Requirements and Tips for Using SAS Business Rules Manager Macros

- The SAS Business Rules Manager macros must be run on the server tier.
- If your SAS environment is in the locked-down state, your access to the file system of the host operating environment might be restricted. In order to run the SAS Business Rules Manager macros, your system administrator must enable the HTTP access method by specifying `ENABLE_AMS=HTTP` on the LOCKDOWN statement. For more information, see “Locked-Down Servers” in SAS Intelligence Platform: Security Administration Guide and “LOCKDOWN Statement” in SAS Intelligence Platform: Application Server Administration Guide.
- If folder administration is enabled, your system administrator must set `brm.import.restriction.override` to `true` in SAS Management Console in order for you to be able to use the import macros. For more information, see “Enable Business Rules Folder Administration” in SAS Business Rules Manager: Administrator’s Guide and “Business Rules Manager Web Advanced Properties” in SAS Business Rules Manager: Administrator’s Guide.
• If the value of a macro option contains a space, comma, forward slash (/), or other special characters, escape these characters by using a macro function such as the %STR function. For example, specify a full path name as %STR(/Users/user_ID/My Folder) or a set of rule flow identification numbers as %STR(10168,10043). For more information, see “%STR, %NRSTR Macro Functions” in SAS Macro Language: Reference.

• You can modify data values in exported CSV files, and then re-import the data. However, do not modify the CSV file structure (column or row order) or the CSV header row. **CAUTION:** If an input file contains errors, the database might become corrupted. Carefully review any changes you make to exported CSV files before you re-import the data.

• The same macro can be run simultaneously by multiple users. However, running import macros concurrently is not recommended.

---

### Macros Available with SAS Business Rules Manager

SAS Business Rules Manager macros are categorized by their functionality. Each macro belongs to one of the following categories:

- **Create terms**  
  Create and load new vocabulary terms into the SAS Decision Manager database.

- **Export**  
  Export content from the SAS Decision Manager database.

- **Import**  
  Import content into the SAS Decision Manager database.

- **Publish rule flows**  
  Publish rule flows to the content server.

- **Run rule flows**  
  Create DS2 package code for rule flows and run the rule flows.

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create terms</td>
<td>%BRM_CREATE_TEMP_TERM (p. 3)</td>
<td>Reads a CSV file or a SAS data set that defines vocabulary terms and produces a SAS data set named WORK.TERM. You can use the WORK.TERM data set as input to the %BRM_LOAD_VOCABULARY macro.</td>
</tr>
<tr>
<td></td>
<td>%BRM_LOAD_VOCABULARY (p. 21)</td>
<td>Loads the vocabulary terms that are defined in the WORK.TERM data set into the SAS Decision Manager database. You can create the WORK.TERM data set by using the %BRM_CREATE_TEMP_TERM macro.</td>
</tr>
<tr>
<td>Export</td>
<td>%BRM_EXPORT_FOLDER (p. 5)</td>
<td>Exports either the definition of a single business rules folder or the definitions all business rule folders into a CSV file. You can modify the CSV file and use it as input to the %BRM_IMPORT_FOLDER macro.</td>
</tr>
</tbody>
</table>
%BRM_CREATE_TEMP_TERM

Reads a CSV file or a SAS data set that defines vocabulary terms and produces a SAS data set named WORK.TERM. You can use the WORK.TERM data set as input to the %BRM_LOAD_VOCABULARY macro.
Category: Create terms

See: “Requirements and Tips for Using SAS Business Rules Manager Macros” on page 1

Syntax

\[
\%\text{BRM\_CREATE\_TEMP\_TERM} \text{(DATAFILE=\textit{input\_file}<, BRM\_USER=\textit{user\_ID}>)};
\]

Required Argument

\textbf{DATAFILE=\textit{input\_file}}

specifies either a SAS data set name or the full path name to a CSV file.

If the input file is a CSV file, the first row of the file must contain valid SAS column names, and the remaining rows must contain column values. The column values can be only numeric or character data. You cannot specify SAS informats in the column data. The column names must be unique. For example, a simple CSV file that specifies two columns, both with numeric data, might look like the following:

```
patientID,BloodPressure
1,140
2,141
3,142
```

Optional Argument

\textbf{BRM\_USER=\textit{user\_ID}}

specifies the user ID that you want to be associated with the data that is imported. This user ID is associated with the imported objects in the SAS Decision Manager database and is displayed in the interface.

\textbf{Default} User ID of the user that is logged on to the server and running the macro

Details

This macro reads a CSV file or SAS data set that defines vocabulary terms and creates a SAS data set named \texttt{WORK.TERM}. You can use the \texttt{WORK.TERM} data set as input to the \%\text{BRM\_LOAD\_VOCABULARY} macro. The \%\text{BRM\_LOAD\_VOCABULARY} macro loads the vocabulary terms into the SAS Decision Manager database. See “\%\text{BRM\_LOAD\_VOCABULARY}” on page 21 for more information.

The \%\text{BRM\_CREATE\_TEMP\_TERM} macro derives domain types and domain values for the vocabulary terms based on the data type of the term as described in \textit{Table 1.1}.

\textit{Table 1.1} Domain Types and Values for Input Terms

<table>
<thead>
<tr>
<th>Term Data Type</th>
<th>Derived Domain Type</th>
<th>Derived Domain Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>Discrete</td>
<td>If there are ten or fewer distinct values in the input data, all of the values are included in the list of domain values. If there are greater than ten distinct values in the input data, individual values are not listed in the domain values.</td>
</tr>
<tr>
<td>Term Data Type</td>
<td>Derived Domain Type</td>
<td>Derived Domain Values</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Date</td>
<td>Continuous</td>
<td>No input values are included in the list of domain values.</td>
</tr>
<tr>
<td>Datetime</td>
<td>Continuous</td>
<td>No input values are included in the list of domain values.</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
<td>True and False</td>
</tr>
<tr>
<td>Numeric</td>
<td>If there are ten or fewer distinct values in the input data, the domain type is Discrete. If there are greater than ten distinct values, the domain type is Continuous.</td>
<td>For Discrete domain types, all of the values in the input data are included in the list of domain values. For Continuous domain types, only the minimum and maximum values are included in the list of domain values.</td>
</tr>
</tbody>
</table>

%BRM_EXPORT_FOLDER

Exports either the definition of a single business rules folder or the definitions all business rule folders into a CSV file. You can modify the CSV file and use it as input to the %BRM_IMPORT_FOLDER macro.

**Category:** Export

**See:** “Requirements and Tips for Using SAS Business Rules Manager Macros” on page 1

**Syntax**

```sas
%BRM_EXPORT_FOLDER (CSV=basename.CSV <, FOLDER_PATH=folder_path>);
```

**Required Argument**

- **CSV=basename.CSV**
  - Specifies the full path name to the CSV file for the exported data.

**Optional Argument**

- **FOLDER_PATH=folder_path**
  - Specifies the full path name of the business rules folder that you want to export. Use a forward slash to separate folder names. By default, %BRM_EXPORT_FOLDER exports all business rules folders. If you specify a folder path name, then only that folder is exported.

**Example**

```sas
folder_path=%STR(Retail/ApprovedLoans)
```
%BRM_EXPORT_LOOKUP
Exports the contents of lookup tables into a CSV file. You can modify the CSV file and use it as input to the %BRM_IMPORT_LOOKUP macro.

Category: Export
See: “Requirements and Tips for Using SAS Business Rules Manager Macros” on page 1

Syntax
%BRM_EXPORT_LOOKUP (CSV=output_filename.CSV<, optional-arguments>);

Required Argument
CSV=output_filename
specifies the full path name to the CSV file for the exported data.

Optional Arguments
FOLDER_PATH=path_name
specifies the full path name to the business rules folder from which you want to export lookup tables. Use a forward slash to separate folder names.

If you specify a folder path name, then the lookup tables only in that folder are exported. For example, if you specify FOLDER_PATH=%STR(Loans/Retail), then the lookup tables only in the Loans/Retail folder are exported. If you specify both LOOKUP=CountryCodes,ZipCodes and FOLDER_PATH=%STR(Loans/Retail), but neither of the specified lookup tables are in the Loans/Retail folder, then no lookup tables are exported.

LOOKUP='lookup_table_1'<,'lookup_table_2'>...
specifies the names of the lookup tables that you want to export. Separate multiple names with commas.

By default, %BRM_EXPORT_LOOKUP exports all lookup tables. You do not need to specify the LOOKUP= option unless you want to export specific tables.

Example
lookup=%STR('BadVINSTates','StateCodes')

%BRM_EXPORT_RULE_FLOW
Exports rule flows from the SAS Decision Manager database into a CSV file. You can modify the CSV file and use it as input to the %BRM_IMPORT_RULE_FLOW macro.

Category: Export
See: “Requirements and Tips for Using SAS Business Rules Manager Macros” on page 1

Syntax
%BRM_EXPORT_RULE_FLOW (CSV=output_filename.CSV RULEFLOWS=ALL | rule_flow_1<,rule_flow_2>...<, FOLDER_PATH=path_name>);
Required Arguments

CSV=output_filename
specifies the full path name to the CSV file for the exported data.

RULEFLOWS=ALL | rule_flow_1<, rule_flow_2>...
specifies the rule flows that you want to export. Specify ALL to export all rule flows. To export only selected rule flows, specify the identification numbers of the rule flows. Separate multiple identification numbers with commas.

Example    ruleflows=%STR(10168,10043)

Optional Argument

FOLDER_PATH=path_name
specifies a business rules folder from which you want to export rule flows. Use a forward slash to separate folder names.

If you specify a folder path name, then the rule flows only in that folder are exported. For example, if you specify both RULEFLOWS=ALL and FOLDER_PATH=%STR(Loans/Retail), then the rule flows only in the folder Loans/Retail are exported. If you specify both RULEFLOWS=%STR(10045,10572) and FOLDER_PATH=%STR(Loans/Retail), but neither of the specified rule flows are in the Loans/Retail folder, then no rule flows are exported.

%BRM_EXPORT_RULESET
Exports rule sets from the SAS Decision Manager database into a CSV file. You can modify the CSV file and use it as input to the %BRM_IMPORT_RULESET macro.

Category: Export
See: “Requirements and Tips for Using SAS Business Rules Manager Macros” on page 1

Syntax

%BRM_EXPORT_RULESET (CSV=output_filename.CSV, RULESETS=ALL | rule_set_1<, rule_set_2>...<, FOLDER_PATH=path_name>);

Required Arguments

CSV=output_filename
specifies the full path name to the CSV file for the exported data.

RULESETS=ALL | rule_set_1<, rule_set_2>...
specifies the rule sets that you want to export. Specify ALL to export all rule sets. To export only selected rule sets, specify the identification numbers of the rule sets. Separate multiple identification numbers with commas.

Tip    To find the identification number for a rule flow, open the rule flow and click the Properties tab.

Example    rulesets=%STR(10168,10043)
Optional Argument

FOLDER_PATH=path_name
specifies the full path name for the business rules folder from which you want to export rule sets. Use a forward slash to separate folder names.

If you specify a folder path name, then the rule sets only in that folder are exported. For example, if you specify both RULESETS=ALL and FOLDER_PATH=%STR(Loans/Retail), then the rule sets only in the folder Loans/Retail are exported. If you specify both RULESETS=%STR(10045,10572) and FOLDER_PATH=%STR(Loans/Retail), but neither of the specified rule sets are in the Loans/Retail folder, then no rule sets are exported.

%BRM_EXPORT_VOCABULARY
Exports vocabularies from the SAS Decision Manager database into a CSV file. You can modify the CSV file and use it as input to the %BRM_IMPORT_VOCABULARY macro.

Category: Export

See: “Requirements and Tips for Using SAS Business Rules Manager Macros” on page 1

Syntax

%BRM_EXPORT_VOCABULARY (CSV=output_filename.CSV
VOCAB=ALL | vocabulary_1<, vocabulary_2>...<, FOLDER_PATH=path_name>);

Required Arguments

CSV=output_filename
specifies the full path name to the CSV file for the exported data.

VOCAB=ALL | vocabulary_1<, vocabulary_2>...
specifies the names of the vocabularies that you want to export. Specify ALL to export all vocabularies. To export only selected vocabularies, specify the names of the vocabularies, enclosed in quotation marks. Separate multiple names with commas.

Interaction To export only a specific list of vocabularies instead of all vocabularies, you must include the FOLDER_PATH= option.

Example vocab=%STR(LRAutoVocab,AcmeAuto)

Optional Argument

FOLDER_PATH=path_name
specifies the full path name of the business rules folder from which you want to export vocabularies. Use a forward slash to separate folder names.

If you specify a folder path name, then the vocabularies only in that folder are exported. For example, if you specify both VOCAB=ALL and FOLDER_PATH=%STR(Loans/Retail), then the vocabularies only in the folder Retail are exported. If you specify both VOCAB=%STR(loanVocab,riskVocabulary) and FOLDER_PATH=%STR(Loans/Retail), but neither of the specified vocabularies are in the Retail folder, then no vocabularies are exported.
%BRM_GET_RULE_FLOORE_CODE

Creates (but does not compile) a DS2 package that contains the SAS code for a specific rule flow. You can run this rule flow package by using the %BRM_RULE_FLOORE macro.

Category: Run rule flows

See: “Requirements and Tips for Using SAS Business Rules Manager Macros” on page 1

Syntax

%BRM_GET_RULE_FLOORE_CODE (RULEFLOW_NAME=name,
RULEFLOW_SK=number, FOLDER_PATH=path_name,
FILELOCATION=package_path_name<, optional-arguments>);

Required Arguments

RULEFLOW_NAME=name

specifies the name of the rule flow that you want to export.

Interaction If you specify both the RULEFLOW_NAME= and FOLDER_PATH= options, then you do not need to specify the RULEFLOW_SK= option.

Example ruleflow_name=Ruleflow1

RULEFLOW_SK=number

specifies the identification number of the rule flow. The identification number is shown in parentheses after the rule flow name on the rule flow History page or in the Properties section of the Results tab on the rule flow Tests page.

Interaction If you specify the RULEFLOW_SK= option, then you do not need to specify the RULEFLOW_NAME= or FOLDER_PATH= options.

Example ruleflow_sk=10014

FOLDER_PATH=path_name

specifies the full path name to the business rules folder in which the rule flow is defined. Separate folder names with forward slashes.

Interaction If you specify both the RULEFLOW_NAME= and FOLDER_PATH= options, then you do not need to specify the RULEFLOW_SK= option.

Example folder_path=%STR(Claims/Processing)

FILELOCATION=package_path_name

specifies the full path name to the file for the DS2 package that is produced by the macro. The path name must exist.

Example filelocation=%STR(C:\MgrApprovals\approvalFlow.sas)
**Optional Arguments**

**RULEFLOW_VERSION=version**

specifies the version of the rule flow to run. If you do not specify a version number, the macro retrieves the current version of the rule flow.

**SERVICETICKET=ticket_identifier**

specifies a central authentication service ticket to use for middle-tier authentication.

See “Central Authentication Service” in *SAS Intelligence Platform: Middle-Tier Administration Guide*

**USERNAME=user_ID**

specifies a user ID that has access to retrieve the rule flow. You must also use the **PASSWORD=password** option to specify the password for the user ID.

**PASSWORD=password**

specifies the password for the user specified with the **USERNAME=** option.

**WEBAUTHDOMAIN=domain**

specifies the authentication domain.

Default: The domain specified in metadata (DefaultAuth). The metadata entry for the domain must specify the user ID and password for the domain.

---

**%BRM_IMPORT_FOLDER**

Imports the folder definitions in the specified CSV file into the SAS Decision Manager database.

**Category:** Import

**See:** “Requirements and Tips for Using SAS Business Rules Manager Macros” on page 1

**Syntax**

```%BRM_IMPORT_FOLDER (CSV=input_filename.CSV, REJECT=reject_filename.CSV<, BRM_USER=user_ID>);```

**Required Arguments**

**CSV=input_filename**

specifies the full path name to the CSV file from which you want to import the data. For more information, see “Format of the Folder CSV Input File” on page 11.

**REJECT=reject_filename**

specifies the full path name to the CSV file to which you want the macro to write any records that were not imported to the SAS Decision Manager database. See “Using the %BRM_IMPORT_FOLDER Macro” on page 11 for more information.

**Optional Argument**

**BRM_USER=user_ID**

specifies the user ID that you want to be associated with the data that is imported. This user ID is associated with the imported objects in the SAS Decision Manager database and is displayed in the interface.
Details

Using the %BRM_IMPORT_FOLDER Macro
The %BRM_IMPORT_FOLDER macro enables you to create new folders. You cannot update the content in existing folders with this macro. The macro uses the path name to determine whether a folder already exists. If the path name already exists, then the folder is rejected.

The %BRM_IMPORT_FOLDER macro runs several validation checks as it imports the folders. For example, it checks whether each folder path name begins with a top-level folder and verifies that individual folder names are not longer than 100 characters. If the macro finds an invalid folder definition in the CSV file, it writes a message to the SAS log, and the folder is rejected. The macro writes the input records for the rejected folder to the CSV file that was specified in the REJECT= option.

Format of the Folder CSV Input File
Each row of the CSV input file identifies a folder. The CSV file must contain all of the columns listed in the following table, in the order listed. You must specify values for all columns, except as noted in the following table. To create a blank column in the CSV file, specify two comma separators without any content between them. For example, to create a folder named Applications and to specify a blank column for the folder description, specify the following in the CSV file:

```
Applications,,N,Loans/Retail
```

Table 1.2 Format of the Folder CSV Input File

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Can Column Be Blank</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOLDER_NM</td>
<td>The name of the folder to which you want to import the contents of the CSV file.</td>
<td>No</td>
</tr>
<tr>
<td>FOLDER_DESC</td>
<td>The description of the folder.</td>
<td>Yes</td>
</tr>
<tr>
<td>TOP_LEVEL_FOLDER_FLG</td>
<td>Specifies whether the folder is a top-level folder. Specify Y or N.</td>
<td>No</td>
</tr>
<tr>
<td>FOLDER_PATH</td>
<td>The path name to the business rules folder to which you want to import the contents of the CSV file. This path name must exist. Separate folder names with forward slashes.</td>
<td>No</td>
</tr>
</tbody>
</table>

%BRM_IMPORT_LOOKUP
Imports lookup tables from the specified CSV file into the SAS Decision Manager database.

Category: Import

See: “Requirements and Tips for Using SAS Business Rules Manager Macros” on page 1
Syntax

%BRM_IMPORT_LOOKUP (CSV=\input_filename.csv, REJECT=\reject_filename.csv<, BRM_USER=\user_ID>);

Required Arguments

CSV=\input_filename specifies the full path name to the CSV file from which you want to import the data. For more information, see “Format of the Lookup CSV Input File” on page 12.

REJECT=\reject_filename specifies the full path name to the CSV file to which you want the macro to write any records that were not imported to the SAS Decision Manager database. See “Using the %BRM_IMPORT_LOOKUP Macro” on page 12 for more information.

Optional Argument

BRM_USER=\user_ID specifies the user ID that you want to be associated with the data that is imported. This user ID is associated with the imported objects in the SAS Decision Manager database and is displayed in the interface.

Default User ID of the user that is logged on to the server and running the macro

Details

Using the %BRM_IMPORT_LOOKUP Macro

The %BRM_IMPORT_LOOKUP macro enables you to do the following tasks:

• add new lookup tables
• add new key-value pairs to existing lookup tables
• update (refresh) existing key-value pairs in existing lookup tables

The macro uses the lookup table name and path name to determine whether a lookup table already exists. If the lookup table already exists, then it is updated. If the path name exists but the lookup table does not exist, the lookup table is created. If the path name does not exist, then the lookup table is rejected.

The %BRM_IMPORT_LOOKUP macro runs several validation checks as it imports the lookup tables. For example, the macro checks whether the LOOKUP_NM or NAME columns in the input file are empty or whether the LOOKUP_NM column specifies an invalid lookup name. All valid key-value pairs are imported. If the macro finds an invalid key-value pair in the CSV file, it writes a message to the SAS log, and the key-value pair is rejected. The macro writes the input records for the rejected key-value pairs to the CSV file that was specified in the REJECT= option.

Format of the Lookup CSV Input File

Each row of the CSV input file identifies a key-value pair and the lookup table in which it belongs. The CSV file must contain all of the columns listed in the following table, in the order listed. You must specify values for all columns, except as noted in the table. 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. These key-value pairs will be imported into the lookup table Country_Codes.
FOLDER_PATH, LOOKUP_NM, DESCRIPTION, NAME, VALUE
Loans/Retail, Country_Codes,, AU, Australia
Loans/Retail, Country_Codes,, CA, Canada

This input file would appear in Microsoft Excel as shown in the following figure.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FOLDER_PATH</td>
<td>LOOKUP_NM</td>
<td>DESCRIPTION</td>
<td>NAME</td>
</tr>
<tr>
<td>2</td>
<td>Loans/Retail</td>
<td>Country_Codes</td>
<td></td>
<td>AU</td>
</tr>
<tr>
<td>3</td>
<td>Loans/Retail</td>
<td>Country_Codes</td>
<td></td>
<td>CA</td>
</tr>
</tbody>
</table>

Note: When you import a lookup table with the `%BRM_IMPORT_LOOKUP` macro, the first line of the input file must be a header row.

**Table 1.3 Format of the Lookup CSV Input File**

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Can Column Be Blank</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOLDER_PATH</td>
<td>The path name to the business rules folder to which you want to import the lookup table. This path name must exist. Separate folder names with forward slashes.</td>
<td>No</td>
</tr>
<tr>
<td>LOOKUP_NM</td>
<td>The name of the lookup table.</td>
<td>No</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>The description of the lookup table.</td>
<td>Yes</td>
</tr>
<tr>
<td>NAME</td>
<td>The lookup key.</td>
<td>No</td>
</tr>
<tr>
<td>VALUE</td>
<td>The lookup value.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**%BRM_IMPORT_RULE_FLOW**

Imports rule flows from the specified CSV file into the SAS Decision Manager database.

**Category:** Import

**See:** “Requirements and Tips for Using SAS Business Rules Manager Macros” on page 1

**Syntax**

```sas
%BRM_IMPORT_RULE_FLOW (CSV=input_filename.CSV, REJECT=reject_filename.CSV<, optional-arguments>);
```

**Required Arguments**

**CSV=input_filename**

specifies the full path name to the CSV file from which you want to import the data. For more information, see “Format of the Rule Flow CSV Input File” on page 14.
REJECT=reject_filename
specifies the full path name to the CSV file to which you want the macro to write any
records that were not imported to the SAS Decision Manager database. See “Using
the %BRM_IMPORT_RULEFLOW Macro” on page 14 for more information.

Optional Arguments

BRM_USER=user_ID
specifies the user ID that you want to be associated with the data that is imported.
This user ID is associated with the imported objects in the SAS Decision Manager
database and is displayed in the interface.

Default: User ID of the user that is logged on to the server and running the macro

OVERWRITE=Y|N
specifies whether existing rule flows can be updated. If you specify N, the updates
are rejected.

Details

Using the %BRM_IMPORT_RULEFLOW Macro
The %BRM_IMPORT_RULEFLOW macro enables you to add new rule flows and to
update existing rule flows. The macro uses the rule flow name and path name to
determine whether a rule flow already exists. If the rule flow name and path name
already exist, then the rule flow is updated (unless OVERWRITE=N is specified when
the macro is invoked). If the rule flow path name exists but the rule flow name does not
exist, the rule flow is created. If the rule flow path name does not exist, then the rule
flow is rejected.

The %BRM_IMPORT_RULEFLOW macro runs several validation checks as it
imports the rule flows. For example, it checks whether a rule set is referenced in a given
rule flow more than once and whether section codes are correct. If the macro finds a
validation error in a rule flow, it writes a message to the SAS log, and the rule flow is
rejected. The macro writes the input records for the rejected rule flow to the CSV file
that was specified in the REJECT= option.

Format of the Rule Flow CSV Input File
Each row of the CSV input file identifies a rule set, and a rule flow provides the
information about how that rule set fits into the rule flow. The CSV file must contain all
of the columns that are listed in the following table, in the order listed. You must specify
values for all columns, except as noted in the table. To create a blank column in the CSV
file, specify two comma separators without any content between them.

For example, to add a rule set to position 1 in the main section of the rule flow named
assignRisk in the Retail/Loans folder, you can specify the following in the CSV file:

.,assignRisk,,Y,main,Y,Loans/Retail,RuleSet1,Loans/Retail,Loan_Vocab,,1

Table 1.4 Format of the Rule Flow CSV Input File

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Can Column Be Blank</th>
</tr>
</thead>
<tbody>
<tr>
<td>RULE_FLOW_SK</td>
<td>The identification number of the rule flow.</td>
<td>Yes</td>
</tr>
<tr>
<td>Column</td>
<td>Description</td>
<td>Can Column Be Blank</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>RULE_FLOW_NM</td>
<td>The name of the rule flow to which you want to add the rule set that is specified in RULE_SET_NM.</td>
<td>No</td>
</tr>
<tr>
<td>RULE_FLOW_SHORT_DESC</td>
<td>The description of the rule flow.</td>
<td>Yes</td>
</tr>
<tr>
<td>RULE_FIRED_OUTPUT_FLG</td>
<td>Specifies whether to create output only for records that fire rules. Specify Y or N. For some types of applications, only the output records for which at least one rule has fired are of interest. Limiting output is useful for applications that detect outliers, such as applications that detect fraud.</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If the input data for the rule set is a snowflake table or a Google BigQuery table, rule-fired data cannot be generated for the rule set. See “How Rules Are Evaluated and When Rule-Fired Records Are Generated” in SAS Business Rules Manager: User’s Guide for more information.</td>
<td></td>
</tr>
<tr>
<td>RULE_SET_SECTION_CODE</td>
<td>The section of the rule flow to which the rule set that is specified in RULE_SET_NM belongs. Specify init, groupstart, main, groupend, or final. The codes groupstart and groupend are valid only if you also specify at least one term for BY_TERM. See “Simple Rule Flows, Complex Rule Flows, and BY Groups” in SAS Business Rules Manager: User’s Guide for more information.</td>
<td>No</td>
</tr>
<tr>
<td>INCLUDE_NODE_OBJECT_FLG</td>
<td>Specifies whether the rule set specified in the RULE_SET_NM field is run when the rule flow executes. Specify Y or N. Selectively running certain rule sets is useful during rule flow development and testing.</td>
<td>No</td>
</tr>
<tr>
<td>RULE_FLOW_PATH</td>
<td>The path name to the business rules folder for the rule flow. This path name must exist. Separate folder names with forward slashes.</td>
<td>No</td>
</tr>
<tr>
<td>RULE_SET_NM</td>
<td>The name of the rule set to be added to the rule flow. A rule set can be added to the same rule flow only once.</td>
<td>No</td>
</tr>
<tr>
<td>RULE_SET_PATH</td>
<td>The path name to the business rules folder for the rule set that is specified by RULE_SET_NM. The rule set must exist at the specified location. Separate folder names with forward slashes.</td>
<td>No</td>
</tr>
<tr>
<td>VOCAB_NM</td>
<td>The name of the vocabulary that the rule set uses. All rule sets in the same rule flow must use the same vocabulary.</td>
<td>No</td>
</tr>
<tr>
<td>BY_TERM</td>
<td>The list of BY-group terms that the rule set uses. Separate multiple BY-group terms with commas. The BY-group terms must be the same for all rule sets that are in the same rule flow. All of the BY-group terms must belong to the same vocabulary. See “Simple Rule Flows, Complex Rule Flows, and BY Groups” in SAS Business Rules Manager: User’s Guide for more information.</td>
<td>Yes</td>
</tr>
</tbody>
</table>
%BRM_IMPORT_RULESET

Imports rule sets from the specified CSV file into the SAS Decision Manager database.

**Category:** Import

**Requirement:**
The vocabulary that is used by a rule set must exist before you import the rule set.

**See:** "Requirements and Tips for Using SAS Business Rules Manager Macros" on page 1

**Syntax**

```
%BRM_IMPORT_RULESET (CSV=input_filename.CSV,
REJECT=reject_filename.CSV,<,optional-arguments>);
```

**Required Arguments**

**CSV=input_filename**
specifies the full path name to the CSV file from which you want to import the data. For more information, see “Format of Rule Set CSV Input File” on page 17.

**REJECT=reject_filename**
specifies the full path name of the CSV file to which you want the macro to write any records that were not imported to the SAS Decision Manager database. See “Using the %BRM_IMPORT_RULESET Macro” on page 17 for more information.

**Optional Arguments**

**BRM_USER=user_ID**
specifies the user ID that you want to be associated with the data that is imported. This user ID is associated with the imported objects in the SAS Decision Manager database and is displayed in the interface.

Default: User ID of the user that is logged on to the server and running the macro

**LOCK=Y|N**
specifies whether to lock the imported rule set.

Default: N

**NEWVERSIONS=Y|N**
specifies whether new, unlocked versions of each rule set are created with the imported content. If you specify Y, any existing unlocked versions of the rule sets are locked before the new unlocked version is imported. This option is useful when you are updating rule sets that are used in rule flows that have been published.
If you specify \texttt{N}, rule sets that are locked are not updated and are written to the reject file specified by the \texttt{CSV=} option.

Default \texttt{N}

\texttt{OVERWRITE=Y|N}

specifies whether existing rule sets can be updated. If you specify \texttt{N}, the updates are rejected.

Default \texttt{N}

Details

\textbf{Using the \texttt{\%BRM\_IMPORT\_RULESET} Macro}

The \texttt{\%BRM\_IMPORT\_RULESET} macro enables you to add new rule sets and to update existing rule sets. The macro uses the rule set name and rule set path name to determine whether a rule set already exists. If the rule set path name and name already exist, then the rule set is updated. If the rule set path name exists but the rule set name does not exist, the rule set is created. If the rule set path name does not exist, then the rule set is rejected.

The \texttt{\%BRM\_IMPORT\_RULESET} macro runs several validation checks as it imports the rule sets. For example, it verifies that the expressions are valid, ensures that the first rule in each rule set uses the IF operator, and verifies that the specified vocabularies exist. If the macro finds a validation error in a rule set, it writes a message to the SAS log, and the rule set is rejected. The macro writes the input records for the rejected rule set and the reason that the record was rejected to the CSV file that was specified in the \texttt{REJECT=} option.

Rule sets that you import with the \texttt{\%BRM\_IMPORT\_RULESET} macro are imported as unlocked versions. Before you can publish rule flows that contain the imported rule sets, you must lock the rule sets.

\textbf{Format of Rule Set CSV Input File}

Each row of the CSV input file specifies a rule, rule set, term, and an expression for that term. The row also specifies whether the expression is a condition expression or an action expression. Each row of the input file can specify only one condition expression or one action expression for a given rule. The CSV file must contain all of the columns that are listed in the following table, in the order listed. You must specify values for all columns, except as noted in the table. To create a blank column in the CSV file, specify two comma separators without any content between them.

For example, the following two lines add a rule to the rule set named riskSet, which uses the Loan_Vocab vocabulary. The first line adds the condition term \texttt{CondTerm} and assigns to it the expression \texttt{<5000}. The second line adds the action term \texttt{ActionTerm} and assigns to it the expression \texttt{'Bad'}.

\begin{verbatim}
,.riskSet,,Loan_Vocab,Loans/Retail,RuleName1,,1,IF,Y,CondTerm,<5000,1,CONDITION
,.riskSet,,Loan_Vocab,Loans/Retail,RuleName1,,1,IF,Y,ActionTerm,'Bad',1,ACTION
\end{verbatim}

\begin{table}[h]
\centering
\caption{Format of the Rule Set CSV Input File}
\begin{tabular}{|c|l|c|}
\hline
Column & Description & Can Column Be Blank \\
\hline
RULE\_SET\_SK & The identification number of the rule set. & Yes \\
\hline
\end{tabular}
\end{table}
<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Can Column Be Blank</th>
</tr>
</thead>
<tbody>
<tr>
<td>RULE_SET_NM</td>
<td>The name of the rule set to which you want to add the rule that is specified in RULE_NM.</td>
<td>No</td>
</tr>
<tr>
<td>RULE_SET_DESC</td>
<td>The description of the rule set.</td>
<td>Yes</td>
</tr>
<tr>
<td>VOCAB_NM</td>
<td>The name of the vocabulary that the rule set uses. All rules in the same rule set must use the same vocabulary.</td>
<td>No</td>
</tr>
<tr>
<td>RULE_SET_PATH</td>
<td>The path name to the business rules folder for the rule set. This path name must exist. Separate folder names with forward slashes.</td>
<td>No</td>
</tr>
<tr>
<td>RULE_NM</td>
<td>The name of the rule to be added to the rule set.</td>
<td>No</td>
</tr>
<tr>
<td>RULE_DESC</td>
<td>The description of the rule.</td>
<td>Yes</td>
</tr>
<tr>
<td>RULE_SEQ_NO</td>
<td>The order number for the rule that is in the rule set. Order numbers in a rule set start with 1.</td>
<td>No</td>
</tr>
<tr>
<td>CONDITIONAL_NM</td>
<td>The operator for the rule. Specify IF, ELSEIF, or OR. The first rule in a rule set must use the IF operator. For information about these operators, see “Controlling Which Conditions Are Evaluated” in SAS Business Rules Manager: User’s Guide.</td>
<td>No</td>
</tr>
<tr>
<td>RECORD_RULE_FIRED_FLG</td>
<td>Specifies whether a rule-fired record is created when the condition for the rule identified in the RULE_NM field evaluates to True. Specify Y or N. If you specify N, a rule-fired record is not created regardless of what the condition evaluates to. Note: If the input data for the rule set is a snowflake table or a Google BigQuery table, rule-fired data cannot be generated for the rule set. See “How Rules Are Evaluated and When Rule-Fired Records Are Generated” in SAS Business Rules Manager: User’s Guide for more information.</td>
<td>No</td>
</tr>
<tr>
<td>LHS_TERM</td>
<td>The term for the expression specified in the EXPRESSION column. Terms that are specified in the LHS_TERM column are the condition and action terms for the rule set. In the rule set editor, these terms appear in the Term column in the list view. They appear as column headings in the horizontal view and as row headings in the vertical view.</td>
<td>No</td>
</tr>
<tr>
<td>Column</td>
<td>Description</td>
<td>Can Column Be Blank</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>EXPRESSION</td>
<td>A single condition or action expression for the term specified in the LHS_TERM column. This expression is the expression that you would enter into a cell in the rule set editor. Enclose character strings in single quotation marks. To specify a missing expression, enter a period and an underscore (_). See “Defining New Rules in a Rule Set” in <em>SAS Business Rules Manager: User’s Guide</em> for more information about expressions.</td>
<td>Yes</td>
</tr>
<tr>
<td>EXPRESSION_ORDER</td>
<td>The order number of the rule’s condition or action expressions. A rule’s condition and action expressions are numbered beginning with 1. For example, a rule might have two condition expressions that are numbered 1 and 2, and it might have three action expressions that are numbered 1, 2, and 3.</td>
<td>No</td>
</tr>
<tr>
<td>EXPRESSION_TYPE</td>
<td>The type of expression. Specify <strong>CONDITION</strong> or <strong>ACTION</strong>.</td>
<td>No</td>
</tr>
</tbody>
</table>

%BRM_IMPORT_VOCABULARY

Imports vocabulary terms from the specified CSV file into the SAS Decision Manager database.

**Category:** Import

**See:** “Requirements and Tips for Using SAS Business Rules Manager Macros” on page 1

**Syntax**

```sas
%BRM_IMPORT_VOCABULARY (CSV= input_filename.CSV, REJECT= reject_filename.CSV<, BRM_USER= user_ID>);
```

**Required Arguments**

**CSV=** `input_filename`

specifies the full path name to the CSV file that defines the vocabulary that you want to import. For more information, see “Format of the Vocabulary CSV Input File” on page 20.

**REJECT=** `reject_filename`

specifies the full path name to the CSV file to which you want the macro to write any records that were not imported to the SAS Decision Manager database. See “Using the %BRM_IMPORT_VOCABULARY Macro” on page 20 for more information.
**Optional Argument**

**BRM_USER=user_ID**

specifies the user ID that you want to be associated with the data that is imported. This user ID is associated with the imported objects in the SAS Decision Manager database and is displayed in the interface.

Default: User ID of the user that is logged on to the server and running the macro

**Details**

**Using the %BRM_IMPORT_VOCABULARY Macro**

The %BRM_IMPORT_VOCABULARY macro enables you to add new vocabulary terms. You can also use this macro to update the description, domain type, and domain values for existing terms. You cannot use this macro to change the data type or name of an existing term.

The %BRM_IMPORT_VOCABULARY macro runs several validation checks as it imports the vocabulary terms. For example, it verifies that term, entity, and vocabulary names are valid, and ensures that a term is not duplicated in a vocabulary. If the macro finds a validation error, it writes a message to the SAS log, and the term is rejected. The macro writes the input records for the rejected term to the CSV file that was specified in the REJECT= option.

**Format of the Vocabulary CSV Input File**

Each row of the CSV input file defines a term, including the term data type, domain type, and the entity and vocabulary that contains the term. The CSV file must contain all of the columns listed in the following table, in the order listed. You must specify values for all columns, except as noted in the table. To create a blank column in the CSV file, specify two comma separators without any content between them.

For example, the following lines add two terms to the Loan_Vocab vocabulary. The first term is named Priority, and it is an integer with domain values in the range 1–10. The second term is named RiskCategory, and it is a character string with domain values 'Low' and 'High'.

```
Loan_Vocab,,AppEnt,,Priority,,Integer,discrete,(1-10),N,N,Loans/Retail
Loan_Vocab,,AppEnt,,RiskCategory,,Character,discrete,(Low;High),N,N,Loans/Retail
```

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Can Column Be Blank</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOCAB_NM</td>
<td>The name of the vocabulary to which you want to add entity and term specified by VOCAB_ENTITY_NM and VOCAB_TERM_NM.</td>
<td>No</td>
</tr>
<tr>
<td>VOCAB_SHORT_DESC</td>
<td>The description of the vocabulary.</td>
<td>Yes</td>
</tr>
<tr>
<td>VOCAB_ENTITY_NM</td>
<td>The name of the entity that the term in the VOCAB_TERM_NM column belongs to.</td>
<td>No</td>
</tr>
<tr>
<td>VOCAB_ENTITY_SHORT_DESC</td>
<td>The description of the entity.</td>
<td>Yes</td>
</tr>
<tr>
<td>Column</td>
<td>Description</td>
<td>Can Column Be Blank</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>VOCAB_TERM_NM</td>
<td>The name of the term.</td>
<td>No</td>
</tr>
<tr>
<td>VOCAB_TERM_SHORT_DESC</td>
<td>The description of the term.</td>
<td>Yes</td>
</tr>
<tr>
<td>VOCAB_TERM_DATA_TYPE_TXT</td>
<td>The data type of the term. Specify Character, Decimal, Integer, Boolean, Date, or Datetime.</td>
<td>No</td>
</tr>
<tr>
<td>VOCAB_TERM_DOMAIN_TYPE_TXT</td>
<td>The domain type for the term. Specify discrete, continuous, or Boolean. A domain value is discrete if it is just an individual value such as 5.3 or 18JUL2012:10:25:00. A domain value is continuous if it specifies a range such as &gt;5 or &lt;18JUL2012:10:25:00. Terms that are assigned the data type Character can have only discrete domain values. Boolean terms can have only Boolean domain values.</td>
<td>No</td>
</tr>
<tr>
<td>VOCAB_TERM_DOMAIN_TXT</td>
<td>The set of expected values for a term. Separate individual domain values with a semi-colon (;). See “Specify Domain Values” in SAS Business Rules Manager: User’s Guide for more information.</td>
<td>Yes</td>
</tr>
<tr>
<td>VOCAB_TERM_INPUT_EXCLUDE_FLG</td>
<td>Specifies whether the term must be mapped to a column in an input data set. Specify Y or N.</td>
<td>No</td>
</tr>
<tr>
<td>VOCAB_TERM_OUTPUT_EXCLUDE_FLG</td>
<td>Specifies whether to exclude the term from the output data sets created by rule flows. Specify Y or N.</td>
<td>No</td>
</tr>
<tr>
<td>FOLDER_PATH</td>
<td>The path name to the business rules folder for the rule flow. This path name must exist. Separate folder names with forward slashes.</td>
<td>No</td>
</tr>
</tbody>
</table>

**%BRM_LOAD_VOCABULARY**

Loads the vocabulary terms that are defined in the WORK.TERM data set into the SAS Decision Manager database. You can create the WORK.TERM data set by using the %BRM_CREATE_TEMP_TERM macro.

**Category:** Create terms  
**See:** “Requirements and Tips for Using SAS Business Rules Manager Macros” on page 1

**Syntax**

```sas
%BRM_LOAD_VOCABULARY (FOLDER_PATH=path_name,
VOCAB_NM=vocabulary_name, VOCAB_ENTITY_NM=entity_name
<, BRM_USER=user_ID>);
```
Required Arguments

FOLDER_PATH=path_name
specifies the path name of the business rules folder to which you want to import the vocabulary terms. Separate folder names with forward slashes.

Requirement The path name must exist. If the path name does not exist, the macro terminates and writes an error message to the SAS log.

Example folder_path=%STR(Loans/Retail/Applications)

VOCAB_NM=vocabulary_name
specifies the name of the vocabulary to which the terms in the WORK.TERM file will be added.

Requirement The vocabulary must not exist. If it already exists, the macro terminates and writes an error message to the SAS log.

VOCAB_ENTITY_NM=entity_name
specifies the name of the entity to which the terms in the WORK.TERM file will be added.

Requirement This entity must not exist. If it already exists, the macro terminates and writes an error message to the SAS log.

Optional Argument

BRM_USER=user_ID
specifies the user ID that you want to be associated with the data that is imported. This user ID is associated with the imported objects in the SAS Decision Manager database and is displayed in the interface.

Default User ID of the user that is logged on to the server and running the macro

%%BRM_PUBLISH_RULE_FLOW
Publishes a specific rule flow.

Category: Publish rule flows

See: “Requirements and Tips for Using SAS Business Rules Manager Macros” on page 1

Syntax

%%BRM_PUBLISH_RULE_FLOW (RULEFLOW_NAME=name, RULEFLOW_SK=number, FOLDER_PATH=path_name, METADATA_FOLDER=path_name);

Required Arguments

RULEFLOW_NAME=name
specifies the name of the rule flow that you want to publish.
Specify either the RULEFLOW_SK= option or both the RULEFLOW_NAME= and FOLDER_PATH= options.

Example

**RULEFLOW_SK=number**

specifies the identification number of the rule flow. The identification number is shown in parentheses after the rule flow name on the rule flow History page or in the Properties section of the Results tab on the rule flow Tests page.

Example

ruleflow_name=Ruleflow1

**FOLDER_PATH=path_name**

specifies the full path name to the business rules folder in which the rule flow is defined. Separate folder names with forward slashes.

Example

folder_path=%STR(Claims/Processing)

**METADATA_FOLDER=path_name**

specifies the path name to the folder that contains the metadata for the rule flow.

Example

metadata_folder=%STR(/Shared Data)

---

**%BRM_RULE_FLOW**

Runs rule flows. You can use the %BRM_RULE_FLOW macro to run packages that were created with the %BRM_GET_RULE_FLOW_CODE macro.

**Category:** Run rule flows

**See:** “Requirements and Tips for Using SAS Business Rules Manager Macros” on page 1

**Syntax**

FILENAME fileref "path_name";
%BRM_RULE_FLOW (INPUTTABLE=libref.table_name, MAPPING=mapfile.mapping, FILELOCATION=fileref, RULEFIRE=Y | N | S | D | Q<, optional-arguments>);

**Required Arguments**

**INPUTTABLE=libref.table_name**

specifies the libref and table name for the input table against which you want to run the rule flow.

**MAPPING=mapfile.mapping**

specifies the file that contains the variable mappings. This file is typically a SAS file. See “Creating a Mapping Table” on page 25.
FILELOCATION=fileref
specifies the fileref for the file that contains the DS2 package code for the rule flow. See SAS DATA Step Statements: Reference for information about the FILENAME statement and how to define filerefs.

RULEFIRE=Y | N | S | D | Q
specifies whether rule-fired data is recorded when the rule flow is run.

Y records both summary and detailed rule-fired data.
N does not record any rule-fired data.
S records only summary rule-fired data.
D records only detailed rule-fired data.
Q collects rule-fired data but does not generate summary or detailed rule-fired tables. The rule-fired data is added to the output table in columns named Rule Fired Count and _RULEFIREDCOUNTS_1.

Note If the input data for the rule set is a snowflake table or a Google BigQuery table, rule-fired data cannot be generated for the rule set. See “How Rules Are Evaluated and When Rule-Fired Records Are Generated” in SAS Business Rules Manager: User’s Guide for more information.

Optional Arguments

CODETYPE=DS1 | DS2
determines whether SAS Business Rules Manager generates DS2 code or DATA step (DS1) code for rule flows. In many cases, you will get better performance by specifying DS1. However, consider specifying DS2 if your input data is in Teradata, Greenplum, or Hadoop, and the SAS Code Accelerator is installed.

Default DS2

Note If the rule flows use data grids, you must specify CODETYPE=DS2.

THREADCOUNT=number
specifies the number of processors that are available for concurrent processing. If the rule flow contains rules in either the INIT or FINAL sections, the value of the THREADCOUNT option is set to 1 when rule flow tests are run. This option is ignored when rule flows are executed in the database.

Default the value of the CPUCOUNT= system option

Interaction This option is used only if CODETYPE=DS2.

See “CPUCOUNT= System Option” in SAS System Options: Reference

Details

Dynamically Running the Latest Rule Flow Version
You can use the &DCM_USE_LATEST_VERSION macro variable and either the &DCM_RULEFLOW_NAME or &DCM_DEPLOYED_RULEFLOW_NAME macro variable to ensure that when a rule flow is run, the latest version of the rule flow is always used. If you specify both &DCM_RULEFLOW_NAME and &DCM_DEPLOYED_RULEFLOW_NAME, then the name specified by &DCM_DEPLOYED_RULEFLOW_NAME is used.
For &DCM_DEPLOYED_RULEFLOW_NAME, specify the name of the published rule flow and the identification number of the rule flow. You can find the published name and identification number in the Name column of the rule flow History page. For example:

```sas
%let DCM_DEPLOYED_RULEFLOW_NAME=published_flow_name(ID_number);
```

**Note:** If you specify &DCM_RULEFLOW_NAME and SAS Business Rules Manager finds multiple rule flows that match the specified name, it writes an error message in the SAS log, and the rule flow is not executed. If you encounter this issue, specify the specific rule flow by using &DCM_DEPLOYED_RULEFLOW_NAME.

Define these macro variables in preprocessing code such as in the Preprocessing Code section of a rule flow test or in the Precode section on the Precode and Postcode tab in SAS Data Integration Studio. Define these variables before you call the %BRM_RULE_FLOW macro. For example:

```sas
%let DCM_USE_LATEST_VERSION=Y;
%let DCM_RULEFLOW_NAME=rule_flow_name;
```

**Note:** SAS Data Integration Studio uses the latest version of the rule flow that matches the variable mappings in the Business Rules transformation. SAS Business Rules Manager writes a note in the SAS log that states which version was selected.

**Using an ODBC Library**

If you are using an Open Database Connectivity (ODBC) library and you are using a schema that is different than the default user schema, specify the schema with the &BRM_DS2_SCHEMA_NAME_OVERRIDE macro variable. Define this macro variable in preprocessing code such as in the Preprocessing Code section of a rule flow test or in the Precode section on the Precode and Postcode tab in SAS Data Integration Studio. Define this variable before you call the %BRM_RULE_FLOW macro. For example:

```sas
%let BRM_DS2_SCHEMA_NAME_OVERRIDE=CUSTOMER_SCHEMA;
```

**Creating a Mapping Table**

**Note:** You must create a mapping table only if you are invoking the %BRM_RULE_FLOW macro in SAS code. In SAS Data Integration Studio and in the SAS Business Rules Manager test interface, the mapping table is created for you.

You must supply a file that maps terms in the rule flow to columns in the input table. You can create this file manually, or you can create and run a rule flow test in SAS Business Rules Manager. The mapping tables that are created when a rule flow test is run are written to the WORK library. The code that produces the mapping table is written to the SAS log.

The mapping table also defines the names and structure of the output table, the rule-fired summary table, the rule-fired details table, and the test information table that are generated by the rule flow. The structure of the rule-fired summary table, rule-fired details table, and test information table is static, and you must define them as shown in “Example: Creating a Mapping File for a Simple Rule Flow” on page 26.

The number in the data set ID column in the example specifies which table the column that is being defined belongs to. The following table lists the possible values for this column and the default table names that are generated when a rule flow is run in SAS Data Integration Studio.
Data Set ID | Table                | Contents                                                                 | Name Generated by SAS Data Integration Studio |
------------|----------------------|--------------------------------------------------------------------------|-----------------------------------------------|
1           | Rule-fired details   | One row for each time that a rule evaluates to true. There might be multiple entries for the same rule, but each entry has different values for the _recordCorrelationKey and RULE_ACTION_FIRE_ID columns. | DCM_RULE_ACTION_FIRE                         |
2           | Test information     | A single record that holds aggregate information about the execution of the rule flow. | DCM_DEPLOYMENT_EXECUTION                      |
4           | Input                | Input data                                                               |                                               |
5           | Output               | Output data                                                              |                                               |
6           | Rule-fired summary   | A summary of how many times each rule fired.                              | DCM_RULE_FIRE_SUMMARY                        |

Note: For more information about the columns in these tables, see Appendix 1, “Rule-Fired and Test Information Tables,” on page 29.

Example: Creating a Mapping File for a Simple Rule Flow

The following example creates a mapping table that maps terms in the rule flow to an input table with five columns. The column names are EngineSize, Make, Model, MSRP, and Type.

The example assumes that the following librefs have been defined: RULEFIRE, DEPLOY, INDATA, and OUTLIB. It uses the table names listed the following table.

<table>
<thead>
<tr>
<th>Data Set ID</th>
<th>Table</th>
<th>Libref and Table Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Rule-fired details</td>
<td>RULEFIRE.Details</td>
</tr>
<tr>
<td>2</td>
<td>Test information</td>
<td>DEPLOY.ThisRun</td>
</tr>
<tr>
<td>4</td>
<td>Input</td>
<td>INDATA.InData</td>
</tr>
<tr>
<td>5</td>
<td>Output</td>
<td>OUTLIB.MyResults</td>
</tr>
<tr>
<td>6</td>
<td>Rule-fired summary</td>
<td>RULEFIRE.Summary</td>
</tr>
</tbody>
</table>

data work.MAPPING;
  attrib table length = $100;
  attrib column length = $100;
  attrib termid length = $100;
  attrib type length = $100;
  attrib datasetid length = $100;
  attrib col_type length = $1;
  attrib col_length length = $5;
  attrib col_format length=$32;
  attrib col_informat length=$32;
call missing(of _all_);
stop;
run;
proc sql;
insert into work.MAPPING
values ('RULEFIRE.Details','RULE_ACTION_FIRE_ID','RULE_ACTION_FIRE_ID','output','1','C','100',''
values ('RULEFIRE.Details','RULE_SET_SK','RULE_SET_SK','output','1','N','8',''
values ('RULEFIRE.Details','RULE_SET_NM','RULE_SET_NM','output','1','C','100',''
values ('RULEFIRE.Details','RULE_SET_NM','RULE_SET_NM','output','1','C','100',''
values ('RULEFIRE.Details','RULE_FIRE_DTTM','RULE_FIRE_DTTM','output','1','N','8','nldatm.
values ('RULEFIRE.Details','DEPLMT_SK','DEPLMT_SK','output','1','N','8','nldatm.
values ('RULEFIRE.Details','DEPLMT_EXECUTION_ID','DEPLMT_EXECUTION_ID','output','1','C','100',''
values ('RULEFIRE.Details','TRANSACTION_MODE_CD','TRANSACTION_MODE_CD','output','2','C','20',''
values ('RULEFIRE.Summary','RULE_SK','RULE_SK','output','6','N','8',''
values ('RULEFIRE.Summary','RULE_NM','RULE_NM','output','6','C','100',''
values ('RULEFIRE.Summary','RULE_SET_SK','RULE_SET_SK','output','6','N','8',''
values ('RULEFIRE.Summary','RULE_SET_NM','RULE_SET_NM','output','6','C','100',''
values ('RULEFIRE.Summary','RULE_FLOW_SK','RULE_FLOW_SK','output','6','N','8',''
values ('RULEFIRE.Summary','RULE_FLOW_NM','RULE_FLOW_NM','output','6','C','100',''
values ('RULEFIRE.Summary','ruleFiredCount','ruleFiredCount','output','6','N','8',''
values ('OUTLIB.MyResults','EngineSize','EngineSize','output','5','N','8',''
values ('OUTLIB.MyResults','Make','Make','output','5','C','13',''
values ('OUTLIB.MyResults','Model','Model','output','5','C','40',''
values ('OUTLIB.MyResults','MSRP','MSRP','output','5','N','8',''
values ('OUTLIB.MyResults','Type','Type','output','5','C','8',''
};
quit;
Appendix 1
Rule-Fired and Test Information Tables

Overview

You can run a rule flow by using either the %BRM_RULE_FLOW macro or the Business Rules transformation in SAS Data Integration Studio. Running a rule flow generates three tables. These tables contain rule-fired information and information about the execution of the rule flow.

Table A1.1 Three Tables Generated by Running a Rule Flow

<table>
<thead>
<tr>
<th>Table</th>
<th>Contents</th>
<th>Name Generated By SAS Data Integration Studio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test information</td>
<td>A single record that contains aggregate information about the execution of the rule flow.</td>
<td>DCM_DEPLOYMENT_EXECUTION</td>
</tr>
<tr>
<td>Rule-fired details</td>
<td>One row for each time that a rule evaluates to true. There might be multiple entries for the same rule, but each entry has different values for the _recordCorrelationKey and RULE-ACTION_FIRE_ID columns.</td>
<td>DCM_RULE_ACTION_FIRE</td>
</tr>
<tr>
<td>Rule-fired summary</td>
<td>A summary of how many times each rule fired.</td>
<td>DCM_RULE_ACTION_FIRE</td>
</tr>
</tbody>
</table>

When the %BRM_RULE_FLOW macro is run outside of SAS Data Integration Studio, the names of the tables are controlled by the mapping file. See “%BRM_RULE_FLOW” on page 23 for information about the macro and the mapping file.
Columns in the Test Information Table

DEPLMT_EXECUTION_ID
the identification string of the specific instance of the rule flow that was executed. Each time a rule flow executes, a different universally unique identifier (UUID) is generated for the specific instance of the rule flow. You can use this UUID to associate the records in the test information table with the records in the rules-fired details table.

DEPLMT_SK
an internal surrogate key for the publish information for the rule flow. The publish information includes who published the rule flow, the version number that was published, and the location to which the rule flow was published. This column is also included in the rules-fired details table. You can use this column to join the two tables.

DEPLMT_NM
the name of the metadata object that was executed.

TRANSACTION_MODE_CD
always set to DIS.

Note: This column has been deprecated.

RECORDS_PROCESSED_NO
the number of records that were processed by the rule flow.

TEST_FLAG
indicates whether the rule flow was run in the SAS Business Rules Manager test interface.

START_DTTM
the date and time at which the rule flow started executing.

END_DTTM
the date and time at which the rule flow finished executing.

Columns in the Rule-Fired Details Table

RULE_ACTION_FIRE_ID
the UUID that is generated for each rule each time it is executed. Each time a rule executes, a different UUID is generated for the specific instance of the rule.

RULE_SET_SK
the identification number of the rule set.

RULE_SET_NM
the name of the rule set.

RULE_SK
the identification number of the rule.

RULE_NM
the name of the rule.
DEPLMT_SK
an internal surrogate key for the publish information for the rule flow. This column is also included in the test information table. You can use this column to join the two tables.

RULE_FLOW_SK
the identification number of the rule flow.

RULE_FLOW_NM
the name of the rule flow.

RULE_FIRE_DTTM
the date and time that the rule was run.

DEPLMT_EXECUTION_ID
the identification string of the specific instance of the rule flow that was executed. Each time a rule flow executes, a different UUID is generated for the specific instance of the rule flow. You can use this UUID to associate the records in the rules-fired details table with the records in the test information table on page 30.

ENTITY_PRIMARY_KEY
the value of the term that was specified with the &BRMPrimaryEntityKey macro variable in preprocessing code.

Note: This column has been deprecated. Use the _recordCorrelationKey column instead.

TRANSACTION_DTTM
the value of the term that was specified with the &BRMTransactionDTTM macro variable in preprocessing code.

Note: This column has been deprecated. Use the _recordCorrelationKey column instead.

_recordCorrelationKey
a UUID that enables you to associate records in the rules-fired details table (DCM_RULE_ACTION_FIRE) with records in the output results table. This column is also added to the output results table, so you can use this key to join the rules-fired details table and the output results table. Joining the tables enables you to enrich the information in the rules-fired details table with transaction times, composite keys, or other information.

Columns in the Rules-Fired Summary Table

RULE_SK
the identification number of the rule.

RULE_NM
the name of the rule.

RULE_SET_SK
the identification number of the rule set.

RULE_SET_NM
the name of the rule set.

RULE_FLOW_SK
the identification number of the rule flow.
RULE_FLOW_NM
the name of the rule flow.

ruleFiredCount
the number of times that the rule specified by the RULE_SK field executes for all of the input records that were processed.
Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.