ready to run.
⟳	The test is running.
✓	The test completed successfully.
✓⚠	The test completed, but warnings were issued in the SAS log. The URI to the log file is shown on the Test Results page. On the Test Results page, click **Test Results** in the navigation panel to display the URI.
⊗	The test did not run successfully. Check the SAS log for information. The URI to the log file is shown on the Test Results page. On the Test Results page, click **Test Results** in the navigation panel to display the URI.

Manage Comments for a Code File Test

You can associate comments and attachments with a code file test.

To open the Comments properties pane, select the code file test on the **Scoring** tab, and click 💬 in the property pane.

To add a new comment, enter the comment in the text box and click **Post**.

To add an attachment to a comment, click 📎, select the file that you want to attach, and click **Post**. (The attachment icon appears after you enter text in text box.) You cannot attach executable files such as BAT and EXE files.

To reply to a comment, click **Reply**, enter your reply in the text box, and click **Post**.

To delete a comment, click 🗑 for that comment.

7

Using Custom Functions

About Custom Functions **198**

Using Custom Functions in a Rule Set or Code File **198**

Create a Custom Function **199**

Using the Custom Function Editor **200**

Rules for Writing Custom Function Files **201**

Managing the Variables in a Custom Function File **202**
The Properties of a Variable 202
Syncing Variables and Code 203
Add Variables from a Data Table or from a SAS Intelligent Decisioning Object . . . 204
Create Custom VariablesCreate Custom Variables 204
Duplicate a Variable 205
Importing and Exporting Variables 206
Delete Variables 207
Edit Variable Properties 207
Edit Metadata for Data Grid Variables 207

Validate a Custom Function File **207**

Testing a Custom Function **207**

Compare Custom Functions **208**

Managing Custom Function Categories **208**
Create a Custom Function Category 208
Edit a Custom Function Category 209
Hide A Custom Function Category 209

Managing Custom Functions **209**
Duplicate Custom Functions 209
Hide Custom Functions 210
Rename a Custom Function 210
Change a Function’s Category 210

Restore or Permanently Delete Hidden Functions or Hidden Categories **211**

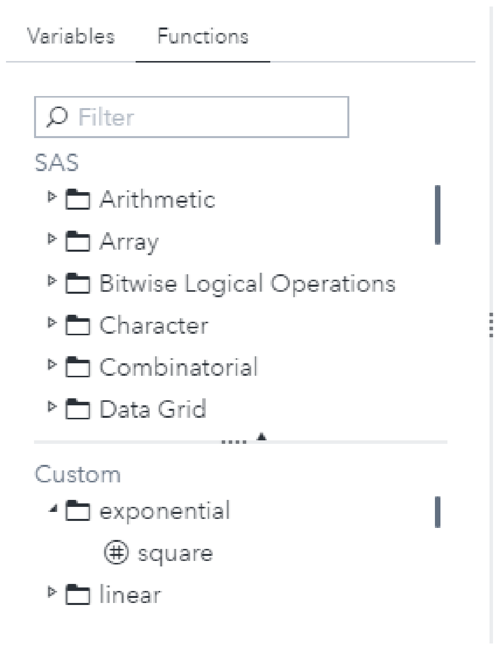
About Custom Functions

You can use custom functions to perform actions that are not available with the standard functions that SAS provides. Custom functions also enable you to encapsulate and reuse business logic. You can use custom functions in rule sets and in DS2 code files.

When SAS Intelligent Decisioning generates the code for a published object, the definitions for custom functions are duplicated each time the function is referenced.

Using Custom Functions in a Rule Set or Code File

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



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

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

Note: You can nest custom function calls. For example, if A and B are custom functions and DIVIDE is the DS2 function provided by SAS, you can nest functions as follows:

```
method B() returns double;
    return 5;
end;
method A() returns double;
    return DIVIDE(10,B());
end;
```

Create a Custom Function

1 In the Custom Functions category view, click **New Custom Function**. The New Custom Function window appears.

2 Enter a name for the function if you do not want to use the default name. The name cannot be the same as the name of an existing SAS function. Function names must be unique within an environment and are limited to 250 characters. Function names must be valid DS2 identifiers. For more information, see "DS2 Identifiers" in *SAS DS2 Programmer's Guide*.

3 (Optional) Enter a description for the new function. Descriptions are limited to 1000 characters.

> **TIP** You can edit the description later on the **Properties** tab.

4 Select the category to which the function belongs.

> **TIP** Custom functions must be associated with a function category. A category named **Category 1** is defined by default. You can change this category name, or create new a new category. For instructions about creating new categories, see "Create a Custom Function Category" on page 208.

5 Click **Save**. SAS Intelligent Decisioning opens the custom function editor.

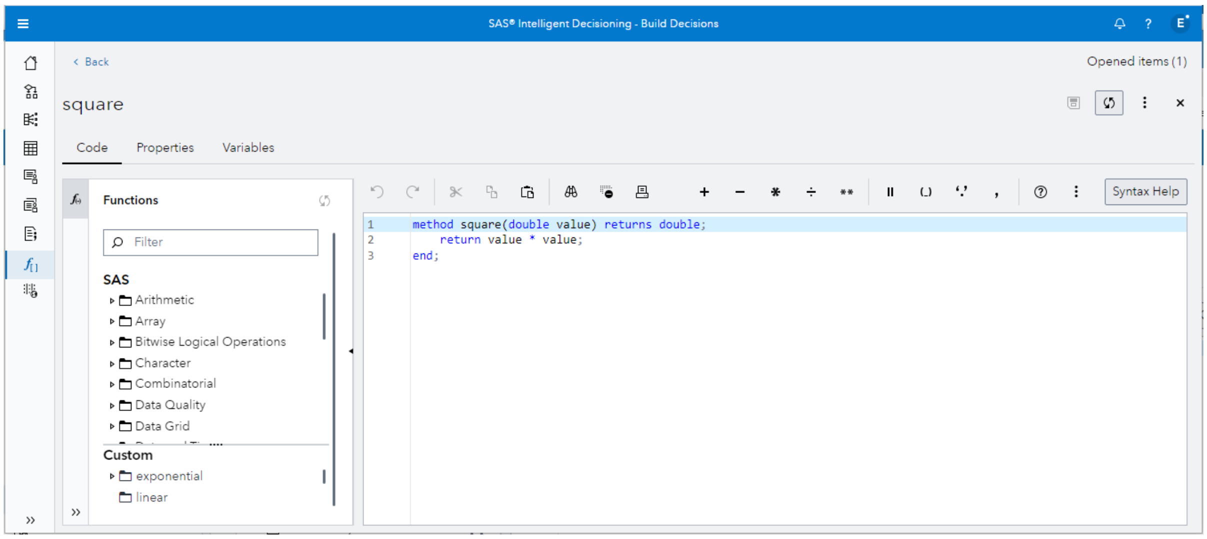
6 Enter the DS2 code for the function. For more information, see the following:

- "Using the Custom Function Editor" on page 200
- "Rules for Writing Custom Function Files" on page 201
- "METHOD Statement" in *SAS DS2 Language Reference*
- *SAS DS2 Programmer's Guide*

Using the Custom Function Editor

Use the custom function editor to enter the code for custom functions.

To open the custom function editor, select a function in the Custom Function category view.



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

You can enter code directly into the code field, or you can use the lists of operators and functions to add them to the code for the custom function.

- To add an operator, click the operator in the rows above the code pane. Click ⋮ to select additional operators that are not visible in the rows above the code pane.
- To add a function call, expand the appropriate function category on the **Functions** tab, and double-click the function name. The custom function editor adds a basic syntax template for the function at the cursor's position in the code field.

  The icon beside each function name indicates the return type of the function. Functions that return character data are displayed with the Ⓐ icon, and functions that return numeric data are displayed with the ⊕ icon.

You can click **Validate** at any time to check the syntax of the function that you are building. If the code does not compile correctly, SAS Intelligent Decisioning displays detailed information in the **Validation Results** window.

When you are finished editing the function, click **Save**.

TIP By default, the autocomplete feature is turned on for SAS functions and custom functions. You can turn off this feature or change what is displayed in the autocomplete list by using the SAS Intelligent Decisioning settings. You can specify that SAS Intelligent Decisioning also displays variable names. For information, see "SAS Intelligent Decisioning Settings" on page 8.

In the autocmplete list, SAS Intelligent Decisioning displays all function names and variable names that match the text that you enter. Double-click an item to add it to the code field. Single-click an item to display information about the item, such as whether it is a function or a variable. For functions, the information includes the returned data type and category. For variables, the information includes the data type and direction (input, output, or both).

The autocomplete list does not include functions that are defined in a custom context file.

Rules for Writing Custom Function Files

When you are developing your DS2 custom function, follow these rules:

- Do not change the name of the method. The method name must match the function name.
- The method signature can include variables that are both input and output variables. Specify these variables with the `in_out` keyword. Custom functions support data grid packages and the DS2 data types that are listed in "Data Types for SAS Data Sets" in *SAS DS2 Language Reference*.

  In the method signature, list input-only parameters first. List input-output and output-only parameters next, using the in_out modifier. Do not specify a length for input-output and output-only parameters. The length for these parameters is derived from the variables that are passed into the method.

  Note: Packages are input-output data types.
- The variables that you use as function parameters must be mapped to input variables in your decision.

Managing the Variables in a Custom Function File

The Properties of a Variable

Property	Description
Name	Variable names must start with a letter or an underscore (_), and they can contain only alphanumeric characters and the underscore. Variable names can be up to 32 characters long. They must be unique within a custom function. **Note:** SAS Intelligent Decisioning does not support double-byte character set (DBCS) characters in variable names. **Note:** Do not use any of these operators or keywords as variable names: AND, OR, IN, NOT, LIKE, TRUE, or FALSE. Do not use _N_ or any DS2 reserved word as a variable name. See "Reserved Words in the DS2 Language" in *SAS DS2 Programmer's Guide* for information about reserved words in the DS2 language.
Data type	Custom functions support data grid packages and the DS2 data types that are listed in "Data Types for SAS Data Sets" in *SAS DS2 Language Reference*. For Boolean variables, you can select `True` or `False` for the initial value. However, SAS Intelligent Decisioning represents Boolean values by using the numbers 1 and 0 in the code that it generates. When SAS Intelligent Decisioning is evaluating Boolean expressions, any non-zero number is considered True. When you are entering expressions, specify `1` for True and `0` for False. **Note:** For content that will be published to a SAS Micro Analytic Service destination, the values of date and datetime input variables must be numeric. They cannot be formatted values such as `18jul2019`. Also, integer variables are converted to decimal variables when the content is published.
Input and **Output**	A variable in a custom function signature can be an input variable, an output variable, or both.
Length	The maximum length for character variables (outside of a data grid) and data grid variables is 10485760 characters. The maximum length for character variables within a data grid is 32767 characters.

Property	Description
Initial value	You can specify an initial value for all data types except data grids. Initial values are not used when you run a custom function. For custom functions, initial values are only for documentation purposes. Note: For content that will be published to a SAS Micro Analytic Service destination, the values of date and datetime input variables must be numeric. They cannot be formatted values such as `18jul2019`. Also, integer variables are converted to decimal variables when the content is published.
Description	Descriptions are limited to 256 characters.

Syncing Variables and Code

To propagate changes that you enter on the **Variables** tab to the code that is shown on the **Code** tab, click **Sync to Code** on the **Variables** tab. Changes to variables are propagated only to the signature of the custom function. The rest of the custom function code is unchanged. Both output-only and input-and-output variables are added with the `in_out` keyword.

To propagate changes that you enter in the method signature on the **Code** tab to the **Variables** tab, click ⋮ and select **Sync Variables** on the **Code** tab. Variables that you add to the signature with the `in_out` keyword are added to the **Variables** tab as both input and output variables. Variables that are not specified with the `in_out` keyword are added to the **Variables** tab as input-only variables. You can specify that any variable is both an input and output variable by selecting both the **Input** and **Output** check box on the **Variables** tab.

If you make conflicting changes to both the **Code** tab and the **Variables** tab and then save the file, SAS Intelligent Decisioning prompts you to select which tab contains the changes that you want to retain. If you make conflicting changes on the two tabs and then click a sync button, only the changes on the tab on which you click the sync button are retained. Those changes are then propagated to the other tab.

Note: You cannot manage local variables on the **Variables** tab. Local variables are variables for which neither the **Input** nor the **Output** checkboxes are selected on the **Variables** tab. If local variables are listed on the **Variables** tab, these variables are removed from the tab when you save the code file or click a sync button.

Add Variables from a Data Table or from a SAS Intelligent Decisioning Object

1 On the **Variables** tab, click **Add variable** and select the object type from which you want to add variables. The Add Variables window appears.

2 Click 🗀. If you are importing variables from a data table, the Choose Data window appears. If you are importing from another SAS Intelligent Decisioning object, the Choose an Item window appears.

> **TIP** In the Choose Data window, you can search for data tables with a specific name by entering `Name: "`*`string`*`"` in the Search field. For example, to search for all tables whose names begin with `HMEQ`, enter `Name: "HMEQ"`.

3 Select the table or object from which you want to import variables, and click **OK** to return to the Add Variables window.

4 Select the variables that you want to import and click +›. To import all of the variables in the table, click +».

5 Click **Add** to add the selected variables, or click **Add and replace** to replace existing variables that have the same name.

6 On the **Variables** tab, select or clear the **Input** and **Output** check boxes as necessary. See "Input Variables, Output Variables, and Temporary Variables" on page 224 for more information.

> **TIP** To filter the variable list, right-click on the **Variable** column heading, enter a filter string in the **Filter** box, and press Enter. To clear the filter, delete the string from the **Filter** box, and press Enter.

7 Click **Sync to Code** to add the variables to the code that is on the **Code** tab, and then save the file. For more information, see "Syncing Variables and Code" on page 203.

Create Custom VariablesCreate Custom Variables

Note: For information about data grid variables, see "Defining Data Grid Variables" in *SAS Intelligent Decisioning: Using Data Grids*.

To create custom variables on the **Variables** tab:

1. Click **Add variable** and select **Custom variable**. The Add Variables window appears.

2. Complete these steps for each variable that you want to add:

   a. Enter the name of the new variable, and select the data type of the variable. See "The Properties of a Variable" on page 202 for additional information.

      > **TIP** To re-add a variable that was used in a locked version, you must specify the same data type that was used in the previous version.

   b. (Optional) Click **Optional** to display the **Description**, **Initial value**, and **Length** fields.

   c. (Optional) Enter a length, initial value, and description for the new variable. Whether you can specify an intial value or length for the variable depends on the variable's data type. See "The Properties of a Variable" on page 202 for additional information.

   d. Click **Add**. SAS Intelligent Decisioning adds the new variable to the table of variables at the bottom of the window. By default, variables are added to the table as both input and output variables.

   e. Verify that the **Input** and **Output** check boxes are selected correctly for each variable.

      - Clear the **Input** check box for any variable that you do not want to be mapped to a column in an input table or for which you do not want to specify a value.
      - Clear the **Output** check box for any variable that you want to exclude from the output data.
      - Clear both the **Input** and **Output** check boxes to create a temporary variable.

3. (Optional) Modify the variable properties in the table of variables.

4. Click **OK** to add the variables and close the Add Variables window.

5. Click **Sync to Code** to add the variables to the code that is on the **Code** tab, and then save the file. For more information, see "Syncing Variables and Code" on page 203.

Duplicate a Variable

1. On the **Variables** tab, select the variable that you want to duplicate, click ⋮, and select **Duplicate**. The Duplicate Variable window appears.

2. Enter a new name for the duplicate variable.

3. (Optional) Enter a description for the variable.

4. Click **Duplicate**.

5 Click **Sync to Code** to add the variables to the code that is on the **Code** tab, and then save the file. For more information, see "Syncing Variables and Code" on page 203.

Importing and Exporting Variables

Import Variables

You can import variables from either comma-delimited (CSV) files or from JavaScript Object Notation (JSON) files. These files must conform to formats described in Appendix 2, "Import File Formats," on page 325. These formats are the same formats that are created when you export variables.

1 Open the custom function into which you want to import variables.

2 On the **Variables** tab, click **Import**, and select either **Comma-delimited (*.csv)** or **JSON (*.json)**. The Import File window appears.

3 Click **Browse** and select the file from which you want to import variables.

4 Select **Add variables** to append the variables to the current list of variables, or select **Replace variables** to replace the current list of variables with the imported variables.

> **TIP** To add new columns to an existing data grid variable, select **Replace variables**.

5 Click **Import**, and then click 🖫 to import the variables and save the custom function.

Export Variables

1 Open the custom function from which you want to export variables.

2 On the **Variables** tab, select the variables that you want to export.

3 Click **Export**, and select the file type to which you want to export the variables. The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Delete Variables

1. On the **Variables** tab, select the check box for the variables that you want to delete, click ⋮, and select **Delete**.
2. Click **Sync to Code** to delete the variables from the code that is on the **Code** tab.

Edit Variable Properties

On the **Variables** tab, click on the name of the variable that you want to edit. The Edit Variable window appears. Edit the properties as needed, and then click **OK**.

Note: When you rename a variable, click **Sync to Code** to change references to that variable within the same object change to use the new name.

See "The Properties of a Variable" on page 202 for additional information.

Edit Metadata for Data Grid Variables

For information, see "Editing Data Grid Variable Metadata" in *SAS Intelligent Decisioning: Using Data Grids*.

Validate a Custom Function File

To validate a custom function file, open the file, click ⋮, and select **Validate**. For custom function files that use a custom context file, specify the custom context file in the **Test custom context** field on the **Properties** tab.

Testing a Custom Function

Before you use a custom function in a rule set, test the function code by including it in a DS2 custom code file and running a scenario test on the code file. For example, you can test the SQUARE custom function with the following custom code file:

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

If your custom function uses a custom context file, specify the custom context file in the **Test custom context** field on the **Properties** tab.

For more information, see "Testing Code Files" on page 187 and "Testing Rule Sets" on page 63.

Compare Custom Functions

1 In the Custom Functions category view, select the two custom functions that you want to compare, click ⋮, and select **Compare code**. SAS Intelligent Decisioning opens the two objects side-by-side in the Compare Code window.

2 (Optional) Click **Export** to export the results of the comparison to a PDF file.

   The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Managing Custom Function Categories

Create a Custom Function Category

1 Click $f_{[\,]}$ to navigate to the Custom Functions category view.

2 Click ⋮ and select **Manage categories**. The Manage Categories window appears.

3 Click **New Category**. The Create New Category window appears.

4 Enter a name for the category if you do not want to use the default name. Category names are limited to 250 characters and cannot contain forward slashes (/).

5 (Optional) Enter a description for the new category. Descriptions are limited to 1000 characters.

6 Click **Save**, and then click **Close**.

Edit a Custom Function Category

1 In the Custom Functions category view, click ⋮ and select **Manage categories**. The Manage Categories window appears.

2 Select the category that you want to edit, click ⋮, and select **Edit**. The Edit Custom Function Category window appears.

3 Modify the category name and description as needed, and click **Save**.

Hide A Custom Function Category

When you hide a custom function category, the category and all of its custom functions are removed from the category list. These items are hidden from view. They are not moved to the recycle bin, and they are not deleted. You cannot create a new category or new functions that have the same names as the hidden items unless you delete the hidden items. You can unhide (restore) or permanently delete the hidden items by using the **Manage hidden functions** menu option.

To hide a custom function category:

1 In the Custom Functions category view, click ⋮ and select **Manage categories**. The Manage Categories window appears.

2 Select the category that you want to hide, click ⋮, and select **Hide**. SAS Intelligent Decisioning prompts you to confirm whether you want to hide the category and all of its functions.

Managing Custom Functions

Duplicate Custom Functions

Note: You cannot duplicate a custom function if it is open.

To duplicate a single custom function:

1 In the Custom Functions category view, select the custom function that you want to duplicate.

2 Click ⋮ and select **Duplicate**. The Duplicate Custom Function window appears.

3 Enter a new name for the duplicate custom function.

4 (Optional) Enter a description for the custom function.

5 Select the category for the function, and click **Duplicate**.

To duplicate multiple custom functions:

1 In the Custom Functions view, select the custom functions that you want to duplicate.

2 Click ⋮ and select **Duplicate**. SAS Intelligent Decisioning duplicates the custom functions and appends `_Copy` to the names of the duplicate copies.

Hide Custom Functions

When you hide a custom function, it is removed from the category list. It is hidden from view, but it is not moved to the recycle bin or deleted. You cannot create new functions that have the same names as the hidden functions unless you delete the hidden functions. You can unhide (restore) or permanently delete the hidden functions by using the **Manage hidden functions** menu option.

To hide custom functions, select the functions in the Custom Functions category view, click ⋮ , and select **Hide**.

Rename a Custom Function

Note: You cannot rename a custom function if it is open.

1 In the Custom Functions category view, select the custom function that you want to rename.

2 Click ⋮ and select **Rename**. The Rename window appears.

3 Enter a new name for the custom function, and click **Rename**.

Change a Function’s Category

1 In the Custom Functions category view, select the custom function, click ⋮ , and select **Change category**. The Change Category window appears.

2 Select the new category, and click **OK**.

Restore or Permanently Delete Hidden Functions or Hidden Categories

1. In the Custom functions category view, click ⋮, and select **Manage hidden functions**. The Manage Hidden Functions window appears.
2. Select the function or category that you want to restore or delete.
3. Click 🗑 to permanently delete the selected item, or click ⬓ to unhide (restore) it.

8

Working with Decisions

About Decisions **215**

Using SAS Workflow with SAS Intelligent Decisioning **216**
About the SID Asset Approval Workflow 216
User Roles in the Workflow 216
Decision Status Values in the Workflow 218
View the Status Change History 219
Display and Export the Included Object Report for Decisions That Are in a Workflow 219

Create a Decision **220**

Views for Editing a Decision **221**
The Decision Flow Tab versus the Decision Tab 221
Controlling the Tab Display 221

Managing the Variables in a Decision **222**
About Variables 222
Add Variables from a Data Table 224
Add Variables from a Rule Set, Decision, or Code File 225
Add Global Variables to a Decision 225
Create Custom Variables on the Variables Tab 226
Duplicate a Variable 227
Importing and Exporting Variables 227
Delete Variables 228
Edit Variable Properties 229
Edit Metadata for Data Grid Variables 229
Determine Which Objects Use a Particular Variable 229

Adding Objects to a Decision **230**
Add an Existing Object 230
Create and Add a New Object 232
Adding Record Contacts Nodes 233
Adding Branches to a Decision 235
Adding Cross-Branch Links 244
Add a Micro Analytic Module 246

Mapping Variables within a Decision **247**
About Decision Variables and Mapping 247
Add Missing Variables 248

Scoring Rows in a Data Grid 249
Mapping Data Grid Variables 249
Map Object Variables to Decision Variables 250

Searching for Objects in a Decision Diagram **251**

Editing the Objects in a Decision **251**
Open an Object from within a Decision 251
Edit the Properties of a Decision Node 252
Reorder Objects 252
Remove an Object from a Decision 253
Replace an Object 253

Update Decisions to Use New Object Versions **253**

Validate a Decision in the Decision Editor **254**

Change the Workflow Status of a Decision **254**

Copy a Decision URL **255**

Compare Decision Content **255**

Compare Decision Code **258**

Generate PDF Documentation for a Decision **258**

Managing Decisions **260**
Duplicating Decisions 260
Delete Decisions 262
Rename a Decision 262
Move Decisions to a Different Folder 262

Managing Versions of Decisions **262**
Set the Displayed Version 262
Create a New Version 263
Creating a New Version by Using Tags 264
Create a New Version Automatically When You Publish a Decision 269
Copy the Content of a Version 269
Delete a Version 270
Upgrade Decisions to Use a New Version of a Nested Decision 270

Determine Which Objects Use a Decision **271**

Checking Out and Committing Decision Versions **272**
About Checking Out and Committing Versions 272
Checking Out and Committing Included Objects Together with A Decision 273
Checking Out and Committing Objects from within A Decision 274
Concurrently Checking Out and Committing Decision Versions 274
Check Out and Commit a Decision Version 275
Determine Whether a Version Is Checked Out 276
Determine Who Has a Version Checked Out 276
Opening the Original Object 277
Undoing a Check Out 277

Managing Version Tags for Decisions **277**
Add a Version Tag 277
Remove a Version Tag 278
Modify a Version Tag 278

Publishing a Decision **278**
Introduction to Publishing 278

Where Can I Publish Decisions? . . . 279
Publishing Decisions That Include Analytic Store Models . . . 281
Publish Decisions . . . 281

Testing Decisions . . . **284**
Ways to Test a Decision . . . 284
Test a Decision . . . 285
Test a Scenario . . . 289
Run a Rule-Fired Analysis . . . 298
Run a Path Tracking Analysis . . . 301
Content That Is Used by Tests and Scenarios for Decisions . . . 303
Validate a Published Decision . . . 303
Working with Test and Validation Test Output Data . . . 305
Status Icons for Tests . . . 306

Manage Comments for Decision Nodes and Tests . . . **306**

Executing Published Content . . . **307**
Executing Content Published to SAS Micro Analytic Service Destinations . . . 307
Executing Content That Has Been Published to SAS Cloud Analytic Services Destinations . . . 308
Content Executed by Published Decisions . . . 309

View All Published Decisions . . . **310**

About Decisions

A decision enables you to combine rule sets, analytical models, treatment groups, code files, record contacts nodes, and conditional logic into a single process. You can also add a decision to another decision.

TIP Your site might have additional custom node types that you can add to a decision. You can create custom node types by using the Decision Management REST API. For more information, see https://developer.sas.com/apis/rest/DecisionManagement/#decisions-decisionnodetypes.

Using SAS Workflow with SAS Intelligent Decisioning

About the SID Asset Approval Workflow

SAS Intelligent Decisioning provides a predefined approval workflow for decisions. This workflow is named SID Asset Approval. Your administrator can enable this workflow for decisions in SAS Intelligent Decisioning by turning on the sas.decisions.workflow.enabled configuration property. For more information, see "Enable the Asset Approval Workflow" in *SAS Intelligent Decisioning: Administrator's Guide*.

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. A workflow is associated with a specific version of a decision. Each decision version that is actively being worked on has its own workflow instance.

Decisions that already existed before the workflow was enabled are not automatically put into the workflow. To put these decisions into the workflow, you must create a new version of the decision.

As the users who are responsible for developing, testing, reviewing, approving, and deploying a particular version of a decision each finish their tasks, they set the status of the workflow to the appropriate value. To set workflow status values, you must be included in the appropriate SAS Intelligent Decisioning workflow permissions group. See "User Roles in the Workflow" on page 216 for more information.

IMPORTANT Do not modify the SID Asset Approval workflow. Updates to your system will overwrite any customized changes to the workflow.

User Roles in the Workflow

Users that participate in the SID Asset Approval workflow can have up to four different roles:

Author
: creates, develops, and tests the decision in SAS Intelligent Decisioning. When the workflow is enabled, and the author creates a new decision or a new version of an existing decision, the decision status is automatically set to Developing. When the author is ready for the new decision version to be reviewed, the author sets the status to Review-ready. Before the author sets the status to Review-

ready, it is recommended that the author verify that the objects used in the decision are locked and cannot be edited.

Authors must be included in the SIDWFAuthor custom user group.

Reviewer
: reviews the new version and decides whether to approve it. For example, the reviewer might compare the new version against the previously approved version and review the tests that were run by the author. The reviewer approves the decision by setting the status to Approved. The reviewer rejects the decision by setting the status back to Developing.

Reviewers must be included in the SIDWFReviewer custom user group.

> **TIP** If the sas.decisions.workflow.authorMayApprove configuration option is turned on, then users who are a member of both the SIDWFAuthor and the SIDWFReviewer custom user groups can approve work that they authored.

Deployer
: determines whether the new version can be published and promoted to other environments for additional testing and production. If the version can be deployed, this user sets the status to Deployment-ready. When the version is ready to be deployed, the deployer can publish and promote the version, and set the status to Deployed.

Deployers must be included in the SIDWFDeployer custom user group.

Auditor
: view workflow status change histories and produce audit reports.

Auditors must be in at least one of the four user groups described in "Granting Access to the History of Workflow Status Changes" in *SAS Intelligent Decisioning: Administrator's Guide*.

Administrator
: can perform any task that the author, reviewer, auditor, or deployer can perform. SAS Intelligent Decisioning workflow administrators must be included in the SIDWFAdmin custom user group.

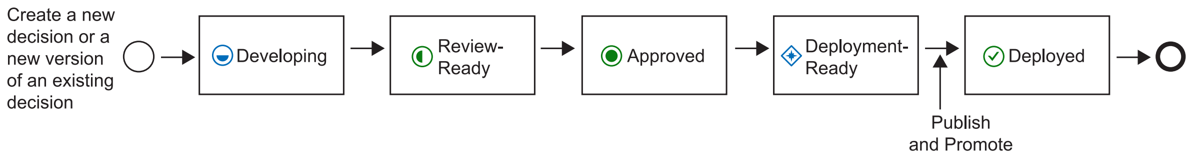
> **TIP** Members of the SAS Administrators group also have permission to make all status changes.

Note: Users in all roles can publish a decision, but it is strongly recommended that only users who have the role of Deployer or Administrator be permitted to publish decisions to testing and production environments. In cases where decisions are published to a Git repository, it is recommended that only users with the roles of Deployer or Administrator have access to the Git repository.

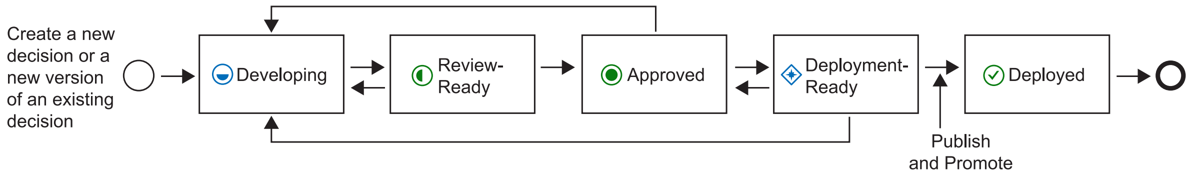
For information about defining custom user groups for the workflow, see "Define Asset Approval Workflow User Groups" in *SAS Intelligent Decisioning: Administrator's Guide*.

Decision Status Values in the Workflow

The basic movement of a decision version through the workflow status values is shown in the following figure.



Until the version’s status is set to Deployed, the version can be moved back to a previous status value. The following figure shows all of the possible transitions between status values.



Note: These diagrams do not show the actual workflow. They show only the possible changes in a version’s status value as it moves through the workflow.

When the SID Asset Approval workflow is enabled, the **Status** column in the Decisions category view shows the current status of the latest version of each decision that was created when the workflow was enabled. The same status value is displayed on the **Properties** tab and in the **Properties** panel in the Decisions category view. The **Status** column is blank for decisions that were created when the workflow was not enabled.

The **Status** column on the **Versions** tab displays the status value for each version of a decision. On the **Versions** tab, you can click to view or change the version’s status and to edit the comments that are associated with the version’s status. For more information, see “Change the Workflow Status of a Decision” on page 254.

If a version's status is not set to Developing, appears beside the version number in the Version column. You can edit or delete a decision version only if its status is set to Developing. You can delete a version only if its status is blank, or if its status is set to Developing and the version’s status has never been set to Deployed. After a version’s status has been set to Deployed, the version cannot be deleted, and its status cannot be changed. To make additional changes to the decision, you must create a new version.

The icon also appears next to the version number for any version that is not the latest version. You can edit the content of only the latest version of a decision, and you can edit its content only if its status is Developing. You can change the workflow status of any version if you have the appropriate permissions and the decision’s status has never been set to Deployed.

Note: A workflow can be canceled by an administrator in SAS Workflow Manager if its status has not been set to Deployed.

View the Status Change History

You must have permission to view the status history. For more information, see "Granting Access to the History of Workflow Status Changes" in *SAS Intelligent Decisioning: Administrator's Guide*.

To view the history of workflow status changes for a decision:

1. Open the decision, click ⋮, and select **View status change history**. The Status Change History Range window appears.
2. Select the dates or versions for which you want to view the history:
   - Select **Dates**, select a start date, and, if needed, select an end date.
   - Select **Versions**, and, if needed, select a different start version and end version.
3. Click **OK**. The Status Change History window appears.

To download a status history report in the Status Change History window, click **Download**. SAS Intelligent Decisioning downloads the report as a comma-separated values (CSV) file. The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Display and Export the Included Object Report for Decisions That Are in a Workflow

To display the versions of the objects that are included in a decision version that is in a workflow, click ☰ in the **Status** column on the **Versions** tab for that decision version. The Included Objects window appears. For assignment rule sets, filtering rule sets, nested decisions, and code files, this window lists each object and the version that is used in the decision. For treatment groups and for lookup tables that are referenced by other objects, the window lists the active version.

Note: This report does not list common rule sets.

To export the Included Objects report:

1. Click **Export** in the Included Objects window. The Create Document window appears.
2. Enter a file name for the report.

3 Select either **CSV** or **PDF**, and click **Create**. The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Note: Approving a decision does not automatically lock the objects that are included in the decision. If you do not lock the included objects before a decision is approved, the included objects can be modified after the decision is approved.

Create a Decision

1 Click to navigate to the Decisions category view.

2 Click **New Decision**. The New Decision window appears.

3 Enter a name for the decision if you do not want to use the default name. Decision names are limited to 100 characters and must be unique within a folder.

Note: Some publishing destinations restrict the characters that can be used in the published name of a decision. For more information, see Table 8.4 on page 282.

4 (Optional) Enter a description for the new decision. Descriptions are limited to 1000 characters.

TIP You can edit the description later on the **Properties** tab.

5 Click , and select the folder where you want to save the decision.

6 Click **Save**. SAS Intelligent Decisioning opens the new decision and displays the **Decision Flow** tab.

TIP Objects that are saved in a folder for which the check-out and commit feature is enabled, such as the `Decision Repository` folder, must be checked out before they can be edited.

7 (Optional) If the decision is in a folder for which the check-out and commit feature is enabled, click the **Versions** tab and check out the latest version of the decision. For more information, see "Check Out and Commit a Decision Version" on page 275.

8 Add objects, and if needed, add variables to the decision. For more information, see "Adding Objects to a Decision" on page 230 and "Managing the Variables in a Decision" on page 222.

9 Click to save the decision.

10 (Optional) If the SID Asset Approval workflow is enabled at your site, click **Move to status**, and change the decision's status. For more information, see "Using SAS Workflow with SAS Intelligent Decisioning" on page 216 and "Change the Workflow Status of a Decision" on page 254.

> **TIP** If the workflow is not enabled at your site, **Move to status** does not appear in the SAS Intelligent Decisioning interface.

Views for Editing a Decision

The Decision Flow Tab versus the Decision Tab

There are two tabs on which you can view and edit decisions.

- On the **Decision Flow** tab, you can edit a decision by using a graphical editor.

  To add objects to a decision, right-click on a node to display a menu or click or . To choose objects that have already been defined by navigating to their location, click . To add branches, record contacts nodes, or create new objects by selecting the object type, click .

- On the **Decision** tab, you can edit the decision by using a tabular view similar to the rule set editor.

  To add objects to a decision, click **Add** if nothing is selected, or select **Edit** ⇨ **Add** if a decision node is already selected. You can add an existing object or click the **New** button in the file selection window for custom code file types to create new objects.

Save your work before you switch tabs. Click to refresh the view in a tab.

Controlling the Tab Display

On the **Decision Flow** tab:

- Click to open the properties panel for the selected object. Click ›› to hide the properties panel.
- Click to open the diagram overview. The diagram overview is a scaled-down version of the entire diagram. You can pan across the diagram and position the view over different sections of the diagram. The section that is currently visible on the screen is outlined. The overview is useful when a decision diagram is too large to display all of the nodes on one screen.

- Click to switch between displaying cross-branch links as either dashed lines or as nodes. See "Overview of Cross-Branch Links" on page 244 for more information.
- Click to switch between displaying decision node titles on one line or on two lines.

On the **Decision** tab:

- Click ∨ or > to collapse or expand a single node in the decision.
- Click or to collapse or expand all of the nodes in the decision.

Managing the Variables in a Decision

About Variables

The Properties of a Variable

Note: For information about data grid variables, see "Using Data Grids in SAS Intelligent Decisioning" in *SAS Intelligent Decisioning: Using Data Grids*.

Property	Description
Name	Variable names must start with a letter or an underscore (_), and they can contain only alphanumeric characters and the underscore. Variable names can be up to 32 characters long. They must be unique within a decision. **Note:** SAS Intelligent Decisioning does not support double-byte character set (DBCS) characters in variable names. **Note:** Do not use any of these operators or keywords as variable names: AND, OR, IN, NOT, LIKE, TRUE, or FALSE. Do not use _N_ or any DS2 reserved word as a variable name. See "Reserved Words in the DS2 Language" in *SAS DS2 Programmer's Guide* for information about reserved words in the DS2 language.
Data type	SAS Intelligent Decisioning supports the following data types: Boolean, character, data grid, date, datetime, decimal, and integer. Binary and varying-length binary variables are supported only in decisions. Binary variables are supported only as input variables or temporary variables in order to support models that require binary data. For Boolean variables, you can select `True` or `False` for the initial value. However, SAS Intelligent Decisioning represents Boolean

Property	Description
	values by using the numbers 1 and 0 in the code that it generates. When SAS Intelligent Decisioning is evaluating Boolean expressions, any non-zero number is considered True. When you are entering expressions, specify `1` for True and `0` for False. **Note:** For content that will be published to a SAS Micro Analytic Service destination, the values of date and datetime input variables must be numeric. They cannot be formatted values such as `18jul2019`. Also, integer variables are converted to decimal variables when the content is published.
Input and **Output**	A variable can be an input variable, an output variable, both, or neither (a temporary variable). See “Input Variables, Output Variables, and Temporary Variables” on page 224 for more information.
Length	The length for Boolean and numeric variable types is set automatically. For output-only data grid variables, the length is set to the value that you specify. For output-only character variables in decisions that are published to container destinations or to SAS Micro Analytic Service destinations, the length that you specify is ignored unless the sas.decisions.variable.length.honorOutputLengthInMAS configuration option is turned on. For more information,see “sas.decisions.variable.length” in *SAS Intelligent Decisioning: Administrator’s Guide*. For character variables and data grid variables that are input-only or input-output variables, the variableLengthOverridden configuration property determines how the length is set. By default, this property is set to Off, and the length is set to the length in the input data. When the variableLengthOverridden property is set to On, the length of input-only and input-output character variables and data grid variables is set to the value that you specify. For more information, see “sas.decisions.variableLengthOverridden” in *SAS Intelligent Decisioning: Administrator’s Guide*. The maximum length for character variables (outside of a data grid), data grid variables, binary variables, and varying-length binary variables is 10485760 characters. The maximum length for character variables within a data grid is 32767 characters.
Initial value	You can specify an initial value for all data types except data grid, binary, and varying-length binary types. Initial values are used only at run time and only for output-only variables. **Note:** For content that will be published to a SAS Micro Analytic Service destination, the values of date and datetime input variables must be numeric. They cannot be formatted values such as `18jul2019`. Also, integer variables are converted to decimal variables when the content is published.
Description	Descriptions are limited to 256 characters.

Input Variables, Output Variables, and Temporary Variables

For each variable, you must specify whether the variable is an input variable, an output variable, both an input variable and an output variable, or a temporary variable.

- Input variables are variables that are present in the input table for a decision. When a decision is deployed in a production system, all input variables must be mapped to table columns in input data. When you test a decision in SAS Intelligent Decisioning, for each input variable, you can either map it to a table column or specify a constant as its input value. If you choose not to map a variable to either a table column or a static value, SAS Intelligent Decisioning displays a warning message. When you create or edit a variable, clear the **Input** check box for any variable that you do not want to be mapped to a column in an input table or for which you do not want to specify a value.
- Output variables are variables that are written to the output table that is created when a decision is run. When you create or edit a variable, clear the **Output** check box for any variable that you want to exclude from the output data.
- Temporary variables are variables that are not present in the input data, and they are not written to the output table. To create a temporary variable for use only while a decision is executing, clear both the **Input** and **Output** check boxes.

When you create a new variable, it is created as an input-output variable by default.

Note: Binary variables are supported in decisions only as input variables or temporary variables in order to support models that require binary data.

Add Variables from a Data Table

1. On the **Variables** tab, click the **Decision Variables** subtab.
2. Click **Add variable** and select **Data table**. The Choose Data window appears, and the list of SAS Cloud Analytic Services (CAS) tables that are loaded into memory is displayed on the **Available** tab.

   If the table that you need does not appear in the list of available tables, try the following solutions:

   - If the table appears on the **Data Sources** tab, right-click on the table, and select **Load** to load the table into memory. If the table does not appear on the **Available** tab, click ↻.
   - If the table does not appear on the **Data Sources** tab, import the data. The process of importing the data loads it into memory. For information about importing data from different sources, see “Making Data Available to CAS” in *SAS Data Explorer: User’s Guide*.

3 Select the table from which you want to import variables, and click **OK**. The Add Variables window appears.

4 Select the variables that you want to import and click **+›**. To import all of the variables in the table, click **+»**.

5 Click **Add** to add the select variables, or click **Add and replace** to replace existing variables that have the same name.

6 On the **Variables** tab, select or clear the **Input** and **Output** check boxes as necessary. See "Input Variables, Output Variables, and Temporary Variables" on page 224 for more information.

> **TIP** To filter the variable list, right-click on the **Variable** column heading, enter a filter string in the **Filter** box, and press Enter. To clear the filter, delete the string from the **Filter** box, and press Enter.

Add Variables from a Rule Set, Decision, or Code File

1 On the **Variables** tab, click the **Decision Variables** subtab.

2 Click **Add variable**, and select **Rule set**, **Decision**, or **Code file**. The Choose an Item window appears.

3 Select the object from which you want to import variables, and click **OK**. The Add Variables window appears.

4 Select the variables that you want to import and click **+›**. To import all of the variables in the table, click **+»**.

5 Click **Add** to add the selected variables, or select **Add and replace** to replace existing variables that have the same name.

6 On the **Variables** tab, select or clear the **Input** and **Output** check boxes as necessary. See "Input Variables, Output Variables, and Temporary Variables" on page 224 for more information.

Add Global Variables to a Decision

In order to use a global variable in a decision, the global variable must be activated. Instructions for defining and activating global variables are in "Create a New Global Variable" on page 142 and "Activate a Global Variable" on page 149.

Note: You can add a global variable that has not been activated to a decision and publish the decision, but the variable's value is set to missing until the global variable is activated

1 On the **Variables** tab, click the **Global Variables** subtab.

2 Click **Select Variables**. The Select Variables window appears.

3 Select the check boxes for the variables that you want to add to the decision, and click **OK**.

4 (Optional) Select the **Output** check box if you want the value of the variable to be written to the output table that is generated when the decision is run.

TIP In the Global Variables category view, the **Value** column displays the value of the latest version of the variable. The **Activated Value** column displays the value of the currently active version. A check mark in the **Deleted** column indicates that the variable has been deleted from the list of global variables.

Create Custom Variables on the Variables Tab

Note: For information about data grid variables, see "Defining Data Grid Variables" in *SAS Intelligent Decisioning: Using Data Grids*.

To create custom variables on the **Variables** tab:

1 Click the **Decision Variables** subtab.

2 Click **Add variable** and select **Custom variable**. The Add Variables window appears.

3 Complete these steps for each variable that you want to add:

a Enter the name of the new variable, and select the data type of the variable. See "The Properties of a Variable" on page 222 for additional information.

TIP To re-add a variable that was used in a locked version, you must specify the same data type that was used in the previous version.

b (Optional) Click **Optional** to display the **Description**, **Initial value**, and **Length** fields.

c (Optional) Enter a length, initial value, and description for the new variable. Whether you can specify an intial value or length for the variable depends on the variable's data type. See "The Properties of a Variable" on page 222 for additional information.

TIP You can specify an expression as the intial value only for variables of type character, integer, decimal, or boolean. To enter an expression, click ✎ to open the expression editor. For more information, see "Using the Expression Editor" on page 33.

d. Click **Add**. SAS Intelligent Decisioning adds the new variable to the table of variables at the bottom of the window. By default, variables are added to the table as both input and output variables.

e. Verify that the **Input** and **Output** check boxes are selected correctly for each variable.

   - Clear the **Input** check box for any variable that you do not want to be mapped to a column in an input table or for which you do not want to specify a value.
   - Clear the **Output** check box for any variable that you want to exclude from the output data.
   - Clear both the **Input** and **Output** check boxes to create a temporary variable.

4. (Optional) Modify the variable properties in the table of variables.

5. Click **OK** to add the variables and close the Add Variables window.

Duplicate a Variable

1. On the **Variables** tab, click the **Decision Variables** subtab.
2. Select the variable that you want to duplicate, click ⋮, and select **Duplicate**. The Duplicate Variable window appears.
3. Enter a new name for the duplicate variable.
4. (Optional) Enter a description for the variable.
5. Click **Duplicate**.

Importing and Exporting Variables

Import Variables

You can import variables from either comma-delimited (CSV) files or from JavaScript Object Notation (JSON) files. These files must conform to formats described in Appendix 2, “Import File Formats,” on page 325. These formats are the same formats that are created when you export variables.

1. Open the decision into which you want to import variables.
2. On the **Variables** tab, click **Import**, and select either **Comma-delimited (*.csv)** or **JSON (*.json)**. The Import File window appears.
3. Click **Browse** and select the file from which you want to import variables.

4 Specify the encoding of the CSV file.

5 Select **Add variables** to append the variables to the current list of variables, or select **Replace variables** to replace the current list of variables with the imported variables.

TIP To add new columns to an existing data grid variable, select **Replace variables**.

Note: If you replace the list of variables in a decision and a variable is used in the decision is not included in the import file, the variable is removed from the list of variables. An error marker is added to the nodes that use that variable in the diagram on the **Decision Flow** tab.

6 Click **Import**, and then click 🖫 to import the variables and save the decision.

Export Variables

1 Open the decision from which you want to export variables.

2 On the **Decision Variables** tab, select the variables that you want to export.

3 Click **Export**, and select the file type to which you want to export the variables. The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Delete Variables

Note: You cannot delete a variable if it is used in the current version of an object. You can delete variables from the current version if they are used only in locked versions, but those variables are still included in the locked versions.

1 On the **Variables** tab, click the **Decision Variables** subtab.

2 Select the check box for the variables that you want to delete, click ⋮ , and select **Delete**.

Edit Variable Properties

On the **Variables** tab, click the **Decision Variables** subtab, and click on the name of the variable that you want to edit. The Edit Variable window appears. Edit the properties as needed, and then click **OK**.

Note: When you rename a variable, references to that variable within the same object change to use the new name. You cannot change the name or data type of a variable that is used in locked versions.

See "The Properties of a Variable" on page 222 for additional information.

Edit Metadata for Data Grid Variables

For information, see "Editing Data Grid Variable Metadata" in *SAS Intelligent Decisioning: Using Data Grids*.

Determine Which Objects Use a Particular Variable

1 On either the **Decision Variables** subtab or the **Global Variables** subtab, select the check box for the variable, click ⋮ , and select **View used by report**. The Select the Objects to Include in the Report window appears.

2 Select the objects within which you want to search for the selected variable. Select **Search the current object and all objects included within it** to search only the objects that are included in the current decision and in the objects that are used in the decision. Select **Search all objects in the SAS Intelligent Decisioning environment** to search all of the rule sets, decisions, and code files that are defined in your current environment.

3 Click **Search**.

In the report, you can use the **Filter** field to filter the list of objects based on the object names. If the variable that was selected in Step 1 above is a data grid variable, and if you are interested only in a specific column within the data grid, you can select the column in the **Column name** field, and then click **Apply**. SAS Intelligent Decisioning narrows the search results to only the objects that use the data grid with the selected column.

Click on an object name to open the object. Click ⓘ next to an object name to display the date on which the object was last modified and the ID of the user who modified the object. For decisions and code files that have been checked out, SAS Intelligent Decisioning also displays the IDs of the users that have checked out the objects. For decisions, SAS Intelligent Decisioning includes the decision's workflow status.

Click ⊞ next to nested decision names to display the list of objects within the nested decision that use the selected variable. Click ▷ and ◢ to open and close the list objects within the nested decision.

Click **Export** to export the Used By report to a PDF file. In the PDF file, you can click on an object name to open the object in a new browser tab.

Adding Objects to a Decision

Add an Existing Object

1 On the **Decision Flow** tab, click 📂 to open the Content Selection window, and then drag the object from the list onto the diagram where you want to add it.

On the **Decision** tab:

a Click **Add** and select the object type that you want to add to the decision. If an object in the decision is selected, select **Edit** ⇨ **Add**, and then select the object type. The Content Selection window appears.

TIP If you do not have permission to update an object, SAS Intelligent Decisioning displays `(Read-Only)` in the title bar next to the object name, and the buttons for modifying and saving the object are disabled.

b Select the object that you want to add, and click **OK**.

Note: The objects that are included in a decision determine the destination types to which you can publish the decision. For more information, see "Where Can I Publish Decisions?" on page 279.

IMPORTANT It is recommended that you do not add a decision as a nested decision within itself.

If you are using the default settings, and you have not already defined decision variables with the same name and data type as the object's input and output variables, SAS Intelligent Decisioning displays the following message: `Some objects in the decision define variables for which no corresponding decision variables have been created.`

A decision that includes an object must define decision variables to which you can map the object's variables. If a decision does not already have a variable of the same name and data type as an object's variable, then SAS Intelligent Decisioning displays the message. For more information, see "About Decision Variables and Mapping" on page 247.

2 (Optional) Click ⋮ in the toolbar, and select **Add missing variables** to add any missing variables to the decision. For more information, see “Add Missing Variables”.

3 (Optional) Reorder the objects in the decision. For more information, see “Editing the Objects in a Decision” on page 251.

4 (Optional) For rule sets, code files, models, and nested decisions, select the version of the object that you want to include in the decision in the **Properties** panel for the object. For models, if you select a specific version, the decision always uses that version, even if newer versions of the model are deployed. If you select **latest**, the decision always uses the most recently deployed version of the model.

> **TIP** You can change the version of most objects at any time by changing the version on the **Properties** panel. See “Edit the Properties of a Decision Node” on page 252 for more information. You cannot specify the version of a treatment group that is used in a decision. For more information, see “Content That Is Used by Tests and Scenarios for Decisions” on page 303 and “Content Executed by Published Decisions” on page 309.

5 (Optional) If you are adding an object that uses a function that is defined in a custom context file, specify the custom context file on the **Properties** tab. See “Associating a Custom Context File with a Decision” on page 164 for more information.

6 (Optional) On the **Properties** tab, select the **Subject ID** variable and the **Subject level** variable that you want to associate with subject contact records that are recorded by the decision.

> **TIP** If you do not specify the subject ID or subject level, the subject ID or subject level is recorded as `NONE` in the subject contact history.

For more information about subjects and subject contact histories, see the following topics:

- “About Treatments and Decisions” on page 84
- “Example: A Decision That Includes a Treatment Group” on page 85
- “Adding Record Contacts Nodes” on page 233
- “Predefined Lookup Tables” on page 116

7 Verify that the object’s variables are mapped to the correct decision variables. For more information, see “Mapping Variables within a Decision” on page 247.

Create and Add a New Object

On the **Decision Flow** tab, you can create and add new objects of any type except models. On the **Decision** tab, you can create and add new data query files, DS2 code files, or Python code files.

1 On the **Decision Flow** tab, click ⊡, and drag the new object type onto the diagram where you want to add it. The **New *object*** window appears.

  On the **Decision** tab:

  a Click **Add** and select **Data query**, **DS2 code file**, or **Python code file**. If an object in the decision is selected, select **Edit** ⇨ **Add**, and then select the object type. The Content Selection window appears.

  b Click **New *object type***. The **New *object type*** window appears.

2 Enter a name for the new object if you do not want to use the default name. File names are limited to 32 characters and must be unique within a folder.

3 (Optional) Enter a description. Descriptions are limited to 1000 characters.

> **TIP** You can edit the description at any time in the **Properties** panel.

4 Click 🗀, and select the folder in which you want to save the object.

5 (Optional) If you are creating a new data query file, select the editor that you want to use to create the file.

> **TIP** For data query files that you choose to edit in SAS Studio, SAS Intelligent Decisioning adds `.cqy.df` to the file name, and the file type is displayed as **Data Query**.
>
> For data query files that you choose to edit in the SQL editor, the file type is displayed as **SQL**. These files can return either a data grid or a single row of scalar variables. You can select the output type on the **Properties** tab.
>
> For more information, see "Query Output Types and Editors" on page 160.

6 Click **Save**. SAS Intelligent Decisioning creates a new object and adds it to the decision.

7 Click 🖫 to save the decision.

8 In the Properties panel on the **Decision Flow** tab, click **Open** to open the new file in the editor.

  Alternatively, on the **Decision** tab, click ☰, click the **Properties** tab, and then click **Open** to open the new file in the editor.

9 Define the content for the new object. For more information, see the following topics:

- Chapter 2, “Working with Rule Sets”
- Chapter 3, “Working with Treatments and Treatment Groups”
- Chapter 6, “Using Custom Code Files”
- Chapter 8, “Working with Decisions”

10 Click 🖫 and **Close** to save and close the new object.

If you added new variables to the object, SAS Intelligent Decisioning displays the message `Some objects in the decision define variables for which no corresponding decision variables have been created.`

11 (Optional) Click ⋮ in the toolbar, and select **Add missing variables** to add any missing variables to the decision. For more information, see “About Decision Variables and Mapping” and “Add Missing Variables”.

Adding Record Contacts Nodes

About Recording Contacts

The primary purpose of record contacts nodes is to record the outcome of a decision. You can use record contacts nodes to record the values of specific variables at specific points in a decision. For treatment groups, a record contacts node records metadata for the treatments.

IMPORTANT Decisions that include record contacts nodes cannot be published to container destinations.

Record contacts nodes record information when the decision that they are in is executed in the publishing destination. The behavior of record contacts nodes differs based on whether the decision is published to a SAS Micro Analytic destination or to a destination of another type.

- For decisions that are published to a SAS Micro Analytic Services destination, the record contacts node writes a record to the subject contact history. The record contains the information that you specified that you want to track, such as the data grid of treatments returned by the decision and the values of other variables. In addition, the record contacts node creates an output variable, the record contacts variable, that contains a response tracking code. An application, such as a customer service application, can use this response tracking code to add additional data to the subject contacts history.
- For destinations of other types, the record contacts node does not write a record to the subject contact history. Instead, the information that you specified that you want to track is written to the record contacts variable. The name of this variable is based on the name of the record contacts node: `rt_`*`node-name`*. This variable is a character string, and it also contains the response tracking code. You can issue a POST request to post the data to the subject contact history. The POST request type is **`application/vnd.sas.decision.subject.contact+json`**. For more information, see the Subject Contacts API documentation.

A decision can contain multiple record contacts nodes. For example, your decision might have different paths for different channels, and you might want a record contacts node on each path.

Add a Record Contacts Node

1 On the **Decision Flow** tab, select , and drag the **Record Contacts** object onto the diagram where you want to add it. SAS Intelligent Decisioning opens the **Properties** panel for the node.

  Alternatively, on the **Decision** tab, select **Add** ⇨ **Record contacts**. If an object in the decision is selected, select **Edit** ⇨ **Add below** ⇨ **Record contacts**.

2 (Recommended) Rename the record contacts node. Record contacts node names must be unique within the decision. To rename the node:

  a On the **Decision Flow** tab, click ⋮ for the node, and select **Rename**. Alternatively, on the **Decision** tab, select the node, and select **Edit** ⇨ **Rename**. The Rename window appears.

  b Enter a new name and click **Rename**.

  TIP When a decision is executed in a destination other than a SAS Micro Analytic Service destination, the record contacts node writes information to a variable. The name of this variable is based on the name of the record contacts node: `rt_`*`node-name`*. If you define a variable with this name on the **Variables** tab, the data in your decision variable will be overwritten by the record contacts node. To avoid this, rename either the record contacts node or the decision variable.

3 Click to edit the properties of the record contacts node.

4 (Optional) Select the variable that contains the channel information to which you want to attribute the contact data that is recorded by the node.

5 (Optional) Select the variables whose values you want to record.

  Note: You cannot track data grids in record contacts node because of the amount of space required to store the data. For character variables, SAS Intelligent Decisioning stores only the first 4000 characters of the value.

6 (Optional) Select **Track treatments** if you want to record which treatments are sent to the calling application. Then, select the data grid variable for the specific set of treatments that you want to track.

  TIP If you specify a subject ID or subject level in the decision properties, this information is included in the records that are generated for the record contacts node. If you do not specify the subject ID or subject level, the subject ID or subject level is recorded as `NONE`.

7 (Optional) Clear the **Record rule-fired data** check box if you do not want to record the rule-fired information for all rules in the decision up to the point at which the record contacts node is included.

Note: This rule-fired data is recorded in the subject contact history. The rule-fired data that is recorded when you select **Record rule-fired data** when you publish a rule set or decision is recorded in the ruleFiredFlags column in the output table.

8 (Optional) Clear the **Record path tracking** check box if you do not want to record the path-tracking information for all nodes in the decision up to the point at which the record contacts node is included.

Note: This path-tracking data is recorded in the subject contact history. The path-tracking data that is recorded when you select **Record path tracking** when you publish a decision is recorded in the pathID column in the output table.

9 (Optional) Clear the **Include in contact policy** check box if you do not want this contact record included in aggregate reports for the channel.

10 Click **OK** to save your changes.

Adding Branches to a Decision

Overview of Branches

Branches enable you to add conditional logic to a decision. Depending on the branch type, a branch can have multiple outgoing paths. You can add four types of branches to a decision:

- “About Yes/No Branches” on page 236
- “About Equals Branches” on page 238
- “About Range Branches” on page 239
- “About Like Branches” on page 240

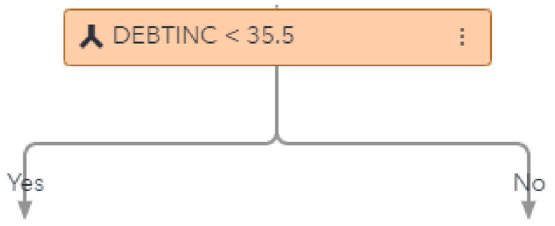
The conditions for the branch paths are evaluated in the order in which you specify them. The first path whose condition evaluates to True is taken by the executing decision. The conditions for the remaining branches are not evaluated.

You can change the labels used for branch paths in Equals, Range, and Like branches. For more information, see “Customize Branch Path Labels” on page 243.

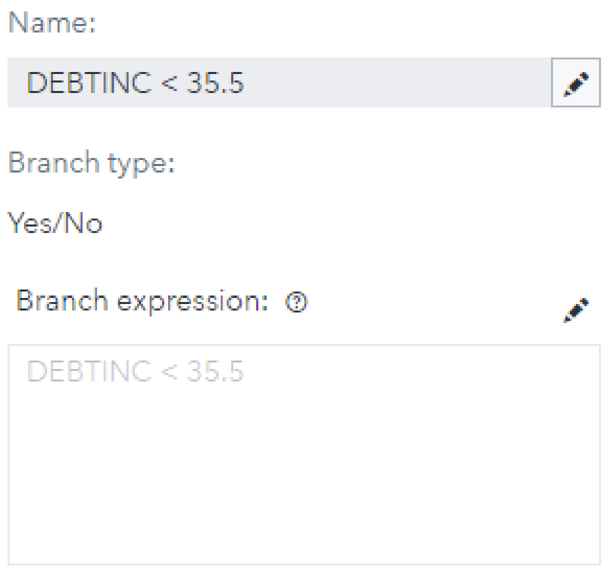
Adding Yes/No Branches

About Yes/No Branches

A Yes/No branch expression is evaluated as a Boolean expression. A Yes/No branch has a single condition and two outgoing paths: Yes (True) and No (False). The default name for a Yes/No branch is **Yes/No**, but you can specify a custom name. For example, the following branch tests whether the value of the DEBTINC variable is less than the constant 35.5. The name in the diagram is customized to display the condition that is being tested.



The following figure shows the property panel for this branch:


Name:

DEBTINC < 35.5

Branch type:

Yes/No

Branch expression:

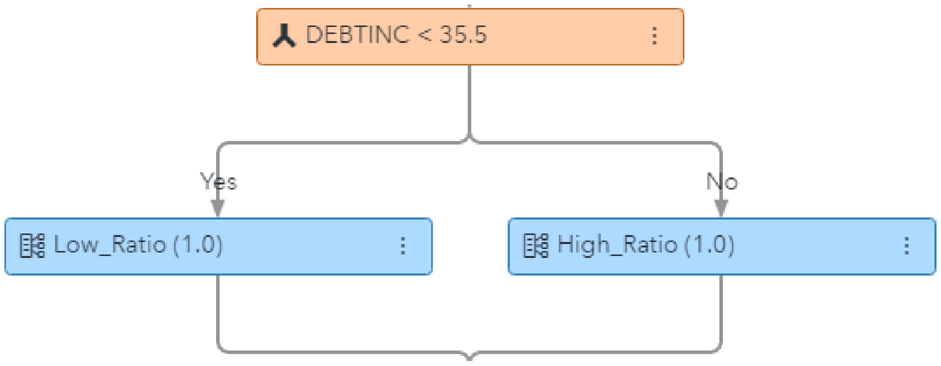
DEBTINC < 35.5

For more information, see “Add a Yes/No Branch on the Decision Flow Tab” on page 237 and “Yes/No Branches and THEN/ELSE Clauses” on page 236.

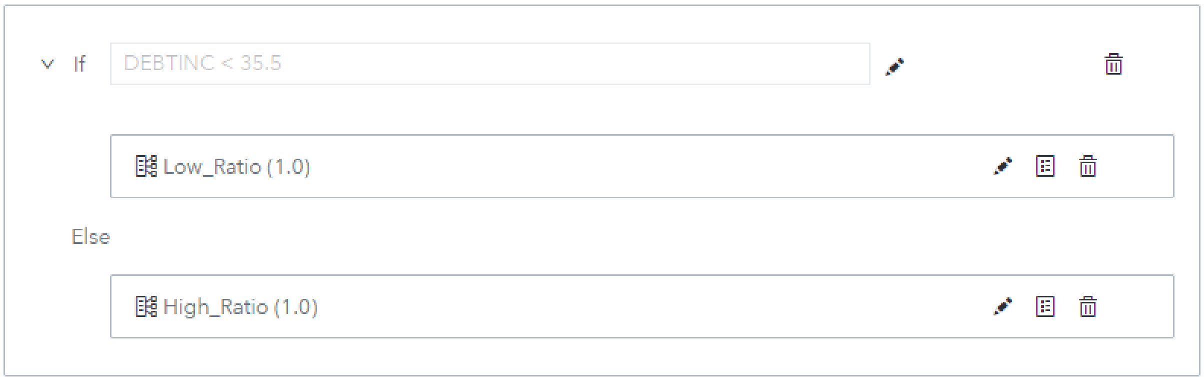
Yes/No Branches and THEN/ELSE Clauses

Objects in a THEN clause on the **Decision** tab correspond to nodes in a **Yes** branch path on the **Decision Flow** tab. Objects in an ELSE clause correspond to nodes in a **No** branch path.

For example, suppose you have the following nodes on the **Decision Flow** tab:



The same nodes appear on the **Decision** tab as an IF-THEN-ELSE statement:



Add a Yes/No Branch on the Decision Flow Tab

Note: When you add a Yes/No branch, any objects that follow the currently selected object become part of the No path.

1 Select ⊡, and drag the **Branch** object from the list of objects onto the diagram where you want to add it. The Create New Branch window appears.

2 (Optional) Enter a name for the branch if you do not want to use the default name. The default name is **Yes/No**.

3 Select **Yes/No**, and click **OK**. SAS Intelligent Decisioning adds the branch and opens the **Properties** panel.

4 On the **Properties** panel, click ✎. SAS Intelligent Decisioning opens the expression editor.

5 Define the expression for the branch. See "Using the Expression Editor" on page 33 for additional information.

6 Click **Save** in the expression editor to save the expression and return to the decision editor.

7 (Optional) Add objects to the Yes and No branch paths. Right-click on the branch node, and select either **Add to yes path** ⇨ ***object*** or **Add to no path** ⇨ ***object***.

For more information, see the following topics:

- "Add an Existing Object" on page 230
- "Create and Add a New Object" on page 232
- "Adding Record Contacts Nodes" on page 233

Add a Yes/No Branch on the Decision Tab

Note: When you add a Yes/No branch, any objects that follow the currently selected object become part of the ELSE clause.

1 Select **Add** ⇨ **Branch**. If an object in the decision is selected, select **Edit** ⇨ **Add** ⇨ **Branch**. The Create New Branch window appears.

2 (Optional) Enter a name for the branch.

3 Select **Yes/No**, and click **OK**. SAS Intelligent Decisioning adds an IF-THEN-ELSE statement to the decision.

4 For the IF condition, click ✎. SAS Intelligent Decisioning opens the expression editor.

5 Define the expression for the branch. See "Using the Expression Editor" on page 33 for additional information.

6 Click **Save** in the expression editor to save the expression and return to the decision editor.

7 (Optional) Add objects to the THEN and ELSE clauses. Click **Edit** and select either **Add to yes path** ⇨ ***object-type*** or **Add to no path** ⇨ ***object-type***.

   For more information, see the following topics:

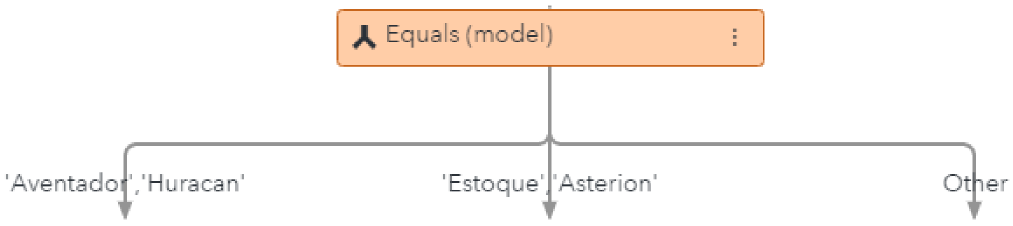
   - "Add an Existing Object" on page 230
   - "Create and Add a New Object" on page 232
   - "Adding Record Contacts Nodes" on page 233

Adding Equals, Range, or Like Branches

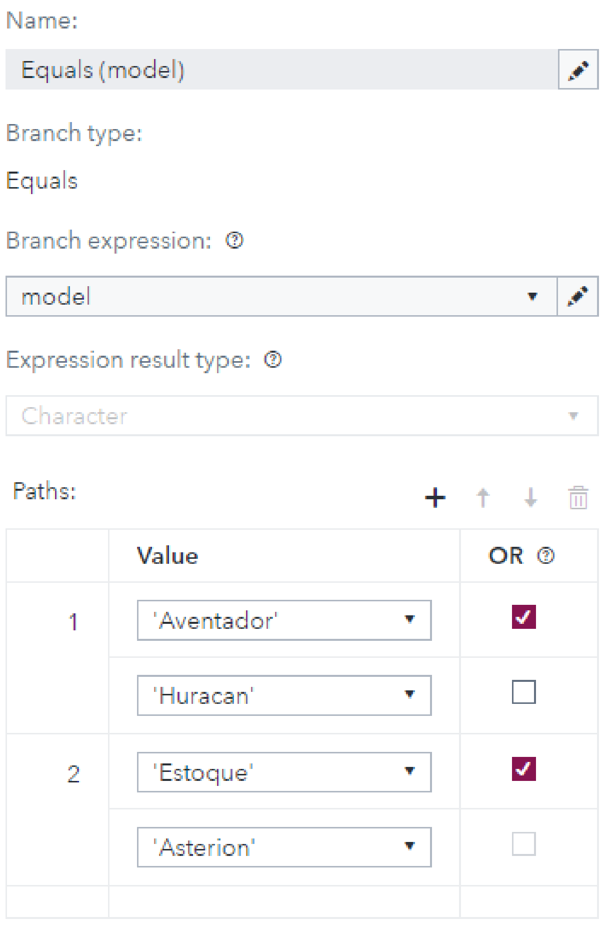
About Equals Branches

An Equals branch compares the value of the branch expression to other variables or to literal values. By default, this branch has one outgoing path for each comparison variable or literal value, plus a branch labeled **Other** for any values that are not included in the branches that you create. However, you can combine an outgoing path with the path that immediately follows it by selecting the **OR** check box.

For example, the following branch compares the value of the character variable MODEL to the string 'Aventador', then to 'Huracan', and so on, in the order listed in the property panel. The first two comparison strings and the last two strings are combined into one path by using the OR operator.



The following figure shows the properties panel for this branch:



Name:

Equals (model)

Branch type:

Equals

Branch expression:

model

Expression result type:

Character

Paths:

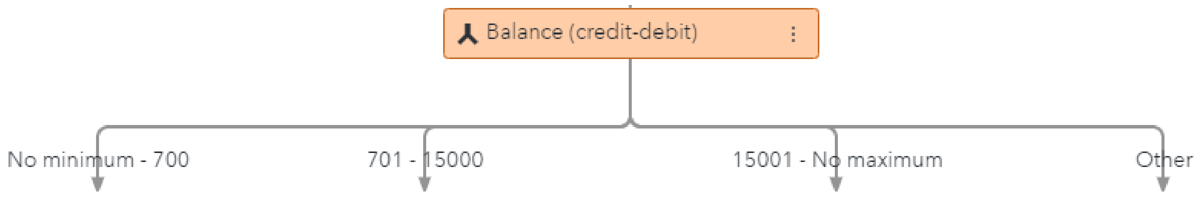
	Value	OR
1	'Aventador'	✓
	'Huracan'	☐
2	'Estoque'	✓
	'Asterion'	☐

For more information, see “Add Equals, Range, or Like Branches” on page 241.

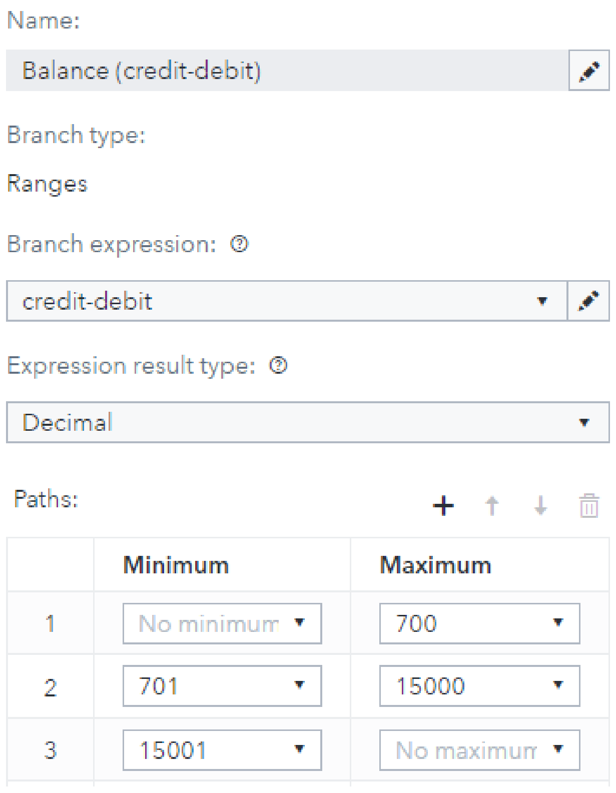
About Range Branches

A Range branch compares the value of the branch expression against one or more ranges of values. You can specify range values by using variables or constants. This branch has an outgoing path for each range, plus a branch labeled **Other** for any values that are not included in the branches that you create.

For example, the following branch compares the value of the expression `credit-balance` to three ranges. The first range has no minimum value, so it is treated as if the minimum is negative infinity. The third branch has no maximum value, so it is treated as if the maximum is positive infinity.



The following figure shows the properties panel for this branch:



Name:

Balance (credit-debit)

Branch type:

Ranges

Branch expression:

credit-debit

Expression result type:

Decimal

Paths:

	Minimum	Maximum
1	No minimum	700
2	701	15000
3	15001	No maximum

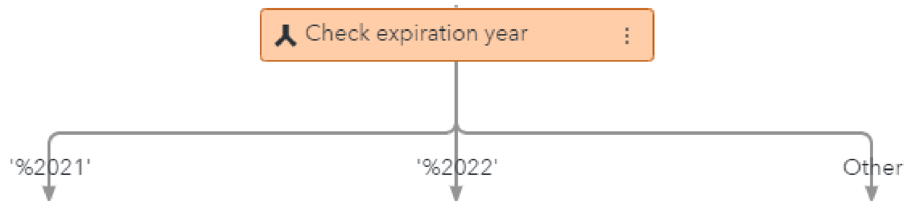
For more information, see "Add Equals, Range, or Like Branches" on page 241.

About Like Branches

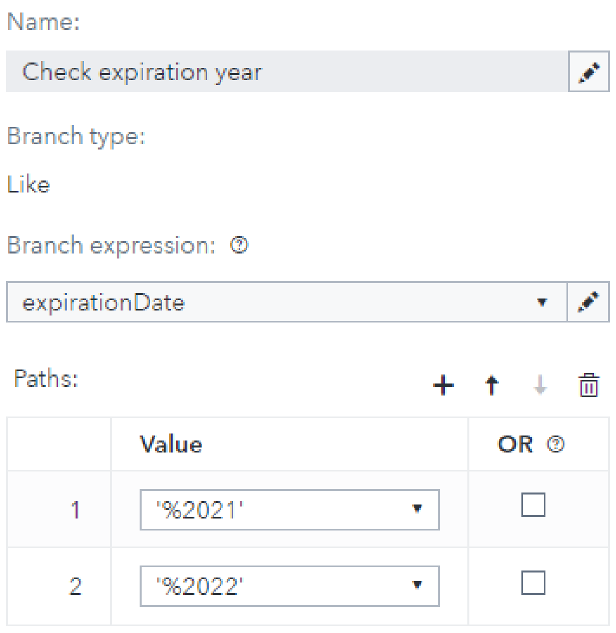
A Like branch compares the value of the branch expression against one or more strings by using the LIKE operator. In LIKE expressions, you can use the underscore (_) and percent (%) characters as wildcards. For more information, see "Using the LIKE Operator" on page 38.

By default, this branch has one outgoing path for each comparison string, plus a branch labeled **Other** for any values that are not included in the branches that you create. You can combine paths by selecting the **OR** check box.

For example, the following branch compares the last four characters of the variable EXPIRATIONDATE to determine whether it ends with the characters "2021" or "2022".



The following figure shows the properties panel for this branch:



Name:

Check expiration year

Branch type:

Like

Branch expression:

expirationDate

Paths:

	Value	OR
1	'%2021'	☐
2	'%2022'	☐

For more information, see “Add Equals, Range, or Like Branches” on page 241.

Add Equals, Range, or Like Branches

1 Select ⊡, and drag the **Branch** object from the list of objects onto the diagram where you want to add it. The Create New Branch window appears.

2 (Optional) Enter a name for the branch.

3 Select **Equals**, **Range**, or **Like**, and click **OK**. SAS Intelligent Decisioning adds the branch and, if you are working on the **Decision Flow** tab, opens the **Properties** panel.

4 If you are working on the **Decision** tab, click ☰ to open the Properties window.For more information, see “The Decision Flow Tab versus the Decision Tab” on page 221.

5 In the **Properties** panel or window, click ✎ to open the branch expression editor.

6 Define the expression for the branch. See “Using the Expression Editor” on page 33 for additional information.

7 If you are defining an Equals branch or a Range branch, select the **Expression result type**. For Like branches, the result of the expression must be a character value.

8 For each outgoing path that you want to add to the branch node, complete the following steps:

a Click + to add the path.

Note: SAS Intelligent Decisioning automatically adds a path labeled **Other**.

b Specify the values or expressions for the path.

Branch Type	Values or Expressions
Equals	Specify a comparison variable or constant of the branch variable for the path. See "About Equals Branches" on page 238 for more information.
Range	Specify a minimum value, a maximum value, or both for the branch variable. Minimum and maximum values are optional. If you do not specify a minimum value, the minimum is treated as negative infinity. If you do not specify a maximum value, the maximum is treated as positive infinity. If the minimum and maximum are the same value, the path is followed only if the branch variable is equal to that value. The minimum and maximum boundary values are inclusive. If the minimum or maximum values for any ranges are static and overlapping, SAS Intelligent Decisioning displays a warning, but you can still publish the decision. See "About Range Branches" on page 239 for more information.
Like	Specify the LIKE expression for the path in single quotation marks. See "Using the LIKE Operator" on page 38 and "About Like Branches" on page 240 for more information.

9 (Optional) Click ↑ or ↓ to reorder the paths. Branch paths are evaluated in order. The decision follows the first path that evaluates to True.

10 (Optional) Combine branch paths. For Equals and Like branches, you can select the **OR** check box to combine a branch path with the path that immediately follows it. The application combines the conditions for the branch paths by using the OR operator.

11 If you are working on the **Decision** tab, click **OK** to add the branches to the decision and close the **Properties** panel.

12 In the decision diagram, add objects to the outgoing branch paths.

On the **Decision Flow** tab, right-click on the branch node, and select **Add to branch path** ⇨ ***Branch label*** ⇨ ***Object type***.

On the **Decision** tab, select **Edit** ⇨ **Add to branch path** ⇨ ***Branch label*** ⇨ ***Object type***.

SAS Intelligent Decisioning opens the appropriate window, depending on the object type.

For more information, see the following topics:

- “Add an Existing Object” on page 230
- “Create and Add a New Object” on page 232
- “Adding Record Contacts Nodes” on page 233

Customize Branch Path Labels

You can change the labels that are used for branch paths in Equals, Range, and Like branches.

1. In the **Properties** panel for the branch, click **More**. The More Branching Properties window appears.
2. Edit the labels in the **Alternate Label** field for each branch path that you want to change, and click **Close** to close the More Branching Properties window.

TIP You can also change the branch variable and search branch path values in the More Branching Properties window.

Rename Branches

The default name for a branch is the branch type, such as "Equals" or "Yes/No". Multiple branches in the same decision can have the same name.

To rename any branch type:

1. Select the branch.
2. In the **Name field** on the **Properties** panel for the branch, click ✎
3. Enter the new name, and click **Rename**.

Alternatively, to rename an Equals, Range, or Like branch:

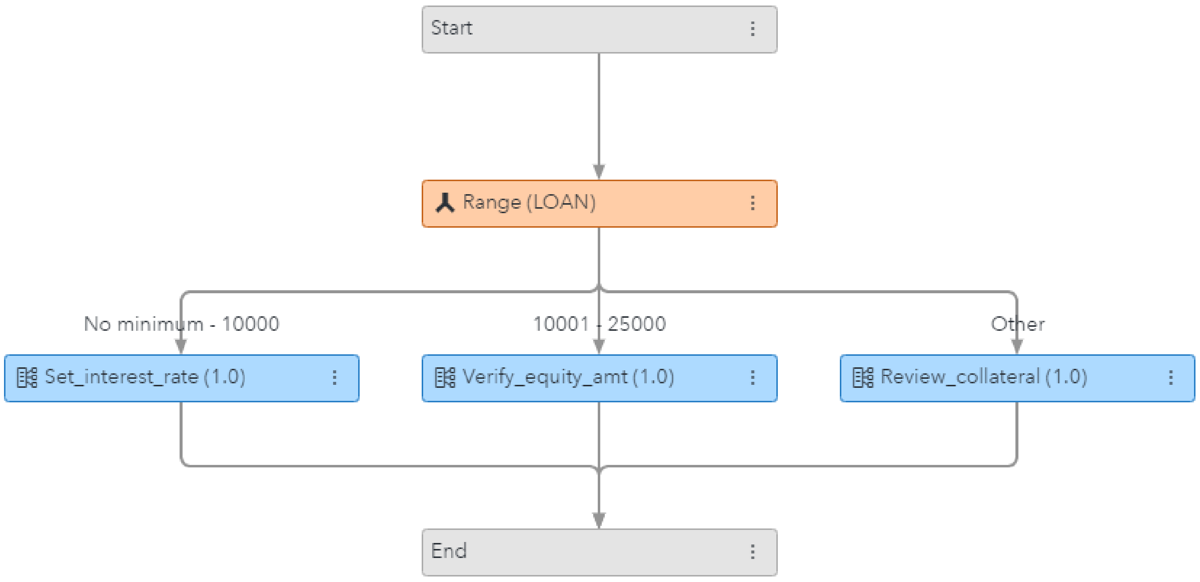
1. Right-click on the branch node, and select **Rename**. The Rename window appears.
2. Enter the new name, and click **Rename**. The maximum length is 100 characters.

Adding Cross-Branch Links

Overview of Cross-Branch Links

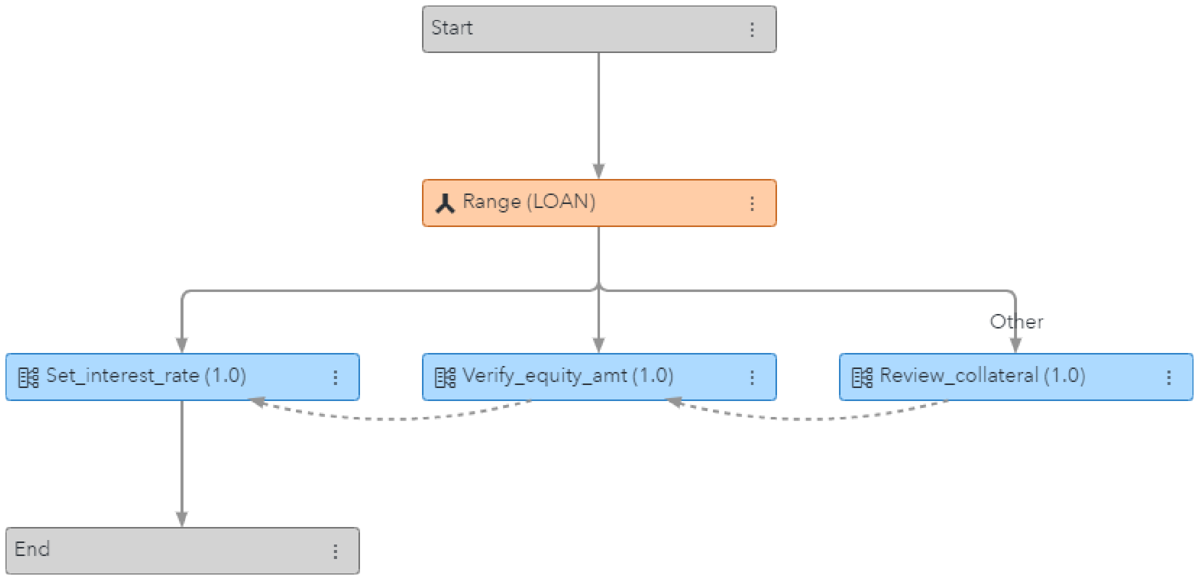
Cross-branch links enable you to easily reuse components in a decision. You can add a cross-branch link to a target node at a branch level that is the same as or lower than the level of the source node. You can add cross-branch links that link to the End node, but you cannot add a cross-branch link that ends at the Start node.

For example, given the decision shown in the following figure, you can add cross-branch links between any of the rule sets.



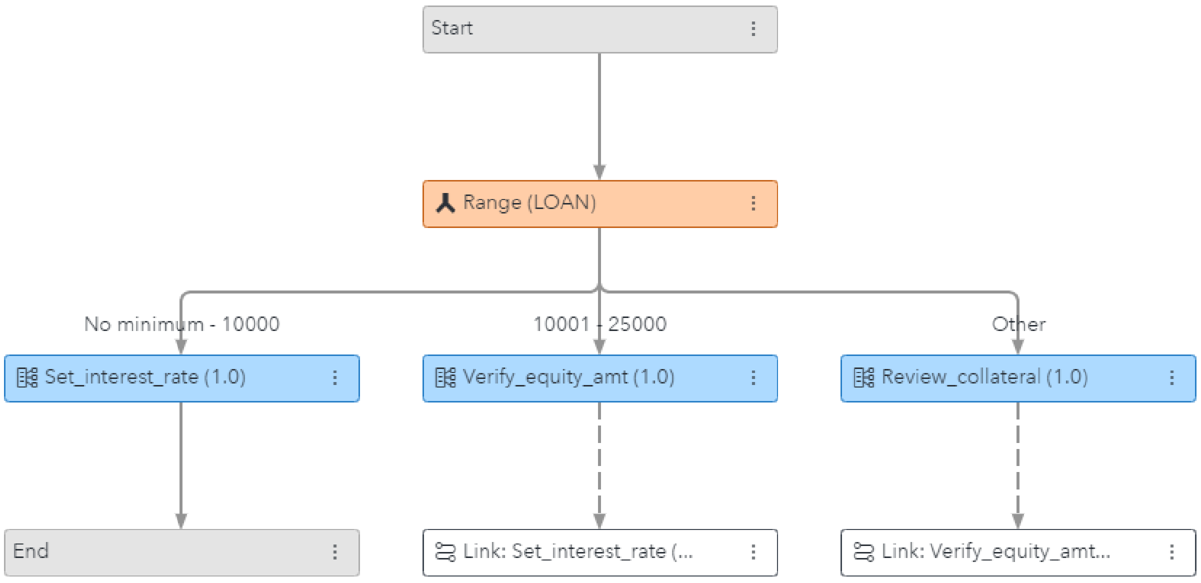
The Range branch has one path for each of three different ranges of loan values. Suppose you want all three paths to include the Set_interest_rate rule set, and you want both the mid-range and Other path to include the Verify_equity_amt rule set. You can add a cross-branch link from the Review_collateral node to the Verify_equity_amt node and a cross-branch link from the Verify_equity_amt node to the Set_interest_rate node.

Figure 8.1 *Cross-Branch Links Displayed as Arcs*



You can display cross-branch links as either arcs or nodes. By default, cross-branch links are displayed as arcs. To switch between the two views, click in the upper right corner of the diagram.

Figure 8.2 *Cross-Branch Links Displayed as Nodes*



TIP To remove or edit a cross-branch link that is displayed as an arc, right-click on the dashed line and select **Remove** or **Properties**.

When you select a cross-branch link node, both the selected node and the link's target node are highlighted.

Add a Cross-Branch Link

1 Right-click the source node (the node where you want the link to start), and select **Add** ⇨ **Cross-branch link**. The New Cross-branch Path window appears. This window displays the nodes that you can link to from the current source node. When you select a target node in the New Cross-branch Path window, that node is highlighted in the decision diagram.

2 Select the node to which you want to link, and click **OK**.

Add a Micro Analytic Module

When you publish content to a SAS Micro Analytic Service destination, SAS Intelligent Decisioning generates code for the content and writes that code as a module in SAS Micro Analytic Service. You can use a Micro Analytic Module node in a decision to call a SAS Micro Analytic Service module. Micro Analytic Module nodes use the MASCall package to call published modules. You can call the modules only for published decisions.

When you add a Micro Analytic Module node to a decision, the decision always uses the most recently published version of the selected module. You do not need to edit and republish the decision in order to pick up newly published versions of the selected modules.

Note: SAS Micro Analytic Service modules that are referenced in a Micro Analytic Module node can use data grids as local variables, but the data grids cannot be passed as input or output parameters to the module.

To execute a decision that contains a Micro Analytic Module node, the setting of the SAS Intelligent Decisioning sas.decisions.masnode.removeTrailingUnderscoresFromInput configuration option must match the setting of the SAS Micro Analytic Service configuration option service.removetrailingunderscoresfrominputs.

IMPORTANT You can publish decisions that contain Micro Analytic Module nodes only to SAS Micro Analytic Service destinations and Git destinations that are compatible with SAS Micro Analytic Service.

Using the %DCM_EXECUTE_DECISION macro to execute in batch a decision that contains a Micro Analytic Module node is not supported. Execution times and performance might be unacceptable.

To add a Micro Analytic Module node to a decision:

1 On the **Decision Flow** tab, click ⊡, and drag the **Micro Analytic Module** object onto the diagram where you want to add it. Alternatively, on the **Decision** tab, select **Add** ⇨ **Micro Analytic Module**. If an object in the decision is selected, select **Edit** ⇨ **Add below** ⇨ **Micro Analytic Module**.

The Choose a SAS Micro Analytic Module window appears.

2 Click 🗀, and select the decision whose module you want to call.

3 Click **Browse**. SAS Intelligent Decisioning displays the list of published modules for the selected decision.

4 Select the module that you want to use, and then click **OK**.

For more information about MASCall packages, see “Performing Calls between SAS Micro Analytic Service Modules” in *SAS Micro Analytic Service: Programming and Administration Guide*.

Mapping Variables within a Decision

About Decision Variables and Mapping

The objects that you add to a decision, such as models, treatment groups, and other decisions, each define their own variables. Each decision that includes an object must define decision variables to which you can map the object’s variables.

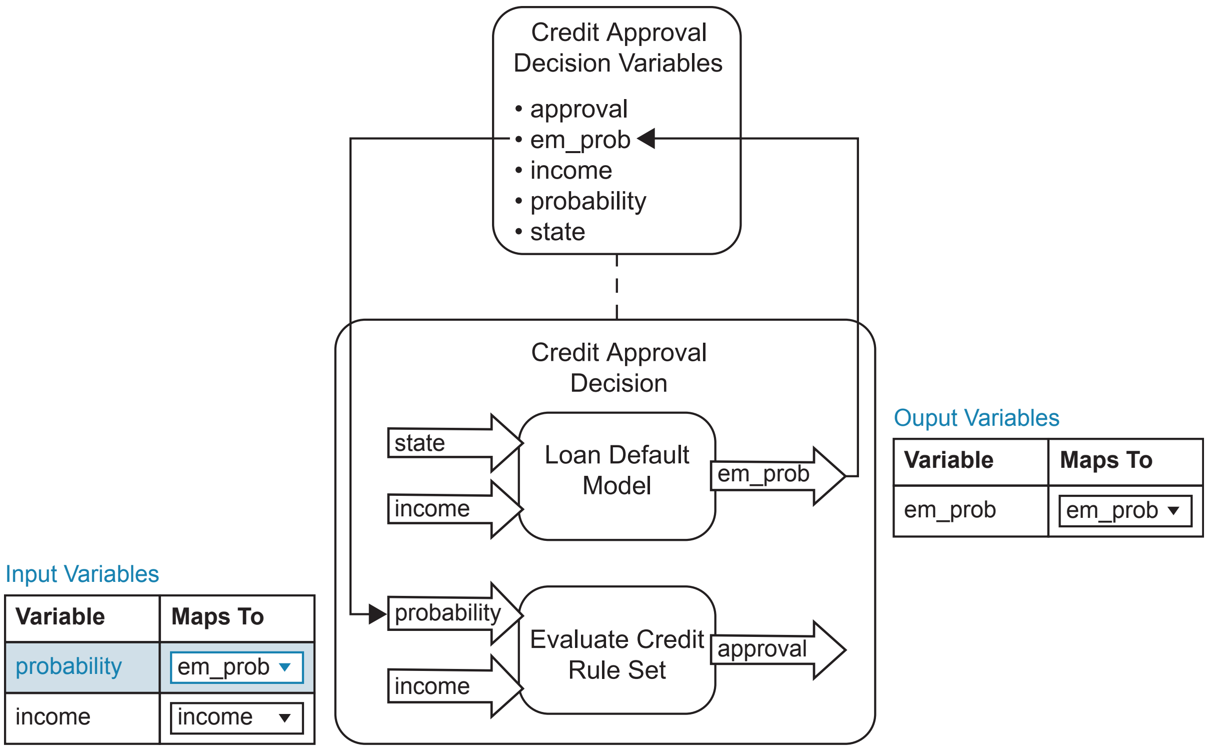
By default, if a decision does not already have a variable of the same name and data type as an object's variable, then when you add the object to a decision, SAS Intelligent Decisioning displays the following message: `Some objects in the decision define variables for which no corresponding decision variables have been created.` You can use the **Add missing variables** option to add the missing decision variables, or you can turn on the **Create variables automatically in decisions** setting if you want SAS Intelligent Decisioning to automatically create decision variables. For information, see “Add Missing Variables” on page 248 and “SAS Intelligent Decisioning Settings” on page 8. If you turn on the **Create variables automatically in decisions** setting, SAS Intelligent Decisioning does not display the message.

When you add a treatment group to a decision, SAS Intelligent Decisioning creates a decision variable of type data grid with the name *group_name*_out. If the treatment group name is longer than 32 characters, then this name is truncated.

When you add a data query file to a decision, the decision variables that are created depend on whether the query selects data and on whether the file contains the comment `/* include sqlReturnInfo */`.For more information, see “Decision Variables for Data Query Files” on page 161.

You can create decision variables with different names, and then change the decision variable mappings on the **Input Variables** and **Output Variables** property panels for objects in the decision.

For example, suppose you have a decision named Credit Approval, and this decision contains a model named Loan Default and a rule set named Evaluate Credit. The model has an output variable named **em_prob**. The value of this variable must be passed as input to the Evaluate Credit rule set, but the rule set is expecting a variable named **probability**. In order for the value to be passed to the rule set, you must map the **em_prob** output variable of the model to the **em_prob** decision variable, and you must map the **probability** input variable of the rule to the **em_prob** decision variable.



Input Variables

Variable	Maps To
probability	em_prob ▾
income	income ▾

Ouput Variables

Variable	Maps To
em_prob	em_prob ▾

IMPORTANT When the decision is published and run in a production environment, the decision expects the input data to contain variables that have the same name and data type as the decision's input variables.

Add Missing Variables

If the **Create variables automatically in decisions** setting is turned off, SAS Intelligent Decisioning does not automatically add decision variables when you add an object to the decision. In these cases, you can use the **Add missing variables** option to define decision variables with the same name and data type as the object's variables and add them to the decision.

1 Click ⋮ and select **Add missing variables**.

2 In the Add Missing Variables window, select the variables that you want to add to the decision, and click **Add**. SAS Intelligent Decisioning adds the decision variables and maps the object's variables to the decision variables.

TIP You can turn on the **Create variables automatically in decisions** setting if you want SAS Intelligent Decisioning to automatically create decision variables. For more information, see "SAS Intelligent Decisioning Settings" on page 8 and "Mapping Variables within a Decision" on page 247.

If you change a setting, you must sign out and sign back in to SAS Intelligent Decisioning in order for the changes to take effect.

Scoring Rows in a Data Grid

Data grid variables are processed like any other character variable unless you do one of the following:

- use data grid functions to process the individual data values within the data grid. For more information, see "Using Data Grid Functions" in *SAS Intelligent Decisioning: Using Data Grids*.
- use the **Score rows in this data grid** option to process each row in the data grid.

To process the rows within a data grid, select the **Score rows in this data grid** option on the **Input Variables** property panel when you are mapping decision variables for the node. If you select this option, you can map the node's input variables to columns in the data grid instead of to columns in the input table. SAS Intelligent Decisioning changes the default variable mappings to columns in the data grid if columns exist that have the same names as the node's input variables. You can customize the variable mappings as needed. When you select this option, the decision node processes all of the rows in the data grid before execution moves to the next node in the decision.

IMPORTANT When the node object is a filtering rule set, and you select **Score rows in this data grid**, rows that do not meet the criteria defined by the rules are removed from the data grid.

Mapping Data Grid Variables

When you add an object to a decision and the object contains a data grid variable, SAS Intelligent Decisioning creates a decision variable for the data grid in the same way that it creates decision variables for object variables of other data types. When you select **Score rows in this data grid** for an object that uses a data grid, you can choose to map the columns in the object's data grid variable either to columns in the decision's data grid variable or to other decision variables. In the lists of variables in the **Input Variables** property pane, the decision's scalar variables are identified by the ⛙ icon, and the decision's data grid columns are identified by the ▦ icon.

For more information, see "Map Object Variables to Decision Variables" on page 250 and "Scoring Rows in a Data Grid" on page 249.

Map Object Variables to Decision Variables

On the **Decision Flow** tab, complete these steps:

1 Select the object whose variables you want to map.

2 (Optional) Click to open the **Input Variables** property panel.

3 (Optional) Select **Score rows in this data grid**, and select the name of the data grid.

Note: This option appears only for objects that process data grid variables. For more information, see "Scoring Rows in a Data Grid" on page 249.

4 (Optional) For each object variable in the **Input Variable** column, select the decision variable that you want to map it to in the **Maps To** column.

5 (Optional) Click .

6 If the object that you selected is a treatment group, select the data grid that contains the outcome of the treatment group.

7 For each object variable in the **Output Variable** column, select the decision variable that you want to map it to in the **Maps To** column. For information about mapping variables for data query files, see "Decision Variables for Data Query Files" on page 161.

On the **Decision** tab, complete these steps:

1 Select the object for which you want to map variables.

2 Click to open the Properties window.

3 (Optional) Select **Score rows in this data grid**, and select the name of the data grid.

Note: This option appears only for rule sets, models, and decisions that process data grid variables. For more information, see "Scoring Rows in a Data Grid" on page 249.

4 For each object variable in the **Input Variable** column, select the decision variable that you want to map it to in the **Maps To** column.

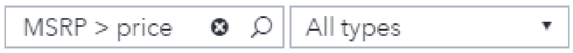
5 If the selected node is a treatment group, select the data grid that contains the outcome of the treatment group.

6 For each object variable in the **Output Variable** column, select the decision variable that you want to map it to in the **Maps To** column. For information about mapping variables for data query files, see "Decision Variables for Data Query Files" on page 161.

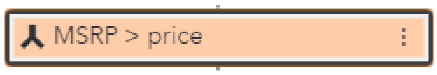
7 Click **OK** to close the Properties window.

Searching for Objects in a Decision Diagram

Use the Search field and the object type drop-down list on the **Decision Flow** tab to search for specific objects or object types in decision diagrams. To search for a specific object, enter the object name in the Search field. To search for object types, select the object type from the drop-down list. Use both fields to narrow the search. For example, to search for all objects with the name “MSRP > price”, enter the following criteria:



For any object that matches the search criteria, SAS Intelligent Decisioning highlights the object node with a thicker, darker outline.



Editing the Objects in a Decision

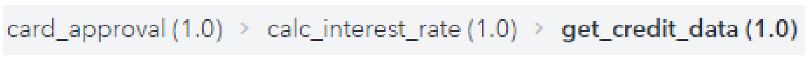
Open an Object from within a Decision

On a node in the **Decision Flow** diagram, double-click on the object name, or click ⋮ for the object that you want to open, and select **Open**.

On the **Decision** tab, select the object that you want to open, click **Edit**, and select **Open**.

The object opens in the appropriate editor. Models open in SAS Model Manager if you have access to that application.

When you object an object on the **Decision Flow** tab, the decision name at the top of the window is replaced with a breadcrumb. When you close an object, SAS Intelligent Decisioning returns to the previous object in the breadcrumb. You can click on any object in the breadcrumb to switch to that object. Any object that appears later in the breadcrumb is automatically closed. For example, if you open version one of the card_approval decision, the calc_interest_rate nested decision, and the get_credit_data rule set, the breadcrumb appears as follows:


card_approval (1.0) > calc_interest_rate (1.0) > get_credit_data (1.0)

TIP If you do not have permission to update an object, SAS Intelligent Decisioning displays `(Read-Only)` in the title bar next to the object name, and the buttons for modifying and saving the object are disabled.

Edit the Properties of a Decision Node

On the **Decision Flow** tab:

1. Select the decision node.
2. (Optional) Click to display the **Input Variables** panel, and modify the input variable mappings for the node.
3. (Optional) Click to display the **Output Variables** panel, and modify the output variable mappings for the node.
4. (Optional) Click to display the **Properties** panel.
   a. (Optional) Select a different object or object version for the selected node.
   b. (Optional) For code file nodes, modify the node description, or click **Open** to edit the code file in the appropriate editor.
   c. (Optional) For cross-branch links, select a different target node.

On the **Decision** tab:

1. Click for the node. The properties window for the node appears.
2. (Optional) On the **Variables** tab, modify the variable mappings.
3. (Optional) On the **Properties** tab, select a different object or object version. For code files, modify the description, or click **Open** to edit the file in the appropriate editor.
4. Click **Close** to save your changes and close the properties window.

Reorder Objects

On the **Decision Flow** tab, you can drag rule sets, branches, models, treatment groups, nested decisions, and code files from one position to another. You cannot move the nodes for cross-branch links, but you can move the target node of a cross-branch link.

On the **Decision** tab, to move an object up or down, including into and out of conditions, select the object and click ↑ or ↓. You can also move the selected object by using Shift + ↑ and Shift + ↓.

Remove an Object from a Decision

On the **Decision Flow** tab, click ⋮ on the object that you want to remove, and select **Remove**. To delete a cross-branch link that is displayed as a dashed line, right-click on the dashed line and select **Remove**.

On the **Decision** tab, click 🗑 on the object that you want to remove.

If you delete the target of a cross-branch link, the link adopts the next node between itself and the End node. If the next node is the End node, the link is deleted.

Replace an Object

To replace an existing object with a different object of the same type, click ✎ on the **Decision** tab. In the window that appears, select the new object, and click **OK**.

To change the target node of a cross-branch link, open the **Properties** panel for the link, click 🗀, and select a new target node.

Update Decisions to Use New Object Versions

You can update the versions of any of the objects that are used in a decision.

1 Open the decision, click ⋮, and select **Update object versions**. For each object that is included in the decision, SAS Intelligent Decisioning determines whether newer versions of the objects are available, and it displays the results in the Update Versions window.

2 (Optional) In the Update Versions window, select the objects that you want to update, or select **Select all** to update all of the objects.

3 Click **Update** to update the object versions that are used in the decision.

TIP To upgrade multiple decisions that use the same object, see

- "Upgrade Decisions to Use a New Version of a Rule Set" on page 52
- "Upgrade Decisions to Use a New Version of a Code File" on page 181
- "Upgrade Decisions to Use a New Version of a Nested Decision" on page 270

Validate a Decision in the Decision Editor

To validate a decision in the decision editor, open the decision. Then, on the **Decision Flow** or **Decision** tab, click **Validate**. SAS Intelligent Decisioning validates everything that can be determined without executing the decision. The following list describes some of the validations that SAS Intelligent Decisioning performs:

- The rule sets that are used in the decision contain no errors, such as missing required values or branch values that overlap.
- The model type of each model that is used in the decision is a supported type, and DS2 code can be generated for the model.
- Treatment groups, lookup tables, and global variables that are used in the decision have been activated.
- The objects that are specified in each of the decision nodes exist, and the current user has permission to access the objects.
- Object variables are mapped to decision variables on page 247.

Error messages are shown in the **Errors** panel and are categorized by error type. For many error types, this panel is updated as you fix the errors. For some errors, such as errors in nested decisions or activation errors, you must click **Validate** again to clear the error message.

> **TIP** If you routinely validate very large decisions, increasing the number of threads available to the validation process can improve performance. For more information, see the sas.decisions.validation.validationCoreThreadPoolSize configuration option.

If the validation finds errors, you can click **Export** in the **Errors** panel to export the list of errors to a PDF file.

Change the Workflow Status of a Decision

1 (Optional) To change the status of a version other than the latest version, click the **Versions** tab, and then click the version number for the version that you want to change.

2 Click **Move to status**, and select the new status value. The Move to *status* window appears.

3 (Optional) Enter a comment. It is recommended that you enter a comment for governance purposes. For example, if you are the author and you are setting the status to Review-ready, describe the changes that you made and the reasons for making the changes.

4 Click **Submit**.

TIP The **Status** column on the **Versions** tab displays the status value for each version. You can click ⓘ to view or change the version's status and to edit the comments that are associated with the version's status. Only the user who changed a decision's status to its current setting can edit the comments for that setting.

Copy a Decision URL

To create a link for external documentation that automatically opens a decision in SAS Intelligent Decisioning:

1 Open the decision.

2 Click ⋮, and select **Copy decision URL**. The Copy URL window appears, and the URL is automatically selected.

3 Click **Copy**, and then click **Close**.

Paste the link into your documentation.

Compare Decision Content

You can compare the contents of two different decisions, or you can compare the contents of two different versions of the same decision.

1 Select the objects that you want to compare.

- To compare the contents of two different decisions, select the decisions in the category view, click ⋮, and select **Compare object contents**.
- To compare the contents of two versions of the same decision, open the decision, click ⋮ on the **Versions** tab, and select **Compare object contents**.

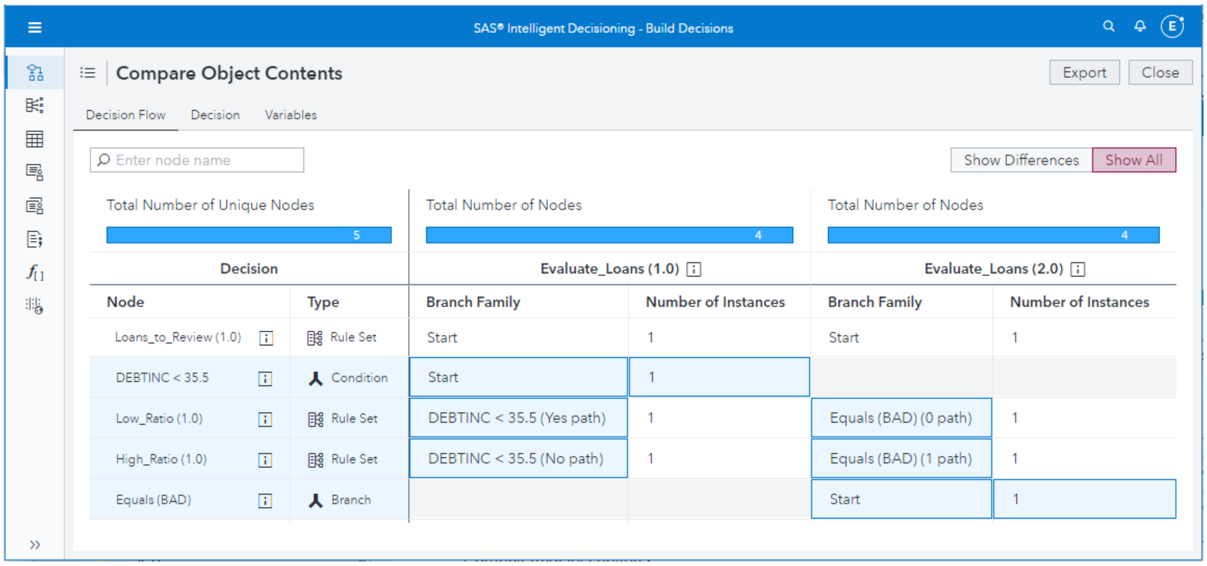
The Select Versions window appears.

2 Select the versions that you want to compare, and click **Compare**. SAS Intelligent Decisioning opens the two objects side-by-side in the Compare Object Contents window.

By default, the **Decision Flow** tab displays all of the nodes in each object and highlights the differences. Click **Show Differences** to limit the display to only the nodes that are different in each object. Click **Show All** to return to the default view.

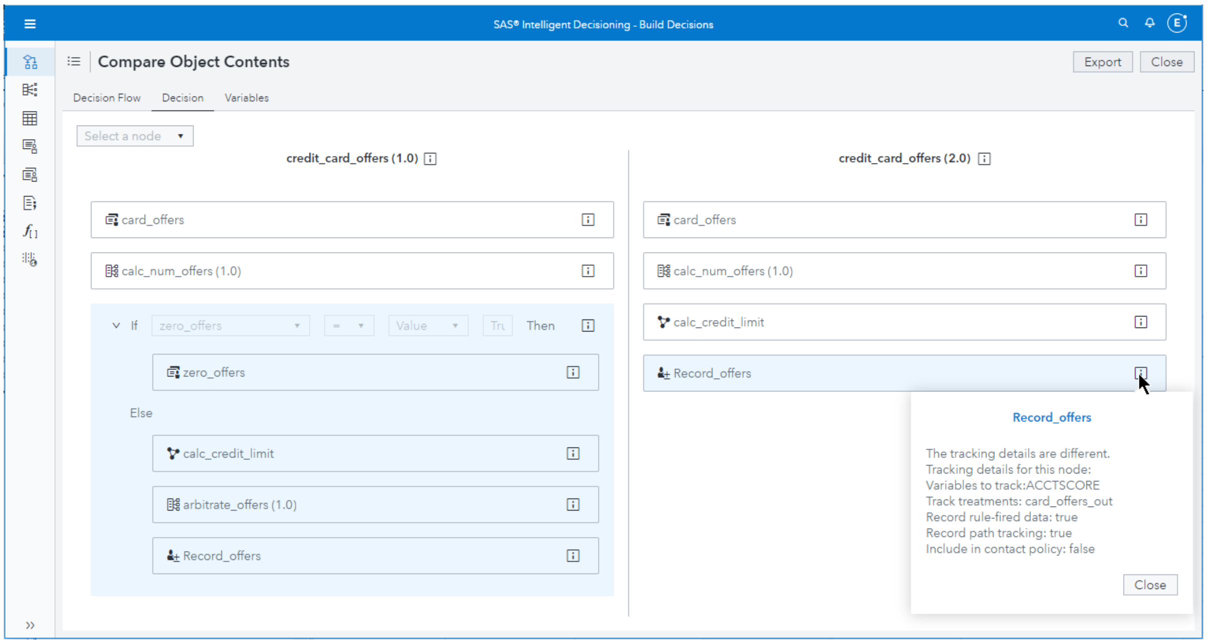
For example, in the following figure, both version 1.0 and version 2.0 of the Evaluate_Loans decision contain the same rule sets, but the branches are different. Version 1.0 uses a Yes/No branch that is based on the value of the DEBTINC variable, and Version 2.0 uses an Equals branch that is based on the value of the BAD variable. In Version 1.0, the Low_Ratio rule set is in the Yes branch path, and in Version 2.0, the Low_Ratio rule set in the BAD=0 path.

TIP Click ⓘ beside a decision name to display its location.



Decision		Evaluate_Loans (1.0)		Evaluate_Loans (2.0)	
Node	Type	Branch Family	Number of Instances	Branch Family	Number of Instances
Loans_to_Review (1.0)	Rule Set	Start	1	Start	1
DEBTINC < 35.5	Condition	Start	1		
Low_Ratio (1.0)	Rule Set	DEBTINC < 35.5 (Yes path)	1	Equals (BAD) (0 path)	1
High_Ratio (1.0)	Rule Set	DEBTINC < 35.5 (No path)	1	Equals (BAD) (1 path)	1
Equals (BAD)	Branch			Start	1

3 (Optional) Click the **Decision** tab to display the contents of each object. In this view, the information icon ⓘ on Record contacts nodes displays which variables and treatments are being tracked, whether rule-fired data or path-tracking are turned on, and whether the contact record is included in aggregate reports. For other node types, this icon displays information about whether a node was added, deleted, moved to another position in the decision, or changed to use a different version of the node object. However, if the decision is large or complex and a node was moved a significant distance, SAS Intelligent Decisioning might not be able to determine if the node was moved or deleted.



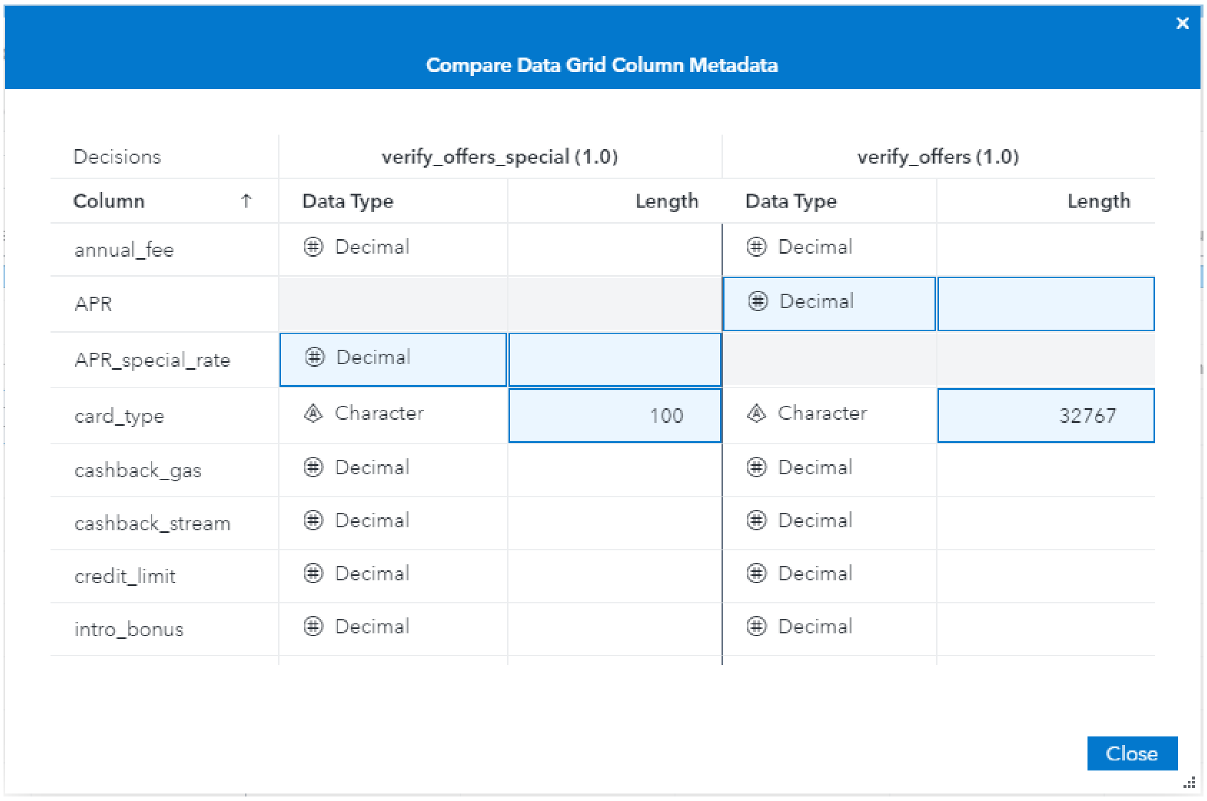
credit_card_offers (1.0)	credit_card_offers (2.0)
card_offers	card_offers
calc_num_offers (1.0)	calc_num_offers (1.0)
If zero_offers = Value Tru Then	calc_credit_limit
zero_offers	Record_offers
Else	
calc_credit_limit	
arbitrate_offers (1.0)	
Record_offers	

Record_offers

The tracking details are different.
Tracking details for this node:
Variables to track:ACCTSCORE
Track treatments: card_offers_out
Record rule-fired data: true
Record path tracking: true
Include in contact policy: false

4 (Optional) Click the **Variables** tab to display the properties of the variables in the two objects. This tab displays the name, type, initial value (if a value is defined), and length of each variable in both objects. The tab also indicates whether the variable is an input variable or an output variable. If the variable list for both objects is the same, this tab displays a message stating that the two objects are identical.

If either of the objects that you are comparing contain a data grid variable, you can click ⓘ beside the variable name to display the data type and length of each data grid column.



Compare Data Grid Column Metadata

Decisions	verify_offers_special (1.0)		verify_offers (1.0)	
Column ↑	Data Type	Length	Data Type	Length
annual_fee	Decimal		Decimal	
APR			Decimal	
APR_special_rate	Decimal			
card_type	Character	100	Character	32767
cashback_gas	Decimal		Decimal	
cashback_stream	Decimal		Decimal	
credit_limit	Decimal		Decimal	
intro_bonus	Decimal		Decimal	

5 (Optional) Click **Export** to export the results of the comparison to a PDF file. The Export Comparison Results window appears.

6 (Optional) Select the information that you want to export, and click **Export**. To export the information on a specific tab, select the tab name. To export the data type and length of each column in data grid variables, select **Data grid metadata**. If you want the PDF file to display only the differences between the two objects, select **Show differences**. To display all of the objects' information, select **Show all**.

   The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Compare Decision Code

You can compare the generated code of two different decisions, or you can compare the generated code of two different versions of the same decision.

1 Select the objects that you want to compare.

   - To compare the generated code of two different decisions, select the decisions in the category view, click ⋮, and select **Compare code**.
   - To compare the generated code of two versions of the same decision, open the decision, click ⋮ on the **Versions** tab, and select **Compare code**.

   The Select Versions window appears.

2 Select the versions that you want to compare, and click **Compare**. SAS Intelligent Decisioning opens the two objects side-by-side in the Compare Code window and highlights the differences.

3 (Optional) Click **Export** to export the results of the comparison to a PDF file. The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Generate PDF Documentation for a Decision

You can generate detailed documentation for a decision as a PDF document. The PDF includes the decision properties, the properties and variable mappings for each node, and a table of the decision variables. You can download documents with additional details about the rule sets, treatment groups, and nested decisions that are used in the decision.

1 Open the decision.

2 Click ⋮ , and select **Create document**. The Create Document window appears.

> **TIP** If this option is disabled, there might be unsaved changes. Click 🖫.

3 (Optional) Enter a name for the document if you do not want to use the default name.

4 (Optional) Select **Show a diagram of the decision** to include the decision diagram in the PDF.

5 (Optional) Select **Include code from code nodes** to include the code from custom code files that are used in the decision.

6 (Optional) Select **Choose additional documents to download** to display a window from which you can download additional documents. You can download documents for any rule sets, treatment groups, and nested decisions that are used in the decision.

7 Click **Create**. SAS Intelligent Decisioning creates the PDF. The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

  If you download additional documents in Step 4, the Download Additional Documents window appears.

8 (Optional) Click **Download** for each additional document that you want to download.

9 (Optional) Click **Close** to close the Download Additional Documents window.

Managing Decisions

Duplicating Decisions

About Duplicating Decisions in Folders for Which the Check-Out and Commit Feature Is Enabled

When you duplicate a decision, SAS Intelligent Decisioning first creates an empty decision, and then updates the decision to add references to the included objects. The default permissions for the Decision Repository enable users to create new objects, but the permissions do not include the ability to update the objects. In order to duplicate an object that is in the Decision Repository, you must check out the object.

If your site has created additional folders for which the check-out and commit feature is enabled, and if those folders have been assigned the same permissions as those assigned to the Decision Repository, you must also check out objects in those folders before you can duplicate them.

Duplicating Included Objects

When you duplicate a decision, you can choose whether to also duplicate the objects that are in the decision. If you choose not to duplicate the included objects, the duplicate decision uses the existing objects. If you choose to duplicate the included objects, the new copy of the decision uses the new copies of the included objects. If the decision includes a nested decision (subdecision), SAS Intelligent Decisioning duplicates the objects that are in the nested decision.

Note: Any models, lookup tables, custom context files, or Micro Analytic modules that are used in the decision are not duplicated. The duplicated decision uses the original objects. If the decision includes treatment groups, the treatment groups are duplicated, but the individual treatments within the treatment groups are not duplicated. The duplicated treatment group uses the original treatments.

When you duplicate a decision and its included objects, you must enter a custom suffix that SAS Intelligent Decisioning appends to the names of the duplicated objects. See “Duplicate Decisions” on page 261. All of the duplicated objects are placed in the same folder

Duplicate Decisions

Note: You cannot duplicate a decision if it is open.

To duplicate a single decision:

1 In the Decisions category view, select the decision that you want to duplicate.

2 Click ⋮ and select **Duplicate**. The Duplicate Decision window appears.

3 (Optional) Enter a new name for the duplicate decision. If you do not specify a new name or specify a suffix in Step 7b SAS Intelligent Decisioning appends `_Copy` to the name of the duplicate object.

4 (Optional) Enter a description for the decision.

5 Select the version of the decision that you want to duplicate.

6 Click 🗀 and select the location where you want to save the duplicate decision. If you select **Duplicate included objects** in the next step, the duplicated objects are also saved to the location that you choose.

7 (Optional) Click the **Duplicate included objects** toggle if you want to duplicate the objects that are included in the decision. Selecting this option enables the **View included objects** link and the **Name suffix** field.

   a (Optional) Click the **View included objects** link to open the **Included Objects** window. This window lists the objects that will be duplicated together with the decision. You can use the **Search** field to search for specific objects by name in order to verify which objects will be duplicated. Objects that will not be duplicated are identified with the ⚠ warning icon. See "Duplicating Included Objects" on page 260 for more information.

   Click **Cancel** to close the window.

   b In the **Name suffix** field, enter a custom string that you want SAS Intelligent Decisioning to append to the names of all of the duplicated objects.

8 Click **Duplicate**.

To duplicate multiple decisions:

1 In the Decisions view, select the decisions that you want to duplicate.

2 Click ⋮ and select **Duplicate**. SAS Intelligent Decisioning duplicates the latest version of the decisions and appends `_Copy` to the names of the duplicate copies. If needed, a number is also appended to the names of the duplicate copies.

Note: When you duplicate a decision, SAS Intelligent Decisioning creates a relationship between the original decision and the duplicate decision. If either object is changed, and you later copy the contents of one object into the other, SAS Intelligent Decisioning displays a warning message. Verify that you want to replace the current contents of the decision before you paste the new content.

Delete Decisions

Note: You cannot delete a decision if it is open.

In the Decisions category view, select the decisions that you want to delete, click ⋮, and select **Delete**. SAS Intelligent Decisioning moves the decisions to the recycle bin. For more information, see "Manage Folders and Folder Content" on page 9.

Rename a Decision

Note: You cannot rename a decision if it is open.

1. In the Decisions category view, select the decision that you want to rename.
2. Click ⋮ and select **Rename**. The Rename window appears.
3. Enter a new name for the decision, and click **Rename**.

Move Decisions to a Different Folder

1. In the Decisions category view, select the decisions that you want to move.
2. Click ⋮ and select **Move**. The Choose a Location window appears.
3. Select the location to which you want to move the decisions, and click **OK**.

Managing Versions of Decisions

Set the Displayed Version

The displayed version is the version whose information is displayed on the other tabs, such as the **Properties**, **Decision**, and **Variables** tabs. On the **Versions** tab, a ✓ in the **Displayed Version** column indicates the displayed version. To change the displayed version, click the version number for the version that you want to view. The displayed version is shown in the title bar.

Create a New Version

Note: For objects that are stored in locations for which the check-out and commit feature is enabled, you cannot manually create a new version. The only way to create a new version is to check out an existing version and commit a new version. For information, see "Check Out and Commit a Decision Version" on page 275.

Note: The current version of an object is the version with the highest version number. When you create a new version, SAS Intelligent Decisioning locks the current version before it creates the new version.

IMPORTANT You cannot unlock a locked version. You cannot save changes to a version that is locked. If you modify a version that is locked and click 🖫, SAS Intelligent Decisioning asks you if you want to replace the current unlocked version with your edited version.

To manually create a new version:

1 On the **Versions** tab, click the version number for the existing version that you want to use as the basis for the new version.

2 Click **New Version**. The New Version window appears.

3 Select the version type: **Minor** or **Major**. Version numbers follow the format *major.minor*. If you select **Major**, the number to the left of the period is incremented. If you select **Minor**, the number to the right of the period is incremented.

4 (Optional) For each version tag that you want to add, either enter a custom tag or begin typing and select a tag from the list of previously entered tags. Press Enter after each tag.

   Note: A tag is limited to 100 characters.

   TIP All tags that have been previously entered for any object are saved in the list that is displayed when you start typing. You cannot delete tags from this list.

5 (Optional) Enter information about the new version in the **Notes** field.

   TIP You can edit these notes at any time on the **Versions** tab.

6 Click **Save**.

Creating a New Version by Using Tags

About Using Tags to Create a Version

If the objects in a decision are tagged with version tags, you can create a new version of the decision by using these object tags to select which version of the included objects to use in the new decision version. This feature is useful when, for example, multiple users are developing the objects that are used within a decision. Each user can assign tags to object versions that they have developed for a new decision version. The person who is creating the new decision version can then use these tags to easily create the new version.

When you use version tags to create a new decision version, the versions of the included objects are ignored. Suppose your current decision uses version 1 of a rule set that has four versions. Versions 2 and 3 of the rule set are tagged with Tag_1, but versions 1 and 4 are not tagged. If you create a new decision version by specifying the tag Tag_1, you will be able to select either version 2 or 3 of the rule set for use in the new decision version. You cannot select version 4 of the rule set because it is not tagged with Tag_1.

Note: You cannot use tags to create new versions of common rule sets. To create a new version of a common rule set, open the rule set and select **New Version** on the **Versions** tab.

Create a New Version by Using Tags

TIP Before you begin, you need to know the version tags of the objects that you want to include in the new decision.

1 Open the decision for which you want to create a new version.

   TIP For decisions that are in folders for which the check-out feature has been enabled, your administrator must give you permission to create a new version by using tags. For more information, see "Grant Permission to Create Versions in Check-Out Folders by Using Tags" in *SAS Intelligent Decisioning: Administrator's Guide*.

2 On the **Versions** tab, click ⋮ and select **Create version from tags**. The Create Version From Tags wizard opens.

3 On the **Choose Strategy: Step 1** page of the wizard, enter up to two tags that are associated with object versions that you want to use to create the new decision version. Press Enter after each tag.

4 (Optional) If you want to select specific versions of each object to use in the new decision version, select **Let me choose the version of each object**.

5 (Optional) If you do not want to automatically map the object's variable to the decision's variables, deselect **Automatically map the object's variable to the decision's variables**.

   If you select this option, new object variables are mapped to decision variables only if decision variables of the same name and type already exist. If a decision variable does not already exist, you must manually map the object variable to a decision variable. For additional information, see "Mapping Variables within a Decision" on page 247.

6 Click **Next**.

   The node objects that have versions that are tagged with a tag that you specified are listed on the **Review and Confirm: Step 2** page of the wizard.

   Note: Common rule sets are not listed in the wizard. To create a new version of a common rule set, open the rule set and select **New Version** on the **Versions** tab.

7 (Optional) Select any node that you want to remove from the update process. Click 🗑, and then confirm the deletion. For any node that you remove from the update process, the existing node is used in the new decision version.

   TIP You can filter the list of nodes by using the **Search** field or by clicking to filter by more specific criteria in the Filter Nodes window.

8 (Optional) If you selected **Let me choose the version of each object** in Step 4 and any node object has multiple versions that are tagged with the same tag, select the version of those objects that you want to use in the new decision version. To use the latest version of all of the objects, click **Use latest version**.

9 Click **Next**.

10 On the **Create Version: Step 3** page of the wizard, do the following:

   a Select the version type: **Minor** or **Major**. Version numbers follow the format *major.minor*. If you select **Major**, the number to the left of the period increments. If you select **Minor**, the number to the right of the period increments.

   b (Optional) Enter any version tags and notes for the new version of the decision.

11 Click **Create**.

The new version of the decision is listed on the **Versions** tab for the decision.

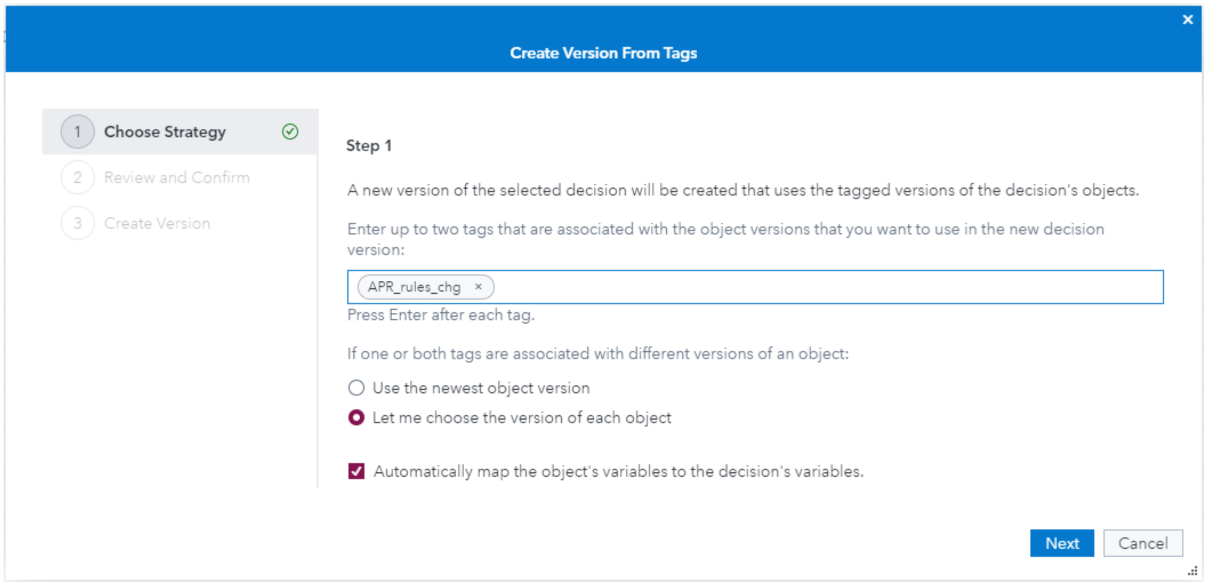
Example

Suppose you have the decision flow Update_credit_history. Version 2 of this decision contains the following objects:

- Version 2 of the rule set Select_accounts, which is not tagged. Version 1 of this rule set is tagged with APR_rules_chg, but this tag was added to the wrong version of this rule set. The tag should have been added to version 2 of the rule set.
- Version 3 of the decision Calc_credit_chgs. Both version 3 and version 4 of this decision are tagged with APR_rules_chg.
- Version 1 of the code file Update_history. This version is the only version of this code file and it is not tagged.

You need to create a new version of Update_credit_history that includes version 4 of the decision Calc_credit_chgs (the tagged version) and version 2 of the rule set Select_accounts (which is not tagged).

1 On page 1 of the Create New Version wizard, specify the tag APR_rules_change and select the option **Let me choose the version of each object**.



Step 1

A new version of the selected decision will be created that uses the tagged versions of the decision's objects.

Enter up to two tags that are associated with the object versions that you want to use in the new decision version:

APR_rules_chg

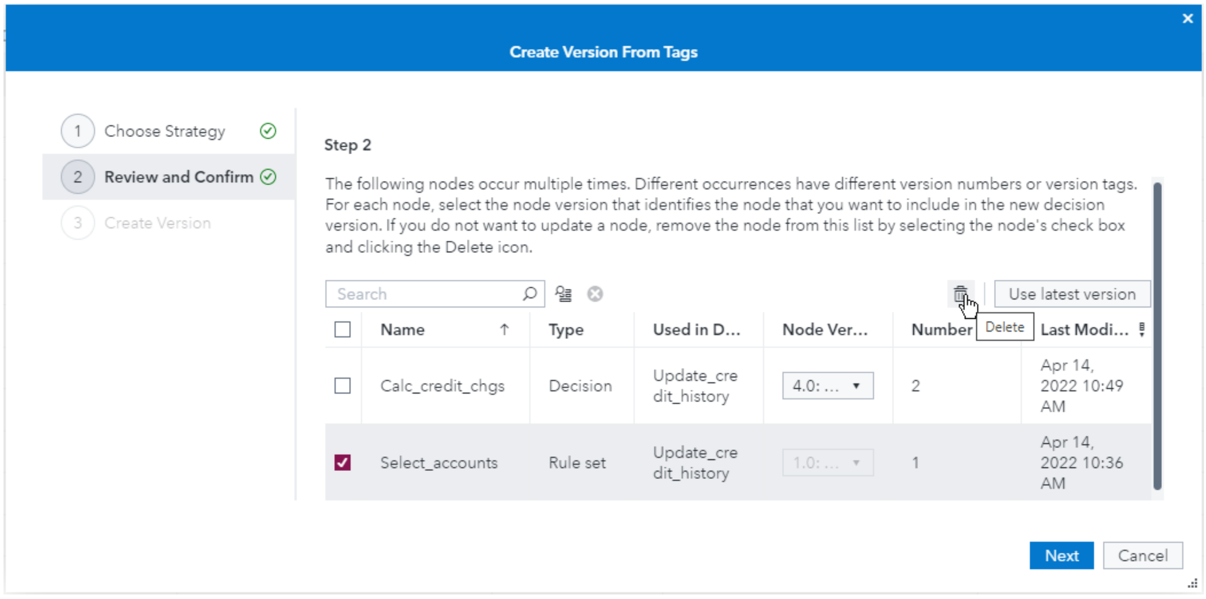
Press Enter after each tag.

If one or both tags are associated with different versions of an object:

- Use the newest object version
- [✓ Let me choose the version of each object]

[✓] Automatically map the object's variables to the decision's variables.

Page 2 of the wizard contains a list of the objects that are tagged with APR_rules_chg.



Create Version From Tags

1 Choose Strategy
2 Review and Confirm
3 Create Version

Step 2

The following nodes occur multiple times. Different occurrences have different version numbers or version tags. For each node, select the node version that identifies the node that you want to include in the new decision version. If you do not want to update a node, remove the node from this list by selecting the node's check box and clicking the Delete icon.

	Name	Type	Used in D...	Node Ver...	Number	Last Modi...
☐	Calc_credit_chgs	Decision	Update_credit_history	4.0: ...	2	Apr 14, 2022 10:49 AM
☑	Select_accounts	Rule set	Update_credit_history	1.0: ...	1	Apr 14, 2022 10:36 AM

If you do not make any changes in this list, then the new version of the decision Update_credit_history will include the objects that are shown in the list. If multiple versions of an object are tagged with the same tag, the wizard includes each version in the Node Version and Tag column for that object. In this example, both versions 3 and 4 of the subdecision Calc_credit_chgs are tagged with APR_rules_chg. By default, the latest version in this list is selected, but you can select a different version in the drop-down list. The new version of the decision Update_credit_history will use the version that you select.

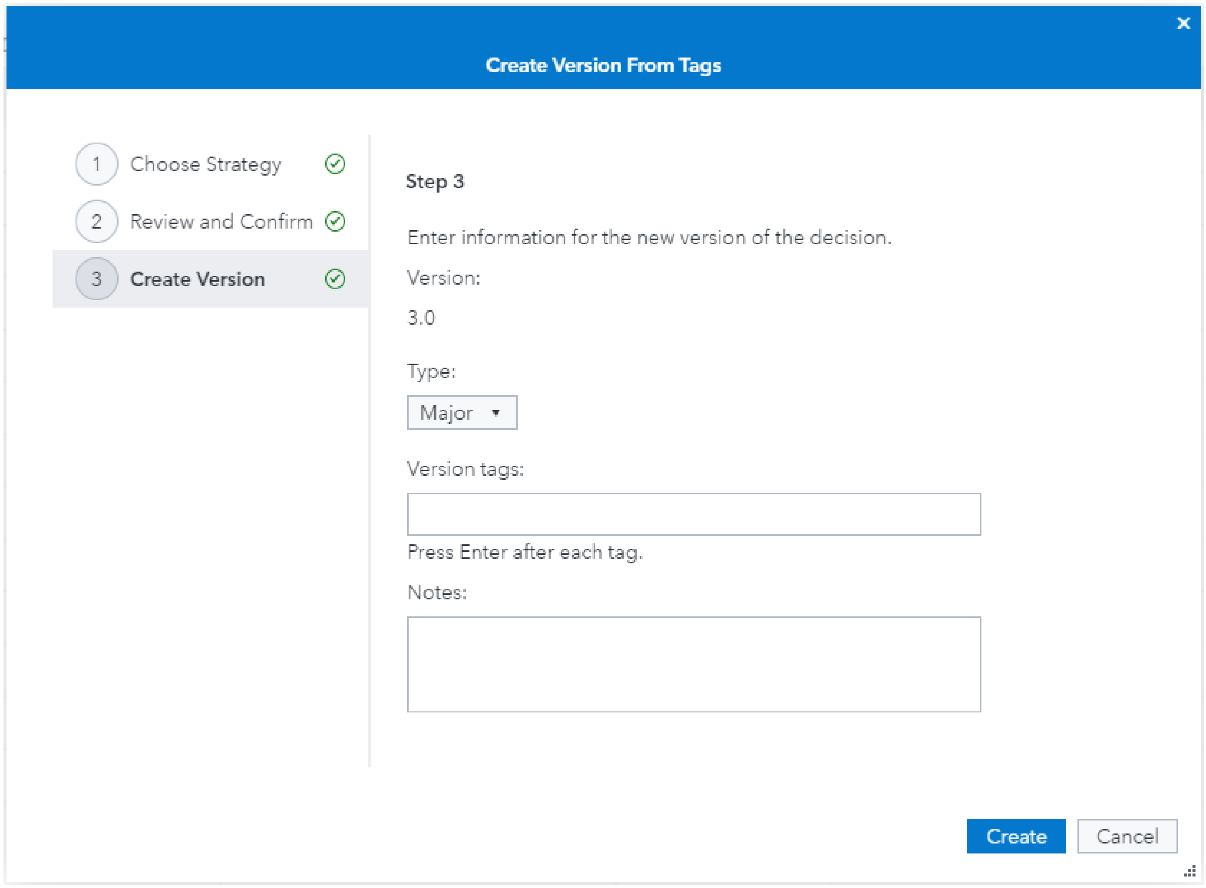
For the Select_accounts rule set, the wrong version (version 1) of the rule set was tagged. Remove the Select_accounts rule set from the list of objects that will be updated in the new version of the decision Update_credit_history.

2 Select the Select_accounts rule set. Click 🗑, and then confirm the deletion.

When you remove an object from the list, the original object that is used in the decision is still used in the new version of the decision.

3 Click **Next**.

4 On page 3 of the wizard, select **Major** for the version type, and then click **Create**.


Create Version From Tags

1 Choose Strategy ✓
2 Review and Confirm ✓
3 Create Version ✓

Step 3

Enter information for the new version of the decision.

Version:

3.0

Type:

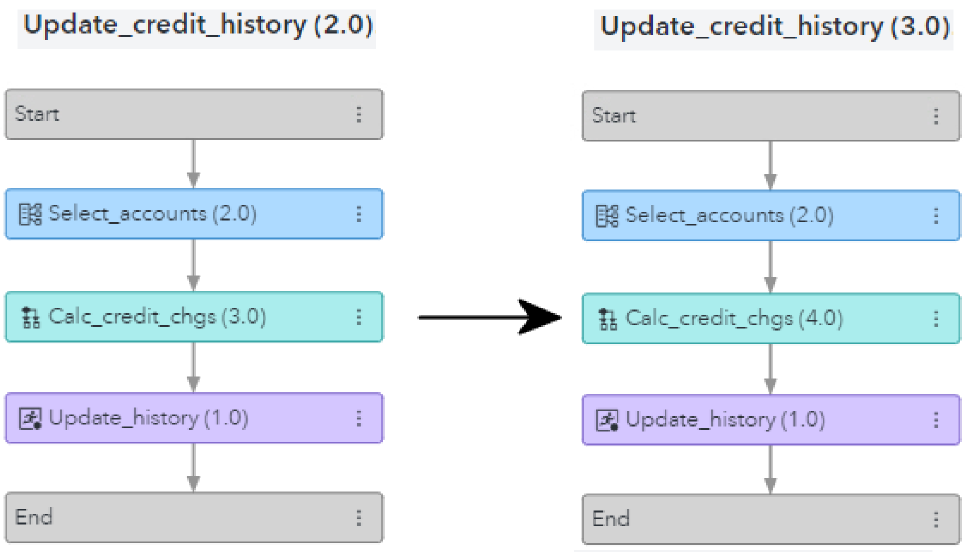
Major

Version tags:

Press Enter after each tag.

Notes:

SAS Intelligent Decisioning creates version 3 of the decision.



Create a New Version Automatically When You Publish a Decision

You can automatically create a new version of a decision when you publish the current version by selecting **Publish and Lock**.For information, see “Publish Decisions” on page 281.

Copy the Content of a Version

You can copy the content of an object’s version in the category view or on the **Version** tab for the object.

1 In the category view, complete these steps:

   a Select the decision whose contents you want to copy.

   b Click ⋮ , and select **Copy version**. The Copy Version window appears.

   c Select the version whose contents you want to copy.

   Alternatively, on the **Versions** tab of the decision whose contents you want to copy:

   a Select the version whose contents you want to copy.

   b Click ⋮ , and select **Copy version**. The Copy Version window appears.

2 Click 🗀, and select the decision into which you want to paste the contents of the version.

   When you paste the contents, SAS Intelligent Decisioning creates a new version of the target decision. The target object contains only the pasted content.

3 Select whether you want to create a new major or minor version.

4 (Optional) Modify the notes that will be associated with new version.

5 (Optional) Add tags that will be associated with the new version. Tags that are associated with a source object version are not automatically added to the new version. See “Add a Version Tag” on page 277

6 Click **Paste Version**, and then click **Yes**.

TIP The input and output designations for a variable on the **Variables** tab for the new version are removed, and the variable is treated as a temporary variable in the following situations:

- The new version of the target object does not use a variable that is used in an earlier version.
- The source and target object originally had a variable of the same name, and you deleted the variable from the source object before you copied it into the target object.

Note: When you copy the contents of a source object into a target object, SAS Intelligent Decisioning creates a relationship between the two objects. If the source object is modified after you copy its contents, and you later copy the contents of the target object back into the source object, SAS Intelligent Decisioning displays a warning message. Verify that you want to replace the current contents of the source object before you paste the new content.

Delete a Version

IMPORTANT When you delete a specific version, that version is deleted permanently. It is not moved into the recycle bin, and it cannot be restored.

Note: In order to be able to delete a specific version of an object, you must have permission to delete the object itself. Also, the configuration option sas.decisions.deleteVersions must be turned on.

On the **Versions** tab, select the version that you want to delete, click ⋮ , and select **Delete**.

You cannot delete the current version.

Upgrade Decisions to Use a New Version of a Nested Decision

If you create a new version of a nested decision that is already used in other decisions, you can upgrade the decisions to use the new version.

1 On the **Versions** tab for the nested decision, click ⋮ , and select **Upgrade decisions**. The Upgrade Decisions window appears. This window lists all of the decisions that include the nested decision.

2 In the **Version to upgrade to** field, select the version of the nested decision to which you want to upgrade the decisions.

3 Select **Automatically map variables** if you want SAS Intelligent Decisioning to automatically map new nested decision variables in the decisions where the nested decision is used.

For information about how SAS Intelligent Decisioning maps variables, see "About Decision Variables and Mapping" on page 247.

4 Select the check boxes for the decisions that you want to upgrade, and click **Upgrade Decisions**.

TIP To update all of the objects that are used in a decision, see "Update Decisions to Use New Object Versions" on page 253.

Determine Which Objects Use a Decision

To list the objects that use a specific decision:

1 On the **Decisions** category page, select the check box for the decision, click ⋮, and select **View used by report**. The All Objects that Use the Selected Item window appears. This window lists all objects that use any version of the selected decision.

2 (Optional) Select a specific version of the decision. SAS Intelligent Decisioning narrows the list to include only the objects that use the selected version of the decision.

Note: The **View used by report** option is also available from within an open decision.

In the report, you can use the **Filter** field to filter the list of objects based on the object names. Click on an object name to open the object. Click ⓘ next to an object name to display the date on which the object was last modified and the ID of the user who modified the object. For decisions and code files that have been checked out, SAS Intelligent Decisioning also displays the IDs of the users that have checked out the objects. For decisions, SAS Intelligent Decisioning includes the decision's workflow status.

Click **Export** to export the Used By report to a PDF file. In the PDF file, you can click on an object name to open the object in a new browser tab.

Checking Out and Committing Decision Versions

About Checking Out and Committing Versions

Your administrator can enable the check-out and commit feature for decisions that are in any folder by specifying the folder in the sas.decisions.checkout.checkoutEnabledFolderPaths configuration option. Enabling this feature for a folder does not automatically modify the permissions for the folder or for the objects in it. You can still modify a decision in the folder without checking it out, but you are expected to check out the latest version before you edit it. However, your administrator might also set permissions that require you to check out decisions in these folders before you can edit them. For more information, see "sas.decisions.checkout" in *SAS Intelligent Decisioning: Administrator's Guide* and "Set Permissions for Check-Out Folders" in *SAS Intelligent Decisioning: Administrator's Guide*.

By default, SAS Intelligent Decisioning defines a folder where you can store decisions that must be checked out before they can be edited by users without administrator privileges. This folder is the `Decision Repository` folder, and it is the default value for the sas.decisions.checkout.checkoutEnabledFolderPaths configuration option. The default permissions for this folder require that non-administrative users check out a version and commit their changes to the checked-out version. Users who do not have administrative permissions cannot edit the decisions in `Decision Repository` without first checking them out.

Note: The sas.decisions.checkoutEnabledFolderPaths option controls the check-out and commit feature for both decisions and code files. The check out and commit feature for lookup tables, treatments and treatment groups, and rule sets are controlled by configuration options that are specific to those object types.

If a version can be or must be checked out before it is modified, the **Check Out** button appears at the top of the **Versions** tab for that object. You can check out any version of an object. You can check out only one copy of a version at a time.

TIP If the sas.decisions.checkout.allowConcurrentCheckout option is turned off, and a user has checked out a decision version, the **Check Out** button for that decision is disabled for all other users. For more information, see "Concurrently Checking Out and Committing Decision Versions" on page 274.

When you check out a version, SAS Intelligent Decisioning writes a working copy of the version into your `My Folder` folder and opens the working copy. SAS Intelligent Decisioning adds "`(Checked Out)`" to the name that is displayed at the top of the window.

While you have a version checked out, the Decisions category view shows two decisions with the same name, but the folders listed in the **Location** column differ for each decision. The original version is in the location specified by the sas.decisions.checkout.checkoutEnabledFolderPaths configuration option, and the checked-out copy is in your `My Folder` folder.

TIP If an object that you have checked out does not appear in the category view, click ↺ to refresh the category view.

Note: When a decision that uses a treatment group is deployed to a production environment, the decision always uses the active version of a treatment group. When you edit a decision that uses a treatment group that you have checked out, the properties panel for the treatment group shows the location of the treatment group in the original folder. The panel does not list the location of the treatment group that is checked out because that copy is not the active version.

A **Commit** button appears at the top of the **Versions** tab for the checked-out version. When you are finished editing the checked-out version, you must commit your changes in order for other users to be able to see them. When you commit your changes, SAS Intelligent Decisioning creates a new version with your changes. If you check out objects that are used in the decision and modify the checked-out decision to use the checked-out objects, SAS Intelligent Decisioning also commits the checked-out objects when you commit the decision. If the decision uses a treatment group that is checked out, and if the treatments are also checked out, then the treatment group and the treatments are all committed when you commit the decision.

If the parent object is deleted before you commit your changes, you will not be able to commit your changes.

You cannot publish the checked-out version that is in `My Folder`. To publish a version with your changes, you must commit your changes, and publish the committed version.

Checking Out and Committing Included Objects Together with A Decision

When you check out a decision version, SAS Intelligent Decisioning displays the list of object versions that are used in the decision in the Choose Objects to Check Out window. You can select any object that you want to check out at the same time. For any object that you choose to check out, SAS Intelligent Decisioning writes a working copy of the version into your `My Folder` folder, and updates the checked-out decision so that it uses the checked-out versions of each object that you select. When you commit the decision, SAS Intelligent Decisioning also commits the objects. If the decision includes a treatment group that is checked out, and if the treatment group includes treatments that are checked out, then both the treatment group and its checked-out treatments are committed when you commit the decision. However, SAS Intelligent Decisioning does not automatically activate the newly committed version of the treatment group. See “Activate a Treatment Group” on page 94 for instructions for activating a treatment group.

If you select an object that is used only in a nested decision, ensure that you also select the nested decision. If you did not check out the nested decision, the object that is used in the nested decision is not committed when you commit your work.

Your decision might use an older version of an object, or it might use different versions of the same object. The list of included objects has an entry for each object version that is used in the decision. If you select two versions of the same object, SAS Intelligent Decisioning checks out the oldest version. SAS Intelligent Decisioning checks out only one version of each selected object. If a selected object is an older version of the object, you must select the **Update the decision to use the latest version of the selected objects** check box in the Choose Objects to Check Out window in order to successfully check out the selected objects. When you select this option, SAS Intelligent Decisioning asks you to confirm that you want to replace references to the selected versions of the objects with references to the latest version. References to any version of any objects that you do not select in the Choose Objects to Check Out window remain unchanged when you commit the decision.

Checking Out and Committing Objects from within A Decision

If a decision is checked out, and if an object that is included in the decision is stored in a folder for which the check-out feature is enabled, then you can check out the object from within the decision. When you check out an included object, the decision is modified to use the checked-out copy of the object and is saved. When you commit the decision, all checked-out objects that are used in the decision are also committed.

If the object is a nested decision, SAS Intelligent Decisioning does not display the list of objects that are included in the nested decision. You cannot check out a nested decision's objects from within a decision. After you check out the nested decision, you can open the nested decision and check out its included objects from within the nested decision itself.

To check out an object from within a checked-out decision, right-click on the node for the object that you want to check out, and select **Check out**.

To commit a single object from within a checked-out decision, right-click on the node for the object, and select **Commit**. SAS Intelligent Decisioning displays the Commit dialog for the object. For more information, see the topics about checking out and committing a version of a specific object type.

Concurrently Checking Out and Committing Decision Versions

The ability for multiple users to check out the same decision at the same time is controlled by the sas.decisions.checkout.allowConcurrentCheckout configuration option. This option is turned on by default.

When this option is turned on, different users can check out the same version of the same object at the same time. Because the objects that are checked out are saved in each user’s `My Folder` location, the default permissions allow individual users to see only the copies that they have checked out.

When this option is turned off and a user has checked out an object, the **Check Out** button for that object is disabled for all other users.

If multiple users check out the same version of the same object at the same time, each user’s changes are preserved in a new version when they commit their changes. One user’s changes do not overwrite another user’s changes.

IMPORTANT If two users attempt to commit changes to the same object simultaneously, the first user's attempt will succeed but the second user might see an error message that the commit has failed. If the second user subsequently commits their changes, the **Modified By** column on the **Versions** tab for both the version committed by the first user and the version committed by the second user displays the user ID of the second user.

Check Out and Commit a Decision Version

1 On the **Versions** tab, click **Check Out**. The Choose Objects to Check Out window appears. This window displays the list of object versions that are used in the decision.

2 (Optional) Select the objects that you want to check out, and click **+›**. See “Checking Out and Committing Included Objects Together with A Decision” on page 273 for more information.

Note: If you select an object that is used only in a nested decision, ensure that you select the nested decision also. If you do not check out the nested decision, the object that is used in the nested decision is not committed when you commit your work.

3 (Optional) Select the check box **Update the decision to use the latest version of the selected objects** to confirm that you want to replace references to the selected versions of the objects with references to the latest version.

Note: If your decision uses an older version of an object or different versions of the same object, you must select this check box in order to check out the object. See “Checking Out and Committing Included Objects Together with A Decision” on page 273 for more information.

4 Click **Check Out** in the Choose Objects to Check Out window to check out the decision and any additional objects that you selected.

SAS Intelligent Decisioning updates the **Properties** tab to indicate that the version is checked out.

5 Modify the checked-out version as needed, and save it.

TIP To discard the changes and delete the checked-out version from `My Folder`, you can commit the object without saving it first. However, committing the object without saving creates a new version of the object whose contents match the contents of the previous version. For information on undoing a check out, see "Undoing a Check Out" on page 277.

6 On the **Versions** tab, click **Commit**. The Commit Decision Version window appears.

7 Select the version type: **Minor** or **Major**. Version numbers follow the format *major.minor*. If you select **Major**, the number to the left of the period is incremented, and the minor number is reset to zero. If you select **Minor**, the number to the right of the period is incremented.

8 (Optional) In the **Version tags** field, enter any version tags that you want to associate with the new version. Press Enter after each tag. Tags are limited to 100 characters each and cannot contain the following characters: $ ' " # & ? ( ) \ /

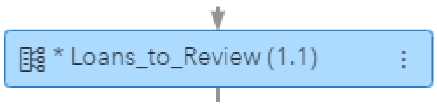
9 (Optional) To add the tags in the **Version tags** field to all of the objects that you are committing, select the **Apply tags to all committed components** check box.

10 (Optional) Enter information about the new version in the **Notes** field.

11 Click **Commit**. SAS Intelligent Decisioning creates a new version with your changes, and deletes the working copy from your **`My Folder`** folder. If you have checked out any objects that are used in the decision, and if the decision is updated to use the checked-out objects, SAS Intelligent Decisioning also commits the additional objects.

Determine Whether a Version Is Checked Out

If an object in a decision is checked out, an asterisk appears before the object name in the diagram on the **Decision Flow** tab.



Also, the **Properties** panel for each object in the decision specifies whether the object is checked out.

Determine Who Has a Version Checked Out

If the current version of an object is checked out, the IDs of the users that checked it out and the timestamps when each user checked it out appear in the **Checked out by** field on the **Properties** tab for the original object. You can also display this information by clicking ↗ beside the version number on the **Versions** tab.

Opening the Original Object

When you check out an object, SAS Intelligent Decisioning adds the field **Original object link** to the **Properties** tab for the checked-out object. This field contains a link to the original object that was checked out. You can use this link to verify that you have checked out the correct version and to compare the original content with the modified content in the checked-out version.

Undoing a Check Out

If both an object and a decision that uses the object are checked out at the same time, or if you checked out the object from within the decision, click ⋮ on the object's node in the decision diagram, and select **Cancel checkout**.

You can discard a checked-out version and any changes that you made by deleting the working copy of the version from your `My Folder` folder if the following conditions are true:

- You have not checked out a decision that uses the object.
- The object was not checked out at the same time as a decision that uses the object, or the object was not checked out from within the decision after the decision was checked out.

The deleted version is moved to the recycle bin. See “Delete Decisions” on page 262.

Managing Version Tags for Decisions

Add a Version Tag

Version tags enable you to better organize and group your content. Version tags are associated with specific versions of an object and not with the entire object. You can add the same tag to any version of any object. To add a tag to a decision version:

1. On the **Versions** tab, position your cursor in the **Version Tags** column for the version that you want to tag. If the version is not tagged, + appears. If the version has at least one tag, ✎ appears.
2. Click + to open the New Version Tags window, or click ✎ to open the Edit Version Tags window.
3. For each tag that you want to add, either enter a custom tag or begin typing and select a tag from the list of previously entered tags. Press Enter after each tag.

Tags are limited to 100 characters each and cannot contain the following characters: $ ' " # & ? ( ) \ /

TIP All tags that have been previously entered for any object are saved in the list that is displayed when you start typing. You cannot delete tags from this list.

TIP To filter the version list based on a tag, right-click on the **Version Tags** column heading, enter a filter string in the **Filter** box, and press Enter. To clear the filter, delete the string from the **Filter** box, and press Enter.

4 Click **Close** to close the window.

Remove a Version Tag

1 On the **Versions** tab, position your cursor in the **Version Tags** column for the version whose tag you want to remove, and click ✎.

2 Click × beside the tag that you want to remove.

3 Click **OK** to close the window. The tag remains in the list of previously entered tags that is displayed when you add a tag, but the tag is no longer associated with version.

Modify a Version Tag

You cannot modify a version tag that already exists. To change the content of an existing tag, delete the tag as described in "Remove a Version Tag" on page 278, and then add the tag again as described in "Add a Version Tag" on page 277.

Publishing a Decision

Introduction to Publishing

Publishing content makes it available to other applications. Publishing a decision creates an entity that can be managed and run in another environment. When you publish content, SAS Intelligent Decisioning generates code for that content and writes it to the destination. The following table describes what form the generated code takes for each destination type.

Table 8.1 *What SAS Intelligent Decisioning Publishes to Each Destination Type*

Destination Type	**What** SAS Intelligent Decisioning **Does**
SAS Cloud Analytic Services (CAS), Teradata, or Apache Hadoop	Adds a row to the model table for the destination
SAS Micro Analytic Service	Writes a Micro Analytic Service module in the service
Git	Creates a directory in the remote Git repository with the same name as the published object and writes the generated code to a file named `scoreResource.txt`. If the decision includes analytic store models, the model files are written as either `.sasast` files or `.astore` files, depending on how the model was built.
Container destinations	Creates a *SAS Container Runtime* container and writes it to the destination. **Note:** The container destination types to which you can publish content from SAS Intelligent Decisioning are Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), and private Docker repositories.

The rows in the model tables, the Micro Analytic Service modules, and the SAS Container Runtime containers all become callable REST API endpoints, independent of SAS Intelligent Decisioning. For content that is published to Git destinations, you can use the SAS Intelligent Decisioning Git Deployment CLI to deploy content from the remote repository either to a CAS destination or to a SAS Micro Analytic Service destination. The deployed content then becomes a callable REST API endpoint.

Where Can I Publish Decisions?

The destinations to which you can publish a decision depend on the destinations that are available and on the objects that are used in the decision. Your system administrator determines which publishing destinations are available. The following table shows the destination types to which you can publish a decision that contains various object types.

Table 8.2 Decision Objects and the Destinations to Which Decisions Can Be Published

Objects in the Decision	SAS Micro Analytic Service and Compatible Git Destinations	SAS Cloud Analytic Services (CAS) and Compatible Git Destinations	Teradata and Hadoop Destinations	Container Destinations
Assignment rule sets, treatment groups, models, nested decisions, or DS2 code files and custom context files that do not contain SQL queries[1, 2]	yes	yes	yes	yes
Filtering rule set nodes[1]	yes	yes	yes	yes
Python code files	yes	yes	no	yes
Data query files, DS2 code files that contain SQL queries, or custom context files that contain SQL queries	yes	no	no	yes[3]
Record contacts nodes	yes	yes	yes	no
Micro Analytic Module nodes	yes	no	no	no

1 Rule sets and nested decisions can use lookup tables, custom functions, and global variables.

2 The score code type of a model determines the destination types to which you can publish a decision that contains the model. See Table 8.3 on page 280.

3 In container destinations, Oracle is the only database that can be queried by using data query files, DS2 code files that contain SQL queries, or custom context files that contain SQL queries.

The score code type of a model determines the destination types to which you can publish a decision that contains those models. The following table lists the score code types a model can have when it is used in a decision that is published to a specific destination type.

Table 8.3 Model Score Code Types and the Destinations to Which Decisions Can Be Published

Model Score Code Type	CAS Destinations	Git Destinations	SAS Micro Analytic Service Destinations	Hadoop and Teradata Destinations	Container Destinations
DATA step	yes	yes	yes	yes	yes
DS2 multi-type	yes	yes	yes	yes	yes
DS2 package	no	yes	yes	no	yes

Model Score Code Type	CAS Destinations	Git Destinations	SAS Micro Analytic Service Destinations	Hadoop and Teradata Destinations	Container Destinations
Python	yes	yes	yes	no	yes[1]

1 You can publish decisions that include a model node for a Python model that is stored in the common model repository. The packages that are used by the model must be specified in the requirements.json file. For more information, see "Scoring Python Models" in *SAS Model Manager: User's Guide*.

SAS Model Manager supports publishing some additional model types independently (not in a decision). For additional information, see "Model Score Code Types and Publishing Destinations" in *SAS Model Manager: User's Guide*.

For more information about publishing destinations, see "Configuring Publishing Destinations" in *SAS Intelligent Decisioning: Administrator's Guide*.

Publishing Decisions That Include Analytic Store Models

If you are publishing a decision that includes an analytic store model, the model's analytic store (ASTORE) file must be in the `/opt/sas/viya/config/data/modelsvr/astore` directory on the CAS server for the destination to which you are publishing the decision. The ASTORE file is copied to that location when you do any of the following:

- run a decision test for the decision that uses the analytic store model
- set the model as a project champion in SAS Model Manager
- publish the model to SAS Micro Analytic Service from SAS Model Manager

If you are publishing a decision that includes an analytic store model and the model has not been set as a project champion or published from SAS Model Manager, you must test the decision before you publish it. For more information, see "Testing Decisions" on page 284.

Publish Decisions

Note: You must validate decisions and correct any errors that are found by the validation process before you publish the decisions. For information, see "Validate a Decision in the Decision Editor" on page 254.

1 Open the decision, and click the **Versions** tab.

   By default, the version that is published when you click **Publish** is the displayed version.

2 (Optional) Click the version number to change the displayed version to the version that you want to publish.

3 (Optional) Enter any tags that you want to associate with the published object. Press Enter after each tag.

4 Either click **Publish** or select **Publish and Lock**.

- To publish the selected version without also creating a new minor version, click **Publish**.
- To lock and publish the selected version and create a new minor version at the same time, click ⋮ , and select **Publish and Lock**.

The Publish Decisions window appears.

5 Select the destination to which you want to publish. The publishing destinations that are available to you depend on what is configured at your site. See *SAS Viya Platform: Publishing Destinations* for more information.

6 (Optional) In the **ITEMS TO PUBLISH** section, complete the following steps for each item that you are publishing:

a Edit the **Published name** if you do not want to use the default name for the published module. The maximum length and character restrictions differ depending on your destination. See Table 8.4.

Table 8.4 Requirements and Restrictions for Published Names

Destination	Maximum Length	Requirements And Restrictions
Container destinations	127	The published name must start with a letter or an underscore. It cannot contain spaces, multi-byte characters, or special characters other than the underscore. The name that you enter is assigned to the SAS_SCR_APP_PATH environment variable. The value of this variable determines the module name. For more information, see "Changing the Endpoint Name for a Container" in *SAS Container Runtime: Programming and Administration Guide*.
Git	128	The published name cannot contain forward slashes (/), single quotation marks ('), or double quotation marks (").
SAS Micro Analytic Service	100	The published name cannot contain the following characters: ! @ # $ % ^ & * ( ) | = ~` \ / . { } " ' ;
SAS Cloud Analytic Services (CAS)	128	The published name cannot contain single or double quotation marks.
Teradata	128	The published name must start with a letter or an underscore. It cannot contain spaces, multi-byte

Destination	Maximum Length	Requirements And Restrictions
		characters, or special characters other than the underscore.
Apache Hadoop	128	The published name cannot contain colons (:) or double quotation marks.

b If you have previously published the decision, turn on the **Replace item with the same name** option in order to replace the previously published decision of the same name in the same destination.

c Select the **Path tracking** check box if you want the published decision to generate path tracking data.

> **TIP** This path-tracking data is recorded in the pathID column in the output table. The path-tracking data that is recorded when you select **Record path tracking** for a record contacts node is recorded in the subject contact history.
>
> This option is disabled if you are publishing content to a SAS Micro Analytic Service destination, a Git destination, or a container destination.

d Select the **Rule-fired tracking** check box if you want the published decision to generate rule-fired data.

> **TIP** This rule-fired data is recorded in the ruleFiredFlags column in the output table. The rule-fired data that is recorded when you select **Record rule-fired data** for a record contacts node is recorded in the subject contact history.
>
> This option is disabled if you are publishing content to a SAS Micro Analytic Service destination, a Git destination, or a container destination.

7 Publish the decisions.

To publish content to a SAS Cloud Analytic Services (CAS) destination, you must reload the CAS destination table in order to make the newly published items available to other applications. You do not need to reload the destination table when you publish content to other destination types.

Select one of the following options:

Publish
publishes the decisions and, if you are publishing content to a CAS destination, automatically reloads the CAS destination table. If another user is executing the code for an item that was previously published to CAS while the destination table is being reloaded, reloading the table might cause temporary problems with accessing the table content. After the table is reloaded, all authorized users can access all items in the table.

Publish without reloading
publishes the decisions but does not reload the CAS destination table. You must manually reload the table in order for the newly published items to be accessible.

Publish and lock
locks and publishes the decision and creates a new minor version, but does not reload the CAS destination table. You must manually reload the table in order for the newly published items to be accessible.

Note: This option is not available unless you chose **Publish and Lock** in Step 4.

The Publishing Results window appears. It displays the names of the published items, their status, and a link to the log that was generated during the publishing process.

8 After the status changes to **Published successfully**, click **Close** to close the Publishing Results window.

TIP To view the publishing history for a decision, click the **History** tab.

9 (Optional) Click **Close** to close the decision.

Testing Decisions

Ways to Test a Decision

There are three types of tests:

Basic test
executes the decision in the SAS Cloud Analytic Services (CAS) destination using the input table that you specify. You can also specify a debugging variable and enable value tracing. For more information, see “Create and Run a New Test” on page 285.

Scenario test
enables you to enter specific input values and the output values that you expect the test to generate. A scenario test identifies differences between the output that you expect to see and the actual output that is generated when the test is run. You can also compare the test definitions and test results of different scenarios. Scenario tests are also run in CAS. For more information, see “Test a Scenario” on page 289.

Publishing validation test
executes the decision in a publishing destination using the input table that you specify. When you publish the decision, a validation test is automatically defined

for that decision in that destination. For more information, see "Validate a Published Decision" on page 303.

Test a Decision

Create and Run a New Test

Testing a decision is optional, but doing so is a best practice. Testing enables you to discover any problems before the decision is published and incorporated into a production system.

IMPORTANT If you are testing a decision that uses functions that are defined in a custom context file, verify that the context file is specified in the **Custom context** field on the **Properties** before running the test.

If you are testing a decision that uses treatment groups and both the decision and the treatment group are checked out, the test uses the checked-out version of the treatment group. If the treatment group is not checked out, the test uses the activated version of the treatment group.

If the decision uses a rule set that uses a lookup table and all three objects (the decision, the treatment group, and the lookup table) are checked out, the test uses the checked-out versions of both the rule set and the lookup table. If the rule set or lookup table are not checked out, the test uses the activated version of the lookup table.

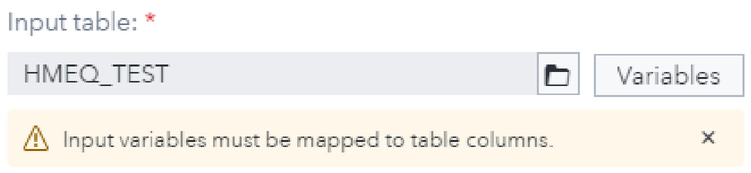
1. On the **Scoring** tab, click the **Tests** tab.
2. Click **New Test**. The New Test window appears.
3. Enter a name for the test if you do not want to use the default name. The name cannot contain forward slashes (/) or curly braces ({}).
4. (Optional) Enter a description for the test. Descriptions are limited to 1000 characters.
5. (Optional) Click 🗀 for the **Location** field, and select the folder where you want to save the test definition and results.

   TIP Selecting a location 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 test files.

6. Click 🗀 for the **Input table** field, select the input table for the test, and click **OK**.
7. Verify or change the variable mappings. To run a full test, map all of the input variables to columns in the input table that you selected for the test. To run a partial test, you can map only the input variables that are needed for the test.

SAS Intelligent Decisioning automatically maps the input variables in the decision to columns in the input table when the names and data types of the variables match those of the table columns.

If any input variables are not mapped to input columns or to static values, the application displays a warning message. At run time, SAS Intelligent Decisioning assigns missing values to input variables that are not mapped.



Input variables must be mapped to table columns.

You can change the automatic variable mappings in the Variable Mappings window.

To change variable mappings:

a Click **Variables**. The Variable Mappings window appears.

b For each input variable, select the table column to which the variable should be mapped. If the input table contains more than 25 columns, click **More columns** to display additional column names. Alternatively, for Decimal, Integer, and Character variables, you can select **Use value** for the table column, and specify a literal value in the **Value** column. When you are entering literal values, remember these rules:

- Do not enclose character strings in quotation marks.
- To specify a missing value for character variables, select **Use value** and leave the **Value** column empty. When SAS Intelligent Decisioning generates code, it generates an empty string (`''`). For numeric values, enter a period (.).

Note: For content that will be published to a SAS Micro Analytic Service destination, the values of date and datetime input variables must be numeric. They cannot be formatted values such as `18jul2019`. Also, integer variables are converted to decimal variables when the content is published.

c Click **OK** to close the Variable Mappings window.

8 (Optional) Click **Advanced** to display the advanced options.

9 (Optional) Click 🗀 and select the library where you want to write the output of the test.

10 (Optional) Enter a name for the test results output table if you do not want to use the default name. The default name is `*test-name_object-name_timestamp_*output`.

11 (Optional) Select the version of the decision that you want to test.

12 (Optional) Select the variable that you want to serve as an input debug variable. You can specify an input-only variable or an input-output variable. The decision writes the name and value of this variable to the log for each input record that is processed. It writes the value just before the logic of the decision is executed for the input record.

For more information, see "Debugging Decision Tests" on page 287.

13 (Optional) Select **Preserve unmapped columns in the output table** if you want columns that are not mapped to an output variable to be written to the output table.

14 (Optional) To generate a data grid variable of all decision variables and their values, select **Record variable values by node**. For more information, see "Enabling Value Tracing" on page 288.

15 (Optional) Select **Keep filtered records in output table** to include records in the test output that would normally be excluded by any of the filtering rule sets in the decision. When you select this option, SAS Intelligent Decisioning creates a new column in the output table named _filter_. If the value in this column is 0, then the record does not match any of the conditions in any filtering rule set, and the record is filtered out of the output when the decision is run in a production environment. If the value in this column is 1, then the record matches at least one filtering condition, and the record is included in the output.

Note: This option is ignored if the decision does not contain a filtering rule set.

16 Click **Run** to run the test. Alternatively, click **Save** to save the test definition without running it.

The status of the test is indicated by the icon in the **Status** column. For explanations of each icon, see "Status Icons for Tests" on page 306.

17 Click ▤ in the **Results** column to view the results of the test.

18 In the test results window, click **Test Results** in the navigation panel to display the URIs and other information for the test. Click **Output**, **Code**, or **Log** to display the output data set, the code that was generated by SAS Intelligent Decisioning, or the SAS log that was generated when the code was run.

TIP You can click on the values of character variables to display the entire value in a separate window. For data grid variables, you can choose to view the variable value in three different formats:

- Click the **Data Grid** tab to view the data grid value as a table.
- Click the **Formatted** tab to view the data grid as a formatted JSON character string.
- Click the **Plain** tab to view the data grid as an unformatted character string.

TIP On the **Log** page, you can click ⤓ to download the log file.

Debugging Decision Tests

When you create a test, you can specify a variable to use as a debugging variable in the **Input debug variable** field. You can specify an input-only variable or an input-output variable. The decision writes the name and value of this variable to the log for each input record that is processed. It writes the value just before the logic of the decision is executed for the input record. For more information, see Step 12 of “Create and Run a New Test” on page 285.

When you specify an input debug variable, you can use the sas.decisions.messageOrder configuration option to control whether the log messages are written as they are produced or after the decision executes. For more information, see “sas.decisions.messageOrder” in *SAS Intelligent Decisioning: Administrator’s Guide*.

When you specify an input debug variable, SAS Intelligent Decisioning automatically sets the maximum number of threads that can be allocated for the test to 1. Setting the thread count to 1 ensures that the variable's values are written to the log in the correct order and are not affected by different threads completing at different times.

To capture variable values for input-only or temporary variables after the decision logic has executed for a specific record, you can specify that the variable is an output variable, and then re-run the test. Before publishing the decision to a production environment, return the input and output settings for the variable to their previous settings. For more information, see “Input Variables, Output Variables, and Temporary Variables” on page 224 and “Edit Variable Properties” on page 229.

TIP To capture intermediate variable values that occur during the execution of the decision, you can use DS2 code files that include PUT statements for the variable. For more information, see Chapter 6, “Using Custom Code Files,” on page 151.

Enabling Value Tracing

When you test a decision, you can select the **Record variable values by node** check box to trace how variable values change for each node in the decision flow. When you select this option, SAS Intelligent Decisioning creates an output data grid variable named nodeTraceDataGrid. This data grid contains a row for each rule set, model, nested decision, and code file node in the decision. It contains one column for every variable in the decision. Condition nodes, branch nodes, and record contact nodes are not included.

The first column in nodeTraceDataGrid contains the node name, and the remaining columns contain the values of the input and output variables, including temporary variables.

Within the nodeTraceDataGrid variable, data grids are written as string variables, and their length is limited to 32767 characters.

When a node sets a variable’s value, that value appears unchanged in all subsequent rows until another node changes the variable to a different value. In the following example, the Code_1 node sets score variable to a value of 1. The Code_2 and Code_3 nodes do not change that value. The Rule_set_1 node changes the value to 1.25. The Code_1 node sets the value of the scoreProbDataGrid variable to null. The Code_3 node sets the value of scoreProbDataGrid to a data grid with two columns: score and probability.

```
{
   "Data Grid": [{
      "metadata": [
         {"Node Name": "string"},
         {"additionalDiscount": "double"},
         {"age": "double"},
         {"cost": "double"},
         {"currentPhone": "double"},
         {"score": "double"},
         {"treatmentName": "double"},
         {"ServicePlans_out": "string"},
         {"newScore": "double"},
         {"probability": "double"},
         {"scoreProbDataGrid": "string"}
      ]
   }, {
      "data": [
         ["Code_1", null, 42, 699.99, null, 1, null, null, 1.1, 0.625547, null],
         ["Code_2", null, 42, 699.99, null, 1, null, null, 1.1, 0.625547, null],
         ["Code_3", null, 42, 699.99, null, 1, null, null, 1.1, 0.625547,
            "{\"Data Grid\":[{\"metadata\":[{\"score\":\"double\"},
             {\"probability\":\"double\"}]},{\"data\":[[1.23,4.56]]}]}"],
         ["Rule_set_1", 99.99, 42, 699.99, null, 1.25, null, null, 1.1, 0.625547,
            "{\"Data Grid\":[{\"metadata\":[{\"score\":\"double\"},
             {\"probability\":\"double\"}]},{\"data\":[[1.23,4.56]]}]}"],
         ["Combined_Treatments", 99.99, 42, 699.99, null, 1.25, null, null, 1.1, 0.625547,
            "{\"Data Grid\":[{\"metadata\":[{\"score\":\"double\"},
             {\"probability\":\"double\"}]},{\"data\":[[1.23,4.56]]}]}"]
      ]
   }]
}
```

Note: Value tracing increases the amount of code that is generated for a decision and the time required to execute the decision. The magnitude of the increases depend on the number of variables and complexity of the decision.

Test a Scenario

Create and Run a Scenario Test

IMPORTANT If you are testing a decision that uses functions that are defined in a custom context file, verify that the context file is specified in the **Custom context** field on the **Properties** before running the test.

If you are testing a decision that uses treatment groups and both the decision and the treatment group are checked out, the test uses the checked-out version of the treatment group. If the treatment group is not checked out, the test uses the activated version of the treatment group.

If the decision uses a rule set that uses a lookup table and all three objects (the decision, the treatment group, and the lookup table) are checked out, the test uses the checked-out versions of both the rule set and the lookup table. If the rule set or lookup table are not checked out, the test uses the activated version of the lookup table.

1 On the **Scoring** tab, click the **Scenarios** tab.

2 Click **New Test**. The New Scenario Test window appears.

3 Enter a name for the test if you do not want to use the default name. The name cannot contain forward slashes (/) or curly braces ({}).

4 (Optional) Click 🗀 for the **Test definition location** field, and select the folder where you want to save the test definition.

> **TIP** Selecting a test definition location is optional, but it is highly recommended. Storing test definitions in a folder simplifies the tasks of setting permissions and transferring the test files.

5 Click 🗀 for the **Output table location** field, and select the folder where you want to save the test results.

6 (Optional) Select **Enable variable change path logging** to record variable value information and variable mapping information for each node in the decision. The application writes this information to the CAS log. Selecting this option enables you to generate a change path report in Step 16. For more information, see "Using Variable Change Path Logging" on page 294.

Note: If the decision is complicated or uses very large data grid variables, the change path file might be very large, and you might encounter memory issues. Do not select both the **Enable variable change path logging** option and the **Enable variable assignment logging** option for the same test.

> **TIP** If you do not see the **Enable variable change path logging** check box, drag the horizontal divider ····▲ down.

7 (Optional) Select **Enable variable assignment logging** to include information in the log about how a variable's value changes as the decision executes. The application writes a log entry each time a variable is assigned a value. Selecting this option enables you to generate a variable assignment report in Step 17. For more information, see "Using Variable Assignment Logging" on page 296.

Note: If the decision is complicated or uses very large data grid variables, the change path file might be very large, and you might encounter memory issues. Do not select both the **Enable variable change path logging** option and the **Enable variable assignment logging** option for the same test.

8 (Optional) Select the version of the decision that you want to test.

9 (Optional) Enter a description for the test. Descriptions are limited to 1000 characters.

10 Enter the values that you want to use for each input variable. You do not have to enter values for every input variable. At run time, SAS Intelligent Decisioning uses missing values to input variables for which you do not specify a value.

Note: Values longer than 32767 characters for character variables that are either input-only or input-output will be truncated. You cannot enter input values for variables of type binary or of type varying-length binary.

To enter values for the columns in a data grid variable:

a Click in the **Value** field, and then click ✎. The Edit Data Grid window appears.

b Click + to add the row, and then enter the values for each column.

Repeat this step for each row of values that you want to add to the data grid.

TIP By default, data grid column names appear across the top of the data grid view, and row numbers appear down the left side. You can click ⤾ to change the view of the data grid so that row numbers appear across the top and data grid column names appear down the left side.

c Click **OK** to save the data grid values and close the Edit Data Grid window.

11 (Optional) For each output variable, select the **Include** check box and enter the expected output value. The **Include** check box controls whether a variable's expected value is used to determine the status of a scenario test. If you select **Include** for a variable and the test does not return the expected value, the test status is set to Completed with warnings (⚠). If you do not select the check box, the application ignores the expected value of that variable when it determines the status of the test.

To enter expected values for the columns in a data grid variable, click in the **Expected Output** field, and follow the instructions in Step 10.

Note: A scenario test cannot verify issues with trailing spaces. For example, it cannot distinguish between a string that contains a single space `' '` and a string that contains three spaces `'   '`.

12 Click **Run** to run the test. Alternatively, click **Save** to save the test definition without running it.

The status of the test is indicated by the icon in the **Status** column. For explanations of each icon, see "Status Icons for Tests" on page 306.

13 Click ▤ in the **Results** column to view the results of the test.

14 On the Test Results page, click **Test Results** in the navigation panel to display the URIs and other information for the test. Click **Input** to display the values of input variables and dynamic attributes. Click **Output**, **Code**, or **Log** to display the output data set, the code that was generated by SAS Intelligent Decisioning, or the SAS log that was generated when the code was run.

The **Log** tab contains two subtabs, **Diagram** and **Log**. The **Diagram** subtab displays a decision diagram in the left panel and the list of errors or warnings in the right panel. Click ⓘ or ▲ to to display list the errors or warnings. In the diagram, a red X identifies a decision node that produced an error and a yellow triangle identifies a node that produced a warning. Click **View log** next to an error or warning in the list to jump to the line that generated that error on the **Log** subtab.

On the **Output** page, click **Show All** to display the expected and actual values of all output variables. Click **Show Differences** to display only the variables whose expected values do not match the actual values that were returned by the test.

TIP You can click on the values of character variables to display the entire value in a separate window. For data grid variables, you can choose to view the variable value in three different formats:

- Click the **Data Grid** tab to view the data grid value as a table.
- Click the **Formatted** tab to view the data grid as a formatted JSON character string.
- Click the **Plain** tab to view the data grid as an unformatted character string.

TIP On the **Output** page, click **Export** to export the output table as a comma-separated values (CSV) file. On the **Log** page, click ⤓ to download the log file.

15 Click **Close** to close test results and return to the **Scoring** tab.

16 (Optional) Generate a variable change path report. Select the scenario test, click ⋮ , and select **Generate path report**. SAS Intelligent Decisioning generates the report, adds it as a comment attachment to the scoring test, and opens the **Comments** property panel.

Click on the report file name to open the report. The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Note: If the decision is complicated or uses very large data grid variables, the report file might be very large, and you might encounter memory issues.

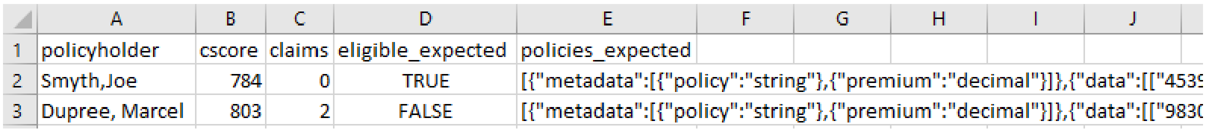
17 (Optional) Generate a variable assignment report. Select the scenario test, click ⋮ , and select **Generate assignment report**. SAS Intelligent Decisioning generates the report, adds it as a comment attachment to the scoring test, and opens the **Comments** property panel.

Click on the report file name to open the report. The result depends on which browser you are using and the browser's default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Import Scenario Test Definitions

You can import scenario test values from a comma-delimited (CSV) file. Each line in the CSV file is imported as one scenario test. In the CSV file, add a column of values for each variable. In the header row, enter the names of the input variables and of the output variables with `_expected` appended to the name.

For example, suppose your scenario test has the input variables `policyholder`, `cscore`, and `claims`, and it has the output variables `eligible` and `policies`. An import file for this test might appear in a spreadsheet application as shown in the following figure.



	A	B	C	D	E	F	G	H	I	J
1	policyholder	cscore	claims	eligible_expected	policies_expected					
2	Smyth,Joe	784	0	TRUE	[{"metadata":[{"policy":"string"},{"premium":"decimal"}]},{"data":[["4539					
3	Dupree, Marcel	803	2	FALSE	[{"metadata":[{"policy":"string"},{"premium":"decimal"}]},{"data":[["9830					

The policies output variable in this example is a data grid variable. To enter values for data grid variables, use the JSON string format described in "Introduction to Data Grids" in *SAS Intelligent Decisioning: Using Data Grids*.

The format for date and datetime variables depends on your locale. Use the same format that is created by the date and datetime pickers when you click or to enter initial values for custom variables on the **Variables** tab.

To import scenario test definitions:

1 On the **Scoring** tab, click the **Scenarios** subtab, and then click **Import Scenarios**. The Import Scenarios window appears.

2 In the **Import from** field, click and select the CSV file that contains the scenario test values.

Note: The import file is limited to 10 MB.

3 Select or enter the encoding of the CSV file.

4 Enter a prefix for the scenario test definitions. SAS Intelligent Decisioning appends a number to this prefix for each test definition. The prefix can include double-byte characters and special characters, including single quotation marks.

5 (Optional) Click for the **Folder location** field, and select the folder where you want to save the test definitions.

6 Click for the **Output table location** field, and select the folder where you want to save the test results.

7 (Optional) Select **Enable variable change path logging** to enable variable change path logging for the imported tests. For more information, see "Using Variable Change Path Logging" on page 294.

8 Select **Enable variable assignment logging** to enable variable assignment logging for the imported tests. For more information, see "Using Variable Assignment Logging" on page 296.

9 Click **Import**.

Compare Different Scenarios

You can display the scenario definitions or results of two or more tests side-by-side. On the **Scenarios** tab, select two or more tests, click ⋮, and select one of the following options:

Compare ⇨ Definitions
displays the input values and the expected output values that you entered for both tests. To edit the input values for a scenario test definition, click ✎ next to the test name under **Input Table**. To edit the expected output values for a test, click ✎ next to the test name under **Output Table**.

Compare ⇨ Results
displays the input values and the actual output values that were generated by the test. To display both the expected values and the actual values in the output table, select **Display expected values**. For each variable for which you selected **Include** in Step 11 on page 291, the application highlights the variable if the actual and expected values do not match.

Click **Export** to export the results comparison as a comma-separated values (CSV) file. The result depends on which browser you are using and the browser’s default download actions. The browser might automatically download the file and display a notification in the bar at the bottom of the browser window. Alternatively, the browser might prompt you to save the file or open it with a specific application.

Using Variable Change Path Logging

When you run a scenario test for a decision, you can select **Enable variable change path logging** to write information about node variables to a log as each node in the decision runs. Each log entry records value and mapping information for the node’s variables. After the scenario test has run, select the **Generate path report** menu option to generate a report from the log entries. SAS Intelligent Decisioning creates the report in a text file and adds this text file to the scenario test as a comment with an attachment.

For all node types except branch nodes, the report records the initial value, the final value, and the mapping for each of the node’s variables. The information for each node, except branches, is enclosed in lines that begin with IN and OUT. For example, if the scenario test executes the treatment group named credit_card_offers followed by the rule set named calc_num_offers, the trace file contains the following lines:

```
IN: decision - credit_card_offers
   IN: treatment group - card_offers
      ...<<node variable values>>...
   OUT: treatment group - card_offers
   IN: ruleset - calc_num_offers
      ...<<node variable values>>...
   OUT: ruleset - calc_num_offers
   ...<<decision variable values>>...
OUT: decision - credit_card_offers
```

Between the IN and OUT lines for each node, the report records the initial value, final value, and mapping for each of the node's variables. For example, if a node has an input-only variable named RATE, an output-only variable named NUM_OFFERS, and an input-output variable named ACCTSCORE, the report might contain the following lines:

```
*Setting inOut Variable - ACCTSCORE ( mapped to ACCTSCORE )
    *ValueReceived: 0.698
    *ValueReturned: 0.796
*Setting input Variable - RATE ( mapped to RETURN )
      *ValueReceived: 15.4
*Setting output Variable - num_offers ( mapped to num_offers )
    *ValueReceived: .
    *ValueReturned: 2
```

Note: A period (.) indicates a missing value. Boolean values are recorded as 0 (false) and 1 (true).

Between the last node of the decision and the END node, the report contains the same information for each decision variable. These lines include the name of the decision. For example:

```
 *Setting credit_card_offers output Variable - zero_offers ( mapped to zero_offers )
    *ValueReceived: .
    *ValueReturned: 0
```

For branches, the report records the branch condition and the path that the test takes out of the branch. For example, suppose the decision has a range branch that has different paths for the CLAGE variable. If the test follows the path for the range of values 101–300, the trace file contains the following line:

```
IN: branch - Range(CLAGE) 101 - 300==>GEN==>
```

In this case, the branch label is the default value `101 - 300`. If you customize the branch label, then `101 - 300==>GEN==>` is replaced with the custom label.

If the decision has a Yes/No branch with the condition `zero_offers=True`, and the test follows the No path, the trace file contains the following line:

```
IN: branch - zero_offers=True onFalse
```

If the **Score rows in this data grid** option is selected in the Input Variables property panel for a node, the IN line for that node lists the data grid that is being scored. For example, if the credit_card_offers decision contains a model named credit_limit and that model scores the data grid named card_offers_out, the trace file contains the following line:

```
IN: model - credit_limit scoring datagrid card_offers_out of credit_card_offers row 1
```

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" in *SAS Intelligent Decisioning: Administrator's Guide*.

Instructions for enabling change path logging and for generating the report are in Step 6 and Step 16 of "Create and Run a Scenario Test" on page 289.

Using Variable Assignment Logging

When you run a scenario test for a decision, you can select **Enable variable assignment logging** to record how variable values change as a decision runs. This option adds the following statement to the generated code:

```
DS2_OPTIONS TRACEVARIABLES;
```

Note: To enable this option, your administrator must define the App.TableServices.DS2.Runtime.TraceVariables logger, and set other configuration options.

If you select this option and configure the logger, SAS Intelligent Decisioning writes a log entry each time a variable is assigned a value. Each log entry includes the following information:

- the line number in the generated code where the value is assigned
- the type of code block in which the value was assigned (`package` or `thread`)
- the name of the code block that assigned the value
- the line number in the individual code block
- the variable name and its assigned value

Note: Log messages are limited to 32000 bytes, including the log message header. In locales that use multi-byte character sets, log entries might be truncated to 8000 characters.

For example, if the decision includes a rule set named rs_hmeq_loan that sets the value of a variable named `risky`, the trace file might contain the following lines:

```
Line 1053 (package "56b88b7c_60c8_4c30_c8ec_82ea1b788180_rs_hmeq_loan" line 102):
    "risky"=0
Line 1058 (package "56b88b7c_60c8_4c30_c8ec_82ea1b788180_rs_hmeq_loan" line 107):
    "risky"=1
```

These entries indicate that on line 1053 of the generated code, the variable risky is set to 0. This line is line 102 of the code block that was generated for the rule set. Similarly, the variable was set to 1 on line 1058 of the generated code.

Note: A period (.) indicates a missing value. Boolean values are recorded as 0 (false) and 1 (true).

For data grid variables, SAS Intelligent Decisioning creates a log entry for any assignment that changes the value of the variable and that is performed by using data grid functions or methods. For example, if the generated code uses the SAS Micro Analytic Service setValue method to set the value of a cell in a data grid named zero_offers_out_grid, the log might include the following entry:

```
Line 556 (package "71a328a8_7336_4c26_70c7_b5901f39d692_credit_card_offers_zero_offers"
    line 44):
Cell for column index 1 added for DataGrid "zero_offers_out_grid"
    with value 8d9bee3d-45f0-4249-b4d7-9aa92b865a92
```

SAS Intelligent Decisioning does not write log entries for direct assignments from one data grid variable to another such as `gridVar1 = gridVar2`.

If the data grid variable is an output variable, the log contains the entire value of the variable that is written to the output table. The log entry includes both the column metadata and the values for each column. For example, if the decision includes a treatment group named tg_hmeq, the decision creates an output data grid variable for the treatment group named tg_hmeq_out. In the following log entry, the value of the tg_hmeq_out variable was set on line 2847 in the generated code:

```
Line 2847 (thread "56b88b7c_60c8_4c30_c8ec_82ea1b788180_decisionflowthread" line 205):
"tg_hmeq_out"=[{"metadata":[{"TREATMENT_DEFINITION_GROUP_ID":"string"},
{"TREATMENT_DEFINITION_GROUP_REVISION_ID":"string"},{"TREATMENT_DEFINITION_ID":"string"},
{"TREATMENT_DEFINITION_REVISION_ID":"string"},{"TREATMENT_DEFINITION_NAME":"string"},
{"CLNO":"decimal"},{"DEBTINC":"decimal"},{"DELINQ":"decimal"},{"DEROG":"decimal"},
{"DISCOUNT":"decimal"},{"OFFERCODE":"string"},{"OBJECTNODEID":"string"}]},
{"data":[["e4c3c86d-419f-41d8-8e7f-4ba4dfc01f80","2389be10-0445-4647-bc63-0221a078431d",
"7ab57211-a67a-4e03-adee-dae8436255b5","bbbbe7cd-e11b-49a4-b7f1-afa736503397",
"hmeq_online_marketing_services",23.2,33,2,234,20,"OMS01",
"8b758d75-1e60-4787-9e44-3421faf1548e"]]}]
```

For information about the JSON strings that are used to represent data grid variables, see "Introduction to Data Grids" in *SAS Intelligent Decisioning: Using Data Grids*.

If the sas.decisions.nodetraces.includeRuleFiredPathTrackInfoInVariableAssignmentLogging configuration option is turned on, the log includes rule-fired information and path-tracking information. However, if this configuration option is turned off, you cannot run rule-fired analyses or path-tracking analyses for scenario tests. For more information about the configuration option, see "Decisions Service Properties" in *SAS Intelligent Decisioning: Administrator's Guide*.

After the scenario test has run, select the **Generate assignment report** menu option to generate a report from the log entries. The application creates the report in a text file and adds this text file to the scenario test as a comment with an attachment.

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” in *SAS Intelligent Decisioning: Administrator’s Guide*.

Instructions for enabling variable assignment logging are in Step 7 of “Create and Run a Scenario Test” on page 289.

For more information about data grid variables, see *SAS Intelligent Decisioning: Using Data Grids*.

For more information data grid packages and methods, see the following resources:

- “DS2 Datagrid Package Methods, Operators, and Statements” in *SAS DS2 Language Reference*
- “Using the Datagrid Package” in *SAS DS2 Programmer’s Guide*

Run a Rule-Fired Analysis

If a rule’s conditions evaluate to true, then the rule is said to have *fired*. Rule-fired data includes summary information about how many times each rule fired and detailed information for each time that a rule evaluates to true. See “How Rules Are Evaluated and When Rule-Fired Records Are Generated” on page 32 for more information.

Note: Rule-fired data is recorded only for rule sets, including rule sets in nested decisions and filtering rule sets that are included directly in a decision. Rule-fired data is not recorded for other objects, including filtering rule sets that are used as eligibility rule sets in treatments.

For filtering rule sets that are included directly in a decision, rule-fired data is recorded for an input record if the rule set does not filter out the record. When the rule set filters out an input record, rule-fired data is not recorded for that input record.

For rule sets that iterate over a data grid (in other words, the rule sets score the rows in the data grid), the rule-fired data indicates that the rules in the rule set fire once for the entire data grid instead of firing once for each row in the data grid.

Rule fired data is recorded for nested decisions only if the sas.decisions.includeRuleFiredInformationForSubdecisions configuration option is turned on.If the sas.decisions.nodetraces.includeRuleFiredPathTrackInfoInVariableAssignmentLogging configuration option is turned off, you cannot run rule-fired analyses for scenario tests. For more information, see “Decisions Service Properties” in *SAS Intelligent Decisioning: Administrator’s Guide*.

TIP This rule-fired analysis uses the data that is in the ruleFiredFlags column in the test results output table. To analyze rule-fired data that is in the subject contact history, use the %DCM_GET_SUBJECTCONTACT_HISTORY and %DCM_RULEFIRE_DETAIL macros. For more information, see *SAS Intelligent Decisioning: Macro Guide*.

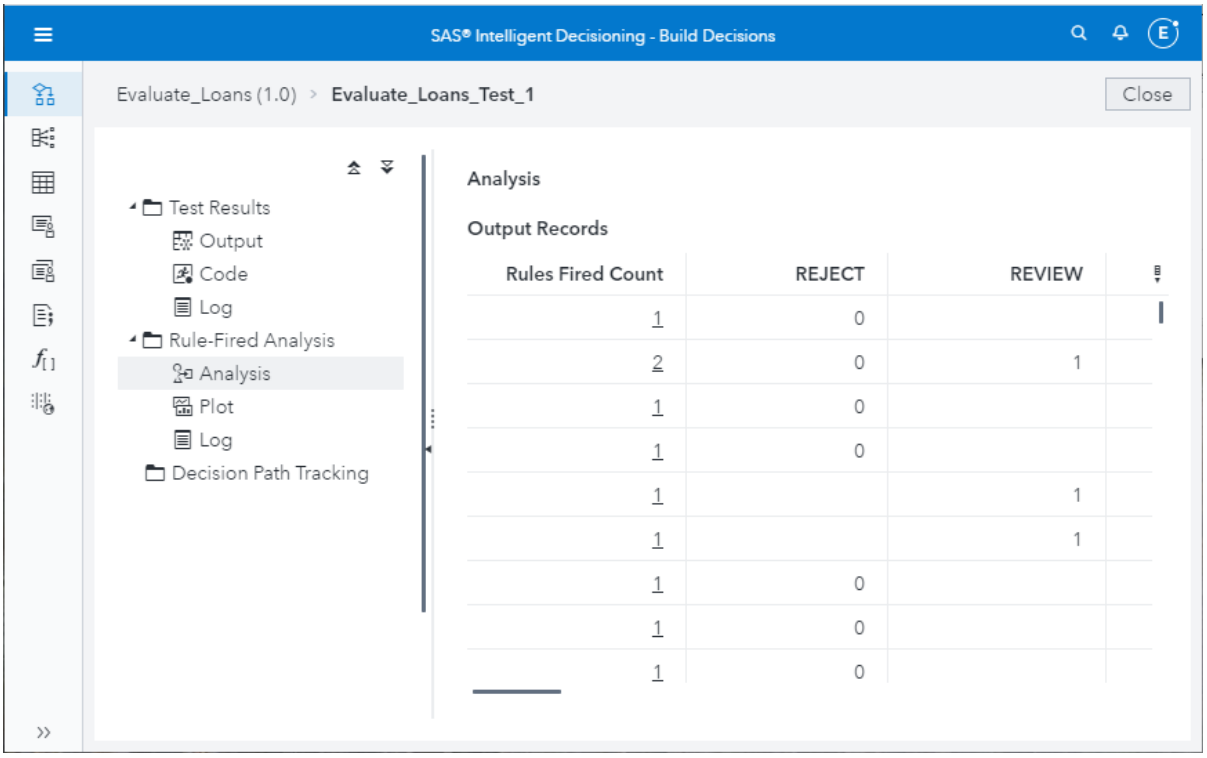
To run a rule-fired analysis:

1 In the test results window, click **Rule-Fired Analysis** in the navigation panel.

2 Click **Run Rule-Fired Analysis**. SAS Intelligent Decisioning analyzes the test results to determine which rules fired for each row in the input table, and displays the Analysis page.

The Analysis page displays the number of rules that fired for each output record that was generated by the decision. The number in the **Rules Fired Count** column is a link to more information. You can click on this link to display the rules that fired for that output row.

For example, in the following display there is one output record for which two rules fired.

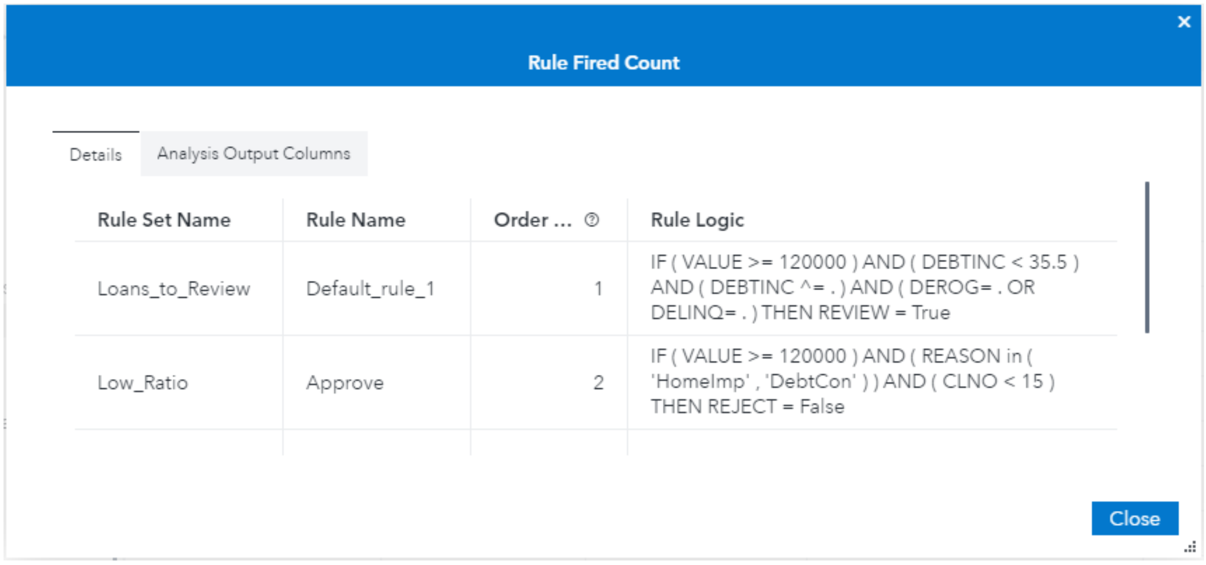

SAS® Intelligent Decisioning - Build Decisions
Evaluate_Loans (1.0) > Evaluate_Loans_Test_1
Close
Test Results
Output
Code
Log
Rule-Fired Analysis
Analysis
Plot
Log
Decision Path Tracking

Analysis

Output Records

Rules Fired Count	REJECT	REVIEW
1	0	
2	0	1
1	0	
1	0	
1		1
1		1
1	0	
1	0	
1	0	

3 Click on a number in the **Rule Fired Count** column. SAS Intelligent Decisioning displays the Rule Fired Count window. This window shows which rules produced the selected output record.


Rule Fired Count

Details | Analysis Output Columns

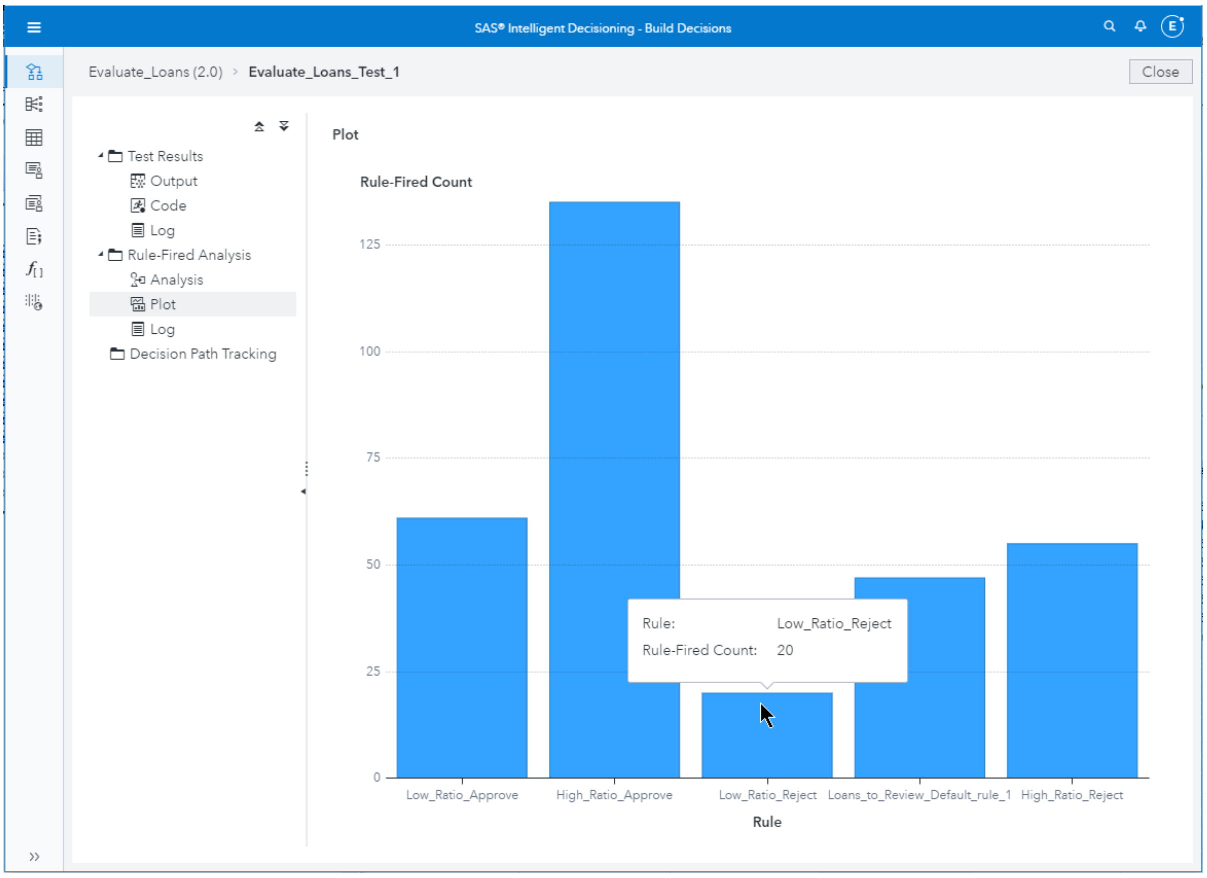
Rule Set Name	Rule Name	Order ...	Rule Logic
Loans_to_Review	Default_rule_1	1	IF (VALUE >= 120000) AND (DEBTINC < 35.5) AND (DEBTINC ^= .) AND (DEROG= . OR DELINQ= .) THEN REVIEW = True
Low_Ratio	Approve	2	IF (VALUE >= 120000) AND (REASON in ('HomeImp' , 'DebtCon')) AND (CLNO < 15) THEN REJECT = False

Close

Note: The value in the **Rule Order** column is the ordinal number of the rule as it occurs in the decision diagram. The values in this column do not indicate the order in which the rules in the decision fired. For decisions that use cross-branch links, click to switch the diagram between two different views of cross-branch links. It might be easier to correlate the rule order numbers with their location in the diagram by displaying the cross-branch links as nodes.

4 Click **Close** to close the Rule Fired Count window.

5 Click **Plot** in the navigation panel. SAS Intelligent Decisioning displays a bar chart that shows how many times each rule fired. Position your cursor over a bar to display the name of the rule and the number of times that the rule fired.



6 Click **Rule-Fired Analysis** in the navigation panel to display the URIs and other information for the rule-fired test.

7 Click **Close** to close the decision.

Run a Path Tracking Analysis

Decision path tracking shows you the route that input records take through the nodes in your decision.

Note: The path-tracking results do not include data for input records that are filtered out with a filtering rule set.

Path-tracking analyses include information for nested decisions only if the sas.decisions.includeRuleFiredInformationForSubdecisions configuration option is turned on.

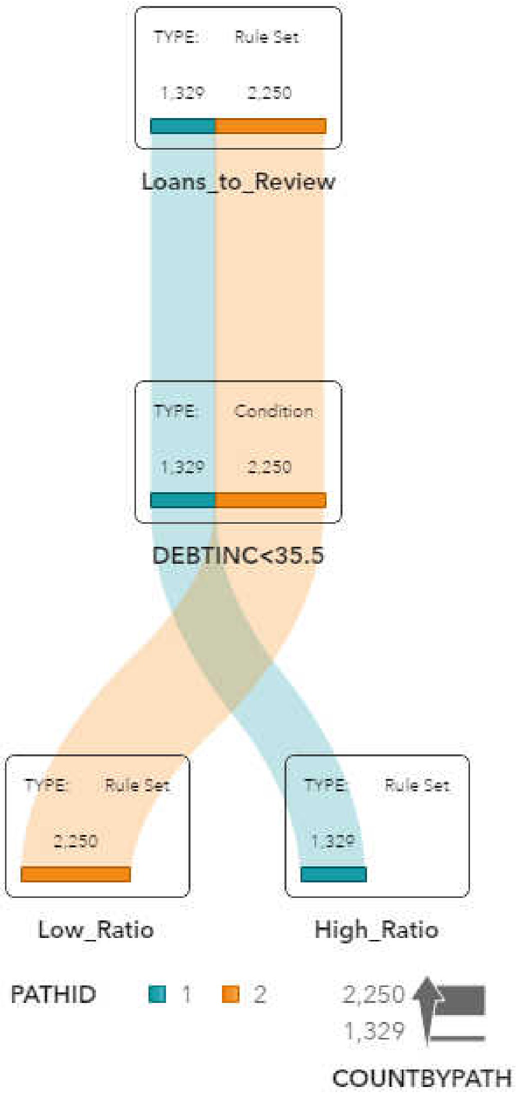
If the sas.decisions.nodetraces.includeRuleFiredPathTrackInfoInVariableAssignmentLogging configuration option is turned off, you cannot run path-tracking analyses for scenario tests.

TIP This path-tracking analysis uses the data that is in the pathID column in the test results output table. To analyze path-tracking data that is in the subject contact history, use the %DCM_DECISION_PATH_FREQUENCY, %DCM_DECISION_PATH_NODES, and %DCM_DECISION_NODES_COUNTS macros. For more information, see *SAS Intelligent Decisioning: Macro Guide*.

1 In the test results window, click **Decision Path Tracking** in the navigation panel.

2 Click **Run Path Tracking** to run a decision path analysis.

3 Click **Analysis and Plot** to display a Sankey diagram that shows the flow of the input records through the nodes in the decision. The numbers in the diagram are the number of rows in the input table that followed each path.



Note: Nodes that are not executed are shown to the right of the Sankey diagram.

4 Click **Node Count** in the navigation panel to display a table showing the number of input records evaluated at each rule set, code file, and model node in the decision.

5 Click **Close** to close the Test Results window.

Content That Is Used by Tests and Scenarios for Decisions

For each object in a decision, the content that is used when the decision is tested depends on the object type.

Object Type	Object Content that Is Used
Nested decision	The version that you select when you add the nested decision to the decision
Rule set	The version that you select when you add the rule set to the decision
Model	The latest version
Treatment group	The active version
Lookup table	The active version
Global variable	The active version
Custom code file	The version that you select when you add the code file to the decision

Validate a Published Decision

You can test the published decision in the publishing destination. When you publish the decision to any destination type except Git, a validation test is automatically defined for that decision in that destination.

Note: When both the SAS Intelligent Decisioning sas.decisions.masnode.removeTrailingUnderscoresFromInput property and the SAS Micro Analytic Service 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.

To run the publishing validation test:

1 On the **Scoring** tab, click the **Publishing Validation** tab. The ○ icon in the **Status** column indicates that the test is not ready to run. The ◉ icon indicates that the test is ready to run.

2 Click on the test name. The Edit Publishing Validation Test window appears.

Note: To generate the name of the publishing validation test, SAS Intelligent Decisioning appends a timestamp to the decision name. The timestamp indicates when the decision was published.

3 (Optional) Click 🗀 in the **Location** field, and select the folder where you want to save the test definition and results.

TIP Selecting a location 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 test files.

4 Click 🗀 in the **Input table** field, and select the input table for the test.

Note: For content that will be published to a SAS Micro Analytic Service destination, the values of date and datetime input variables must be numeric. They cannot be formatted values such as `18jul2019`. Also, integer variables are converted to decimal variables when the content is published.

Note: If the input table contains a character column, and that column contains control characters in any row, do not use the table as input for publishing validation tests.

If you are validating content that was published to SAS Micro Analytic Service, the time required to run the test depends on the number of worker threads on your system, the number of threads in the middle tier, and the network latency between CAS and the middle tier server. It is recommended that you select an input table with as few input records as needed to accurately test the published content. See *SAS Micro Analytic Service: Programming and Administration Guide* for more information.

5 (Optional) Expand the **Advanced** section, click 🗀 in the **Output data library** field, and select a library to store the validation test output data.

6 Click **Run** to run the test. Alternatively, click **Save** to save the test definition without running it.

The status of the test is indicated by the icon in the **Status** column. For explanations of each icon, see "Status Icons for Tests" on page 306.

7 Click ▤ in the **Results** column to view the test results.

8 In the test results window, click **Test Results** in the navigation panel to display the URIs and other information for the test. Click **Output**, **Code**, or **Log** to display the output data set, the code that was generated by SAS Intelligent Decisioning, or the SAS log that was generated when the code was run.

TIP You can click on the values of character variables to display the entire value in a separate window. For data grid variables, you can choose to view the variable value in three different formats:

- Click the **Data Grid** tab to view the data grid value as a table.
- Click the **Formatted** tab to view the data grid as a formatted JSON character string.
- Click the **Plain** tab to view the data grid as an unformatted character string.

TIP On the **Log** page, you can click ⤓ to download the log file.

9 Click **Close** to close the decision.

Working with Test and Validation Test Output Data

After you run a test or a publishing validation test, you can work with the output table in other SAS applications to analyze the data, create and compare models, discover relationships hidden in the data, and generate reports based on the data.

Note: The actions available to you depend on the applications that are available at your site.

On the Test Results page, select the **Output** table in the navigation pane, click **Actions**, and select one of the following options:

Explore Lineage
opens SAS Lineage Viewer. SAS Lineage Viewer enables you to better understand the relationships between objects in your SAS Viya applications. These objects include data, transformation processes, reports, and visualizations. For more information, see *SAS Lineage: User's Guide*.

Explore and Visualize Data
opens the output table in SAS Visual Analytics. SAS Visual Analytics enables you to create, test, and compare models based on the patterns discovered during exploration of the data. You can export the model before or after performing model comparison for use with other SAS products or to put the model into production. SAS Visual Analytics supports a range of visualization, discovery, and reporting features. For more information, see *Welcome to SAS Visual Analytics*.

Prepare Data
opens the output table in SAS Data Studio. SAS Data Studio enables you to perform data transforms such as joining tables, appending data to a table, transposing columns, creating calculated columns, and so on. For more information, see *SAS Data Studio: User's Guide*.

Status Icons for Tests

Icon	Status
●	The test is not ready to run. The test definition is not complete, or it might contain errors.
◎	The test has been defined and can be run. Some input variables have not been mapped or have not been assigned a value, so the test might execute only a single path through the decision flow.
◉	The test is defined correctly and is ready to run.
◌	The test is running.
✓	The test completed successfully.
✓⚠	The test completed, but warnings were issued in the SAS log. The URI to the log file is shown on the Test Results page. On the Test Results page, click **Test Results** in the navigation panel to display the URI.
⊗	The test did not run successfully. Check the SAS log for information. The URI to the log file is shown on the Test Results page. On the Test Results page, click **Test Results** in the navigation panel to display the URI.

Manage Comments for Decision Nodes and Tests

You can associate comments and attachments with the Start and End nodes in a decision and with decision tests.

To open the Comments properties pane, open the decision, select the Start node, End node, or decision test, and click 💬 in the property pane.

To add a new comment, enter the comment in the text box and click **Post**.

To add an attachment to a comment, click 📎, select the file that you want to attach, and click **Post**. (The attachment icon appears after you enter text in text box.) You cannot attach executable files such as BAT and EXE files.

To reply to a comment, click **Reply**, enter your reply in the text box, and click **Post**.

To delete a comment, click 🗑 for that comment.

Executing Published Content

How you execute published content depends on the destination to which the content is published.

Executing Content Published to SAS Micro Analytic Service Destinations

The user who is executing the published content must be authenticated. In SAS Viya, authentication options vary, based on which interface and operating system are used in your environment. External mechanisms include direct LDAP authentication, host authentication, Kerberos, Security Assertion Markup Language (SAML), and OAuth 2.0 with OpenID Connect. For additional information, see *SAS Viya Platform: Authentication*

When a decision is published from SAS Intelligent Decisioning to a SAS Micro Analytic Service destination, an EXECUTE step is created in the published module. For information about the request and response data formats used in this step, see Execute a step in the REST API documentation for the Micro Analytic Score API. For an example that uses Python to execute a published decision in the maslocal destination, see “Execute a Published Decision” in *SAS Intelligent Decisioning: Decision Management REST API Examples*.

IMPORTANT If the SAS Micro Analytic Service configuration property `service.removetrailingunderscoresfrominputs` and the SAS Intelligent Decisioning property `sas.decisions.masnode.removeTrailingUnderscoresFromInput` are not defined or are set to **`False`**, add an underscore to the name of each input variable. If these options are set to **`True`**, do not add underscores. Your administrator can add the SAS Micro Analytic Service property to the `supplementalProperties` section in the `sas.microanalyticservice.system` configuration definition in SAS Environment Manager. The default value for these options is **`False`**.For additional information, see “sas.microanalyticservice.system: supplementalProperties” in *SAS Micro Analytic Service: Programming and Administration Guide* and “sas.decisions.masnode.removeTrailingUnderscoresFromInput” in *SAS Intelligent Decisioning: Administrator’s Guide*.

Executing Content That Has Been Published to SAS Cloud Analytic Services Destinations

To execute content that has been published to SAS Cloud Analytic Services (CAS), use the CAS Model Publishing and Scoring action set. For example, the following code runs a model named Evaluate_Loans on the local CAS server.

```
/* Start a CAS session named _mmcas_. */
cas _mmcas_;

/* Create librefs for all existing caslibs so that they */
/* are visible in the SAS Studio Libraries tree.          */
caslib _all_ assign;

proc cas;
    /* Specify the session to use for the runModelLocal action. */
    session _mmcas_;

    /* Define the parameter list for the runModelLocal action. */
    /* Reload the destination model table ("targetCode")       */
    /* before you execute the decision.                        */
    destination_model_table = "targetCode";
    destination_model_lib = "public";

    destination_model = "Evaluate_Loans";

    dp_inputTable="hmeq_test";
    dp_inputCASLib="public";

    dp_outputTable="hmeq_test_dm";
    dp_outputCASLib="public";

    parmlist = {
        modelTable={
            name=destination_model_table,
            caslib=destination_model_lib
            },
        modelName=destination_model,
        inTable={
            name=dp_inputTable,
            caslib=dp_inputCASLib
        },
        outTable={
            name=dp_outputTable,
            caslib=dp_outputCASLib
        }
    };

    /* Load the modelPublishing action set. */
    loadactionset "modelPublishing";

    /* Run the model locally on the CAS server. */
```

```
      action runModelLocal submit result=r  status=rc / parmlist;
   run;
   quit;
```

If SAS Data Studio is licensed at your site, you can submit this code in SAS Data Studio. To open SAS Data Studio, click ☰ and select **Prepare Data**. For more information, see *SAS Data Studio: User's Guide*.

You can view additional examples of using this CAS action set to execute published content by viewing the test results that are generated by publishing validation tests. On the Test Results page for a decision, click **Code** to display the code that was generated by SAS Intelligent Decisioning. For information about running publishing validation tests and viewing the results, see "Validate a Published Decision" on page 303.

For more information about CAS and the Model Publishing and Scoring action set, see the following documentation:

- *Getting Started with CASL Programming*
- *SAS Cloud Analytic Services: CASL Reference*
- "Model Publishing and Scoring Action Set" in *SAS Visual Analytics: Programming Guide*
- *SAS Cloud Analytic Services: User's Guide*

Content Executed by Published Decisions

When you execute a published decision, the version of the content that is executed depends on the publishing destination.

Destination Type	Content That Is Executed by Published Decisions
SAS Micro Analytic Service	Most of the published decision's content (the SAS Micro Analytic Service module) is locked. Updates to rule sets, models, custom functions, or custom code files that are used in the decision do not affect the published module. Treatment groups are embedded in decisions if the masInlineTreatmentGroup configuration option is turned on. If this option is off, treatment groups are not embedded, and published modules use newly activated versions of treatment groups. For more information, see "sas.decisions.masInlineTreatmentGroup" in *SAS Intelligent Decisioning: Administrator's Guide*. Global variables are not locked. If a new version of a global variable is activated, the newly activated version is used by the published module. Lookup tables are embedded in decisions if the lookupStaticBinding configuration option is turned on. If this option is off, lookup tables are not embedded, and published modules use newly activated versions of lookup tables. For more information, see "sas.businessrules.lookupStaticBinding" in *SAS Intelligent Decisioning: Administrator's Guide*.

Destination Type	Content That Is Executed by Published Decisions
	For Micro Analytic Module nodes, the decision always uses the most recently published version of the specified module.
SAS Cloud Analytic Services	Most of the published decision’s content is locked. Updates to rule sets, models, treatment groups, custom functions, or custom code files that are used in the decision do not affect the published decision. The current values of global variables are included in generated code if the inlineGlobalVariableValues configuration option is turned on, and published modules do not use newly activated versions of global variables. If this option is turned off, the generated code uses a SAS format to retrieve the current value of the global variable when the published module is run. For more information, see “sas.decisions.inlineGlobalVariableValues” in *SAS Intelligent Decisioning: Administrator’s Guide*. Lookup tables are locked if the lookupStaticBinding configuration option is turned on. If this option is off, lookup tables are not locked, and published modules use newly activated versions of lookup tables. For more information, see “sas.businessrules.lookupStaticBinding” in *SAS Intelligent Decisioning: Administrator’s Guide*.
Teradata, Hadoop, and Container Destinations	All of the published decision’s content, including treatment groups, custom functions, lookup tables, and global variables is included inline in the published module. Updates to the objects used in the decision are not used by the published module.

View All Published Decisions

The Deployments category view is a dashboard that provides a centralized view of information about all the decisions that are being used in your environment.

Click ⇨ to navigate to the Deployments category view.

The tiles at the top of this view visually display key metrics about the total number of decisions in your environment. For example, the tiles display information such as destination types, deployments to single destinations or to multiple destinations, and deployments with and without models.

TIP You can click ⌃ to hide the tiles.

The tabular view below the tiles displays details about each published decision in your environment. For example, details include information such as the decision name, the published destination, the published date, the number of nodes, and associated tags. By default, the decisions that were published within the last 6 months are included.

The table also includes the current version for each decision. You can click a decision version in the Decision (Version) column to see the details for that version. The decision version opens on the **Decision Flow** tab. For more information about viewing and editing decisions, see "About Decisions" on page 215 .

You can search for a decision by name by typing in the search field. Click **Advanced search** to search for deployed decisions that were published on a specific date or within a range of dates.

Appendix 1

Querying the Subject Contact Database

About the Subject Contact Database 313

How to Query the Subject Contacts Database 314

Using Views Defined by SAS Intelligent Decisioning 315

Columns in the dcm_trt_fixed_attr_vw and dcm_trt_dynamic_attr_vw Views . . 316

Columns in the dcm_treatments_contacts_vw View 318

Examples: 321

Determine Whether Any Treatments Were Sent to a Subject 321

Determine Whether a Treatment was Sent to a Subject within a Specific Time Frame 322

Determine Whether a Treatment with a Specific Attribute Value Was Sent to a Subject 322

Determine Whether a Treatment Was Presented to a Subject within a Specified Time Frame 322

Determine Whether a Subject Responded to a Treatment within a Specified Time Frame 323

Determine Whether a Subject Responded to a Treatment with a Specific Attribute Value within a Specified Time Frame 323

Determine Whether a Subject Responded to Any Treatment within a Specified Time Frame 324

About the Subject Contact Database

The subject contact database contains the records written by record contacts nodes that are in decisions. In real-time destinations (SAS Micro Analytic Service destinations), each record contacts node writes a record to the subject contact database. In all other destinations (SAS Cloud Analytic Services [CAS], Teradata, and Apache Hadoop), these nodes create output variables that contain the contact information in JSON syntax. You can use these variables and the SAS Decision Management REST API to update the subject contact database. For example, your enterprise might have a batch job that runs at specific times and updates the subject contact database.

When you add a record contacts node to a decision, you specify the data grid variable and the decision variables that you want to track in the subject contacts record. When the decision contains a treatment, you usually specify the output data grid that is produced by the treatment as the data grid to track. This data grid contains information about the treatments for which the subject qualifies. You usually also track the decision variables that are mapped to dynamic attributes. The record contacts node also generates a response tracking code that you can use in a REST API request to update the subject contact database with information about the subject’s response to the treatments.

For more information about treatments, attributes, and record contacts nodes, see Chapter 3, “Working with Treatments and Treatment Groups,” on page 83 and “Adding Record Contacts Nodes” on page 233. Information about the Subject Contacts REST API is available at https://developer.sas.com.

You can query the subject contact records and treatment definitions to determine which treatments a subject qualifies for, whether a treatment has been presented to a subject, which treatments a subject has responded to, and so on. SAS Intelligent Decisioning provides database views that you can use to query the database.

How to Query the Subject Contacts Database

If you are using the PostgreSQL instance that was installed with SAS Viya for your database, the recommended way to query the database is to use the SAS Decision Management REST API. Complete reference information for the Decision Management API is available at http://developer.sas.com. For examples of using the Subject Contacts API, see the examples on GitHub. For additional information and examples, see *SAS Intelligent Decisioning: Decision Management REST API Examples*.

If you are using a third-party database, you can use the database views that are provided by SAS Intelligent Decisioning. You can create a data query file in the SQL editor that uses these views and include the data query file in a decision. For more information, see “Using Views Defined by SAS Intelligent Decisioning” on page 315 and “Data Query Files” on page 159.

Using Views Defined by SAS Intelligent Decisioning

SAS Intelligent Decisioning defines views that you can use to query the subject contacts records and treatment definitions. These views are created automatically when the subject contacts service and the decision-treatment service are started.

SAS Intelligent Decisioning defines the following three views:

dcm_trt_fixed_attr_vw
: contains fixed treatment attributes for treatments that are members of a treatment definition group. This view enables you to retrieve the list of fixed attributes for a specific version of a treatment group. Treatments can be used in multiple treatment groups, so the table might contain multiple records for the same attribute. This view includes the values that were assigned to the attributes. The attribute values for the same treatment version are the same, but the attribute values might differ between treatment versions.

dcm_trt_dynamic_attr_vw
: contains dynamic treatment attributes for treatments that are members of a treatment definition group. This view enables you to retrieve the list of dynamic attributes for a specific version of a treatment group. Treatments can be used in multiple treatment groups, so the table might contain multiple records for the same attribute.

  These attributes can be assigned either a static value in the treatment group, or they can be mapped to a decision variable and receive their value from the variable when the decision executes.

  For attributes that are assigned a static value in a group, this view includes the values that were assigned to them. The values of these attributes might be different for each treatment group in which the treatment that defines the attribute appears.

  For dynamic attributes that are mapped to a decision variable, the attribute values are not present in this view. To trace the values of these attributes, the decision variables must be tracked by the record contacts node. For more information, see “Adding Record Contacts Nodes” on page 233.

dcm_treatment_contacts_vw
: contains detailed contact information for each treatment that was sent to a subject. This table enables you to determine which treatments were offered to a subject either for a specific response (identified by the response tracking code) or for the entire set of history records that are retained in the database. The number of treatments associated with a specific response tracking code or subject ID depends on how many treatments were returned by the decisions that created the contact records.

Note: To query both treatment definitions and subject contact records, you must join an attribute view with a treatment view. To join the views, both the subject contacts records and the treatment definitions must reside in the same database.

These views assume that the input and output values of the treatment groups are not modified before they are recorded by a record contacts node.

Columns in the dcm_trt_fixed_attr_vw and dcm_trt_dynamic_attr_vw Views

Both attribute tables contain the same columns. The dcm_trt_fixed_attr_vw view contains information about fixed attributes, and the dcm_trt_dynamic_attr_vw view contains information about dynamic attributes. For information about treatment attributes, see "Attributes and Attribute Aliases" on page 88.

The following table describes each column in these views and lists the associated schema property in the Decision Management REST API.

Column	**Description**	**Schema Property**
attribute_nm	Name of the attribute.	name in the Treatment Definition Attribute object in the Treatment Definition Collection
attribute_val_fixed_flg	A flag that indicates whether the attribute is fixed or dynamic. For dynamic attributes, this column is set to `false`. For fixed attributes, this column is set to `true`.	readOnly in the Treatment Definition Attribute object in the Treatment Definition Collection
mapping_type_cd	A code that indicates whether dynamic attributes get their value from the decision flow or are set to a static value. For dynamic attributes, this column is set to `constant` if the attribute is assigned a static value in the treatment group. This column is set to `variable` if the attribute gets its value from the decision flow. For fixed attributes, this column is set to `null`.	mappingType in the Treatment Definition Group Attribute Value Mapping object in the Treatment Definition Collection
attribute_default_val	The default value that was set for the attribute when the attribute was defined in the treatment. For dynamic variables, this value might be overridden if the attribute is assigned a static value in the treatment group or if the attribute gets its value from the decision flow. For fixed attributes, this value is the value that is used in the decision flow.	defaultValue in the Treatment Definition Attribute object in the Treatment Definition Collection

Column	Description	Schema Property
attribute_val	The actual value or the variable mapping for dynamic attributes. For dynamic attributes that get their value from the decision flow, this column is set to the name of the variable to which the attribute is mapped in the decision flow. For dynamic attributes that are assigned a static value in the treatment group, this column is set to the static value. For fixed attributes, this column is set to the name of the variable to which the attribute is mapped in the decision flow.	value in the Treatment Definition Group Attribute Value Mapping object in the Treatment Definition Collection
treatment_nm	Name of the treatment definition that contains the attribute.	name in the Treatment Definition object in the Treatment Definition Collection
treatment_id	ID of the treatment definition.	id in the Treatment Definition object in the Treatment Definition Collection
treatment_revision_id	ID of the treatment definition version.	definitionRevisionId in the Treatment Definition Group Member object in the Treatment Definition Group Collection
treatment_version_no	Version number of the treatment definition.	definitionRevisionName in the Treatment Definition Group Member object in the Treatment Definition Group Collection. For example, treatment_version_no could be `2.14`, where 2 is the treatment definition major revision number and 14 is the treatment defintion minor version number.
treatment_grp_nm	Name of the treatment definition group that contains the treatment.	name in the Treatment Definition Group object in the Treatment Definition Group Collection
treatment_grp_id	ID of the treatment definition group.	id in the Treatment Definition Group object in the Treatment Definition Group Collection

Column	Description	Schema Property
treatment_grp_revision_id	Version ID of the treatment definition group.	definitionRevisionId in the Treatment Definition Group Member object in the Treatment Definition Group Collection
treatment_grp_version_major_no	Major version number of the treatment definition group. If the treatment definition group is 2.5, then this column is set to `2`.	majorRevision in the Treatment Definition Group object in the Treatment Definition Group Collection
treatment_grp_version_minor_no	Minor version number of the treatment definition group. If the treatment definition group is 2.5, then this column is set to `5`.	minorRevision in the Treatment Definition Group object in the Treatment Definition Group Collection

Columns in the dcm_treatments_contacts_vw View

The dcm_treatments_contacts_vw view contains detailed information about subject contact records. The following table describes each column in the view and lists the associated schema property in the Decision Management REST API.

Column	Description	Schema Property
contact_id	A unique identifier for the contact record. A contact record with a new contact_id is created when a decision determines that a subject is eligible for a specific treatment and writes the contact record to the database.	id in the contact record
subject_id	The ID of the subject if a subject ID variable is defined in the properties for the decision flow.	subjectId in the contact record
subject_level	The subject level if a subject level variable is defined in the decision flow. For more information, see "Predefined Lookup Tables" on page 116.	subjectLevel in the contact record
created_by_nm	The ID of the user who ran the decision that created contact record.	createdBy in the contact record

Column	Description	Schema Property
modified_by_nm	The ID of the user who last modified the contact record. For example, subject contact records can be modified to include a recording of the treatment presentation or to include information about the subject’s response to the treatment.	modifiedBy in the contact record
creation_dttm	A timestamp that indicates when the contact record was created.	creationTimeStamp in the contact record
modified_dttm	A timestamp that indicates when the contact record was last modified.	modifiedTimeStamp in the contact record
contact_channel_txt	The channel through which the contact occurred. For more information, see “About Channels” on page 90.	channel in the contact record
response_tracking_cd	The response tracking code that was generated by the record contacts node.	responseTrackingCode in the contact record
conclusion_response_txt	The subject’s response to the results of the decision. When the decision includes treatments, this value might represent an overall response to all of the treatments in the subject contact record.	conclusionResponseValue in the contact record
conclusion_response_type_txt	A summary value, category name, or other description for the response that is specified in the conclusion_response_txt column. For example, this value might be `Evaluating`, `No response`, or `Closed-Lost`.	conclusionResponseType in the contact record
object_type_cd	A string that identifies the type of resource that created the contact record. Currently, the only value in this column is `decision`.	objectType in the contact record
object_revision_id	The ID for the version of the object that created the contact record.	objectRevisionId in the contact record
object_url_txt	The URI for the decision that created the contact record (for example, `/decisions/flows/545783f1-4fb3-45aa-be7d-d9e5a13ed943`).	objectUri in the contact record
exclude_from_contact_rule_txt	Specifies whether the contact record is included in aggregate reports for the channel. Column values are `true` or `false`.	excludeFromContactRule in the contact record
rule_fired_txt	Rule-fired information for the decision if that information was recorded by the record contacts node.	ruleFired in the contact record

Column	Description	Schema Property
traversed_rule_path_list_txt	Path-tracking information for the decision if that information was recorded by the record contacts node.	pathTraversed in the contact record
treatment_receiver_id	A string that identifies the entity that received the treatment. If this entity is the subject, then the treatment_receiver_id column contains the same value as the subject_id column. **Note:** This column is present in the table, but it is not currently used.	receiverId in the contact record
treatment_receiver_role_txt	A string that identifies the role of the entity that received the treatment. For example, the receiver might be the subject or an agent. **Note:** This column is present in the table, but it is not currently used.	receiverRole in the contact record
treatment_revision_id	The ID of the treatment definition version.	treatmentRevisionId in the treatmentForConsideration table
treatment_id	The ID of the treatment definition.	treatmentId in thetreatmentForConsideration table
presented_txt	Specifies whether the treatment was presented to the subject. The values in this column are a character string representation of a Boolean value: `true` or `false`.	presented in the treatmentForConsideration table
presentation_dttm	A timestamp that indicates when the treatment was presented to the subject. The timestamp format is Coordinated Universal Time (UTC): `yyyy-mm-ddThh:mm:ss.sssZ`	presentedTimeStamp in the treatmentForConsideration table
response_dttm	A UTC timestamp that indicates when the subject responded to the treatment.	respondedTimeStamp in the treatmentForConsideration table
response_txt	The subject's response to the treatment. For example, this value might be a value or code such as `Purchased`, `Rejected`, `Requested quote`, `Email opened`, or `rt1`.	responseValue in the treatmentForConsideration table
response_type_txt	A summary value, category name, or other description for the response that is specified by the value of the response_txt column. For example, this value might be `Positive`, `Negative`, or `Neutral`.	responseType in the treatmentForConsideration table

Column	Description	Schema Property
response_channel_txt	The channel through which the user responded to the treatment. The response channel might be different from the contact channel. For more information, see "About Channels" on page 90.	responseChannel in the treatmentForConsideration table
treatment_group_id	The ID of the treatment group that contains the treatment.	treatmentGroupId in the treatmentForConsideration table
consider_id	The primary key of the dcm_treatment_consideration table.	An internal ID for each treatment that was sent to a subject. This ID is unique within the view. This column is in not in the schema.
treatment_group_revision_id	The ID of the treatment definition group version.	treatmentGroupRevisionId in the treatmentForConsideration table
object_node_id	The ID of the node in the decision whose URL is specified in the object_url_txt column. For example, if the same treatment group is included in a decision multiple times, this ID enables you to determine which node generated an offer.	objectNodeId in the treatmentForConsideration table

Examples

Example 1: Determine Whether Any Treatments Were Sent to a Subject

The following query determines whether any treatments were sent to the subject with the ID 98.

```
select count(distinct contact_id)

from subjectcontacts.dcm_treatment_contacts_vw
where subject_id='98';
```

Example 2: Determine Whether a Treatment was Sent to a Subject within a Specific Time Frame

The following query determines whether the specified treatment (the treatment with the ID fcdfabaf-9932-430e-8cf3-149b212881d7) was sent to the subject with the ID 98 between March 1, 2022 and March 31, 2022.

```
select count(distinct contact_id)

from subjectcontacts.dcm_treatment_contacts_vw
where creation_dttm >='2022-03-01 00:00:00.000'
and creation_dttm <= '2022-03-31 23:59:59.999'
and subject_id='98'
and treatment_ID = 'fcdfabaf-9932-430e-8cf3-149b212881d7';
```

Example 3: Determine Whether a Treatment with a Specific Attribute Value Was Sent to a Subject

The following query determines whether the specified treatment was sent to subject 98 at a price of 50.00 euros between January 1, 2022 and March 31, 2022. The price is a dynamic attribute that has been assigned a static value in the treatment group, so the query joins the dcm_treatment_contacts_vs view and the dcm_trt_dynamic_attr_vw view.

```
select count(distinct contact_id)

from subjectcontacts.dcm_treatment_contacts_vw as sub,
treatmentdefinitions.dcm_trt_dynamic_attr_vw as treat
where sub.treatment_id = treat.treatment_id
and sub.subject_id='98' and sub.creation_dttm >='2022-01-01
00:00:00.000'

and sub.creation_dttm <= '2022-03-31 23:59:59.999'
and treat.treatment_ID = 'fcdfabaf-9932-430e-8cf3-149b212881d7'
and treat.attribute_nm='price' and treat.attribute_val='50.00';
```

Example 4: Determine Whether a Treatment Was Presented to a Subject within a Specified Time Frame

The presence of a timestamp in the presentation_dttm column indicates that the treatment was presented to the subject. The following query determines whether the specified treatment was presented to subject 98 between May 15, 2022 and June 30, 2022.

```
select count(distinct contact_id)

from subjectcontacts.dcm_treatment_contacts_vw
where creation_dttm >='2022-05-15 00:00:00.000'
and creation_dttm <= '2022-06-30 23:59:59.999'
and subject_id='98'
and treatment_ID = 'fcdfabaf-9932-430e-8cf3-149b212881d7'
and presentation_dttm is not null;
```

Example 5: Determine Whether a Subject Responded to a Treatment within a Specified Time Frame

The presence of a timestamp in the response_dttm column indicates that the subject responded to the treatment. The following query determines whether subject 98 responded to the specified treatment between May 15, 2022 and June 30, 2022.

```
select count(distinct contact_id)

from subjectcontacts.dcm_treatment_contacts_vw
where creation_dttm >='2022-05-15 00:00:00.000'
and creation_dttm <= '2022-06-30 23:59:59.999'
and subject_id='98'
and treatment_ID = 'fcdfabaf-9932-430e-8cf3-149b212881d7'
and response_dttm is not null;
```

Example 6: Determine Whether a Subject Responded to a Treatment with a Specific Attribute Value within a Specified Time Frame

The following query determines whether subject 98 responded to the specified treatment at a price of 100.00 euros between January 1, 2022 and March 15, 2022. The price is a dynamic attribute that has been assigned a static value in the treatment group, so the query joins the dcm_treatment_contacts_vs view and the dcm_trt_dynamic_attr_vw view.

```
select count(distinct contact_id)

from subjectcontacts.dcm_treatment_contacts_vw sub,
treatmentdefinitions.dcm_trt_dynamic_attr_vw treat
where creation_dttm >='2022-01-01 00:00:00.000'
and creation_dttm <= '2022-03-15 23:59:59.999'
and subject_id='98'
and treat.treatment_ID = 'fcdfabaf-9932-430e-8cf3-149b212881d7'
and attribute_nm='price' and attribute_val='100.00'
and response_dttm is not null;
```

Example 7: Determine Whether a Subject Responded to Any Treatment within a Specified Time Frame

The following query determines whether subject 98 responded to any treatment in the first six months of 2022.

```
select count(distinct contact_id)

from subjectcontacts.dcm_treatment_contacts_vw
where creation_dttm >='2022-01-01 00:00:00.000'
and creation_dttm <= '2022-06-30 23:59:59.999'
and subject_id='98' and response_dttm is not null;
```

Appendix 2

Import File Formats

Introduction 325
CSV Format of the Variable Import File 325
JSON Format of the Variable Import File 327

Introduction

You can import rule set and decision variables from either comma-delimited (CSV) files or JavaScript Object Notation (JSON) files. These files must conform to formats described in "CSV Format of the Variable Import File" on page 325 and "JSON Format of the Variable Import File" on page 327. For more information, see "Importing and Exporting Variables" on page 227.

CSV Format of the Variable Import File

Each row of the CSV input file contains the name of either a decision or rule set variable or of a data grid column if one of the decision or rule set variables is a data grid. The first line of the input file must be a header row. The CSV file must contain all of the columns listed in the following table, in the order listed. The value for some columns can be blank, as noted in the table. To create a blank column in the CSV file, specify two comma separators without any content between them.

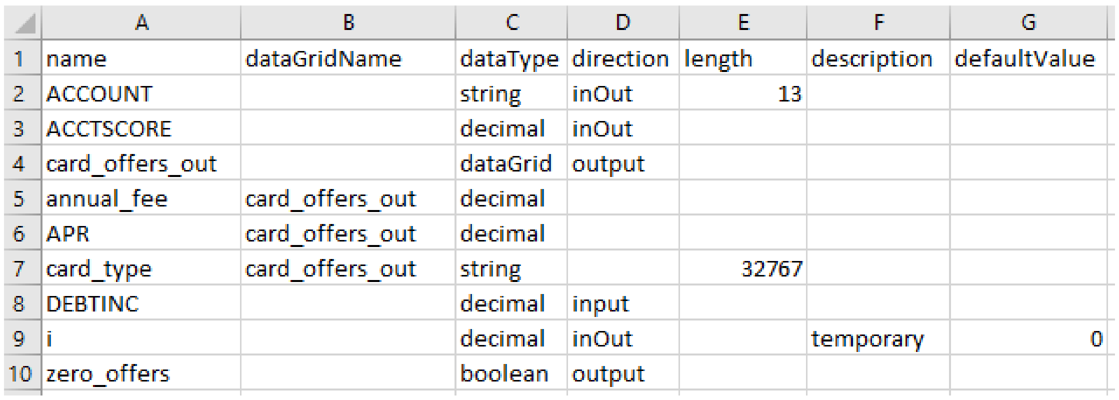
Table A3.1 CSV Format of the Variable Import File

Column	Description
name	The name of a rule set variable, a decision variable, or a data grid column. If one of the rule set or decision variables is a data grid variable, then specify the name of a column within the data grid.
dataGridName	If the name column contains the name of a data grid column, then the dataGridName column contains the name of the rule set or decision variable that contains the data grid.
dataType	The data type of the variable. You can specify `decimal`, `integer`, `date`, `datetime`, `boolean`, or `datagrid`. For character variables, specify `string`.
direction	Whether the rule set or decision variable is an input-only variable (`input`), an output-only variable (`output`), both (`inOut`). Do not specify a direction for data grid columns. If you do not specify a direction for a rule set or decision variable, the direction is imported as `inOut`. To designate a variable as a temporary variable, edit the variable properties on the **Variables** tab.
length	The length of the variable. Specify lengths for character variables only (strings or data grids).
description	A description of the variable. Descriptions are limited to 256 characters.
defaultValue	An initial (default) value for the variable.

The following example specifies nine variables. The variable card_offers_out is a data grid. The variables annual_fee, APR, and card_type are columns in the data grid. The variable i is a temporary variable that has an initial value of zero.

```
name,dataGridName,dataType,direction,length,description,defaultValue
ACCOUNT,,string,inOut,13,,
ACCTSCORE,,decimal,inOut,,,
card_offers_out,,dataGrid,output,,,
annual_fee,card_offers_out,decimal,,,,
APR,card_offers_out,decimal,,,,
card_type,card_offers_out,string,,32767,,
DEBTINC,,decimal,input,,,
i,,decimal,inOut,,temporary,0
zero_offers,,boolean,output,,,
```

In a spreadsheet application, this CSV file appears as shown in the following figure.



	A	B	C	D	E	F	G
1	name	dataGridName	dataType	direction	length	description	defaultValue
2	ACCOUNT		string	inOut	13		
3	ACCTSCORE		decimal	inOut			
4	card_offers_out		dataGrid	output			
5	annual_fee	card_offers_out	decimal				
6	APR	card_offers_out	decimal				
7	card_type	card_offers_out	string		32767		
8	DEBTINC		decimal	input			
9	i		decimal	inOut		temporary	0
10	zero_offers		boolean	output			

Note: Set the encoding value of the spreadsheet to the same value as the CSV file.

JSON Format of the Variable Import File

The JSON file contains a set of name-value pairs for each variable. The following table lists the fields that can be included for each variable.

Table A3.2 *Fields in the JSON Variable Input File*

Field	Description
name	The name of a rule set variable or a decision variable.
dataType	The data type of the variable. You can specify `decimal`, `integer`, `date`, `datetime`, or `datagrid`. For Boolean variables, specify `integer`. For character variables, specify `string`.
direction	Whether the rule set or decision variable is an input-only variable (`input`), an output-only variable (`output`), both (`inOut`). Do not specify a direction for data grid columns. If you do not specify a direction for a rule set or decision variable, the direction is imported as `inOut`. To designate a variable as a temporary variable, edit the variable properties on the **Variables** tab.
description	A description of the variable. Descriptions are limited to 256 characters.
length	The length of the variable. Specify lengths for character variables only (strings or data grids).
dataGridExtension	The columns in the data grid, if the variable specified in the name field is a data grid. Specify the name and data type for each column in the data grid. If the column is a character column, also specify the length.
defaultValue	An initial (default) value for the variable.

The following example specifies nine variables. The variable card_offers_out is a data grid. The variables annual_fee, APR, and card_type are columns in the data grid. The variable i is a temporary variable that has an initial value of zero.

```
[
    {
        "direction": "inOut",
        "name": "ACCOUNT",
        "dataType": "string",
        "length": 13
    },
    {
        "direction": "inOut",
        "name": "ACCTSCORE",
        "dataType": "decimal"
    },
    {
        "direction": "output",
        "name": "card_offers_out",
        "dataType": "dataGrid",
        "dataGridExtension": [
            {
                "name": "annual_fee",
                "dataType": "decimal"
            },
            {
                "name": "APR",
                "dataType": "decimal"
            },
            {
                "name": "card_type",
                "dataType": "string",
                "length": 32767
            }
        ]
    },
    {
        "direction": "inOut",
        "name": "DEBTINC",
        "dataType": "decimal"
    },
    {
        "direction": "inOut",
        "name": "i",
        "dataType": "decimal",
        "description": "temporary",
        "defaultValue": 0
    },
    {
        "direction": "output",
        "name": "zero_offers",
        "dataType": "boolean"
    }
]
```