IBM WebSphere ILOG JRules V7.1.1.3

SupportPac LB02 Version 2.0: WebSphere ILOG JRules Integration with the SPSS Predictive Analytics Suite

Copyright

Trademarks

Table of contents

Product overview

IBM WebSphere ILOG JRules Integration with the SPSS Predictive Analytics Suite helps you make better operational decisions using business rules and predictive analytics.

Traditionally, **predictive analytics** have been used in an offline mode to help people make better decisions – decisions that take into account insights about the possible future, based on analysis of historical data. Recently, there has been increasing interest in leveraging predictive models within the operational environment, at the point at which an automated or semi-automated decision is being made by people or by systems. This emerging discipline is often referred to as Decision Management.

Business rules are frequently used to express corporate policies and human expertise, and they are increasingly managed and deployed within operational systems using Business Rule Management Systems (BRMS).

Using business rules and predictive analytics together can lead to improved operational decisions. Predictive insight can be injected into business rules in a few different ways:

Business rules can be written incorporating the results of predictive model execution directly into the business rule text

For example, consider a business rule that dynamically computes product prices in realtime on an online retail travel website. Several regression models are run each week on the prior week's data, one for each product line, and the coefficients produced by these models are directly copied into the business rules that calculate the price. This results in business rules that take into consideration the most recent trends in customer behavior when computing the price.

Business rules can be generated from or written based on a predictive model

Examples of this are Decision Tree models and Association models, where the model itself, or selected parts of it, are naturally represented in the form of business rules.

Business rules can be written that reference predictive scores at runtime

This approach requires that the predictive models be deployed in the runtime environment and be available for execution by the BRMS when a score is needed.

This SupportPac provides two features that support the usage of business rules and Predictive Analytics together:

- ♦ PMML Import of Decision Tree models
- ♦ JRules runtime integration with the SPSS Scoring Service

PMML import of Decision Tree models

You can create a decision tree from a PMML Decision Tree model, for authoring and deployment in JRules. Decision Tree models are commonly used for predicting a variable, for example, the probability of a customer purchasing a particular product. The model is typically created by a statistical analyst.

You import a PMML TreeModel into a JRules decision tree to visualize and understand its content, and then modify, manage, and deploy the resulting artifact as you would any other decision tree. While a PMML TreeModel only generates a score, in JRules you can decorate the leaf nodes of the model with additional actions to take.

JRules also addresses cases not covered by the original Decision Tree model:

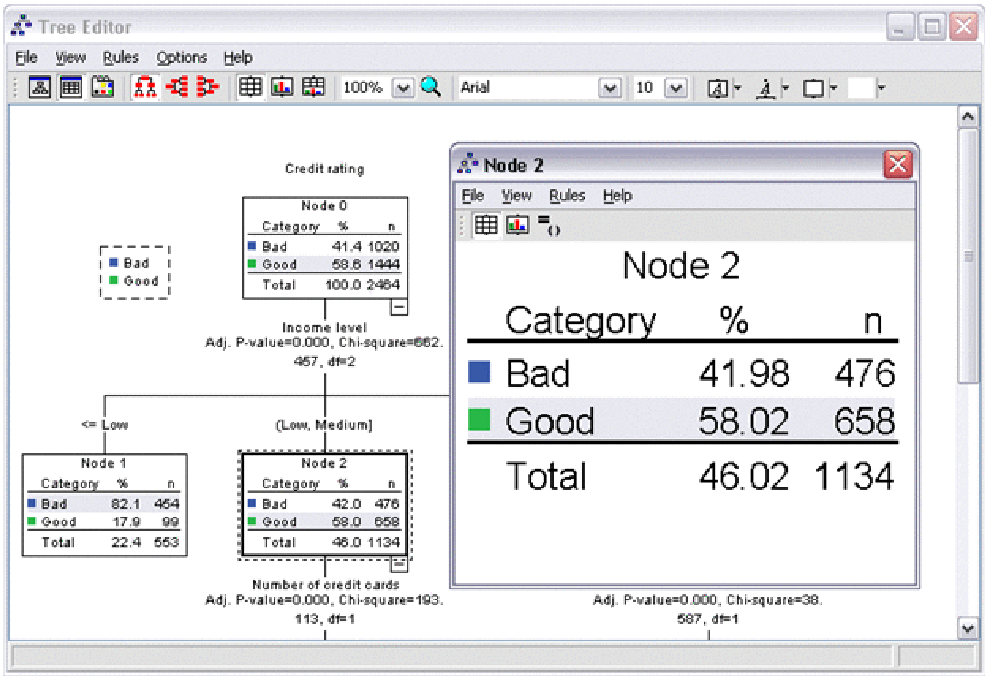
- Cases that were not indicated by the training data used by the data mining expert to create the PMML file.
- Cases where you want to experiment with different scores/decisions that are not generated by the TreeModel creation algorithm but make sense from a business perspective.

The sections that follow describe the interaction between the Decision Tree model, PMML TreeModel files, and JRules decision trees. Specific instructions on how to import PMML files in JRules are provided in *Importing PMML in JRules*.

Decision Tree models

Decision Tree models are produced by data mining algorithms (such as CHAID, C&RT, ID3, C4.5/C5.0) that identify various ways of splitting a dataset into branch-like segments, forming an inverted tree that starts with the root node at the top of the tree. Decision Tree models are used frequently in the data mining community for classification and prediction as they are easy to understand, easy to use, support both quantitative and qualitative measurements, and are very robust. Data mining workbenches, like SPSS Modeler, provide rich toolsets for creating and validating Decision Tree models.

The following image shows a Decision Tree model in SPSS Modeler:


Tree Editor
File View Rules Options Help
100%
Arial
10
Credit rating
Bad
Good
Income level
Adj. P-value=0.000, Chi-square=662.457, df=2
<= Low
(Low, Medium]
Number of credit cards
Adj. P-value=0.000, Chi-square=193.113, df=1
Adj. P-value=0.000, Chi-square=38.587, df=1
Node 2
File View Rules Help

Node 0

Category	%	n
🟦 Bad	41.4	1020
🟩 Good	58.6	1444
Total	100.0	2464

Node 1

Category	%	n
🟦 Bad	82.1	454
🟩 Good	17.9	99
Total	22.4	553

Node 2

Category	%	n
🟦 Bad	42.0	476
🟩 Good	58.0	658
Total	46.0	1134

Node 2

Category	%	n
🟦 Bad	41.98	476
🟩 Good	58.02	658
Total	46.02	1134

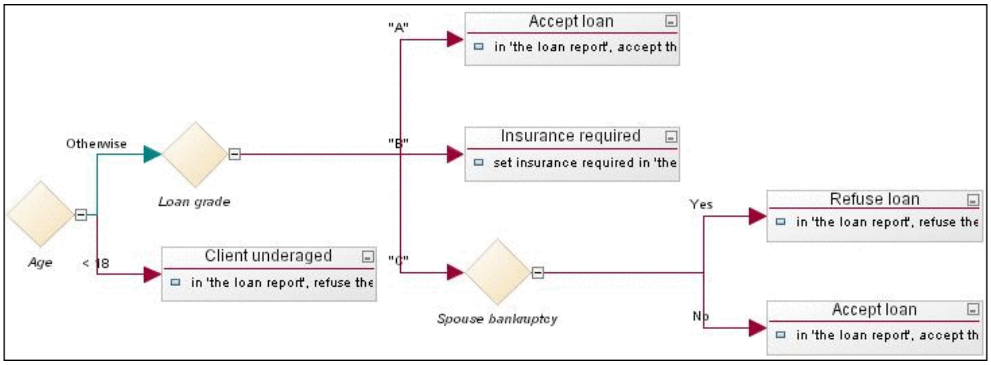
PMML files

PMML (Predictive Model Markup Language) is the leading standard for statistical and data mining models. It uses XML to represent mining models, so that models can be shared. In other words, using PMML, models can be developed on one system using one application and deployed on another system using a different application.

PMML is an open source project (see *http://sourceforge.net/projects/pmml/*). You can validate your PMML document on the DMG.org website (*http://www.dmg.org/converter*).

JRules decision trees

In addition to supporting the authoring, management, deployment, and execution of business rules as artifacts, JRules also supports decision tables and decision trees. Decision trees provide a convenient way of viewing and managing sets of business rules, especially when the rules are asymmetrical in structure. In a JRules decision tree, the path from the first condition (or root node) to the action (or leaf node) corresponds to one business rule.


"A"
Accept loan
in 'the loan report', accept th
Otherwise
"B"
Insurance required
set insurance required in 'the
Loan grade
Yes
Refuse loan
in 'the loan report', refuse the
Age
< 18
Client underaged
in 'the loan report', refuse the
"C"
Spouse bankruptcy
No
Accept loan
in 'the loan report', accept th

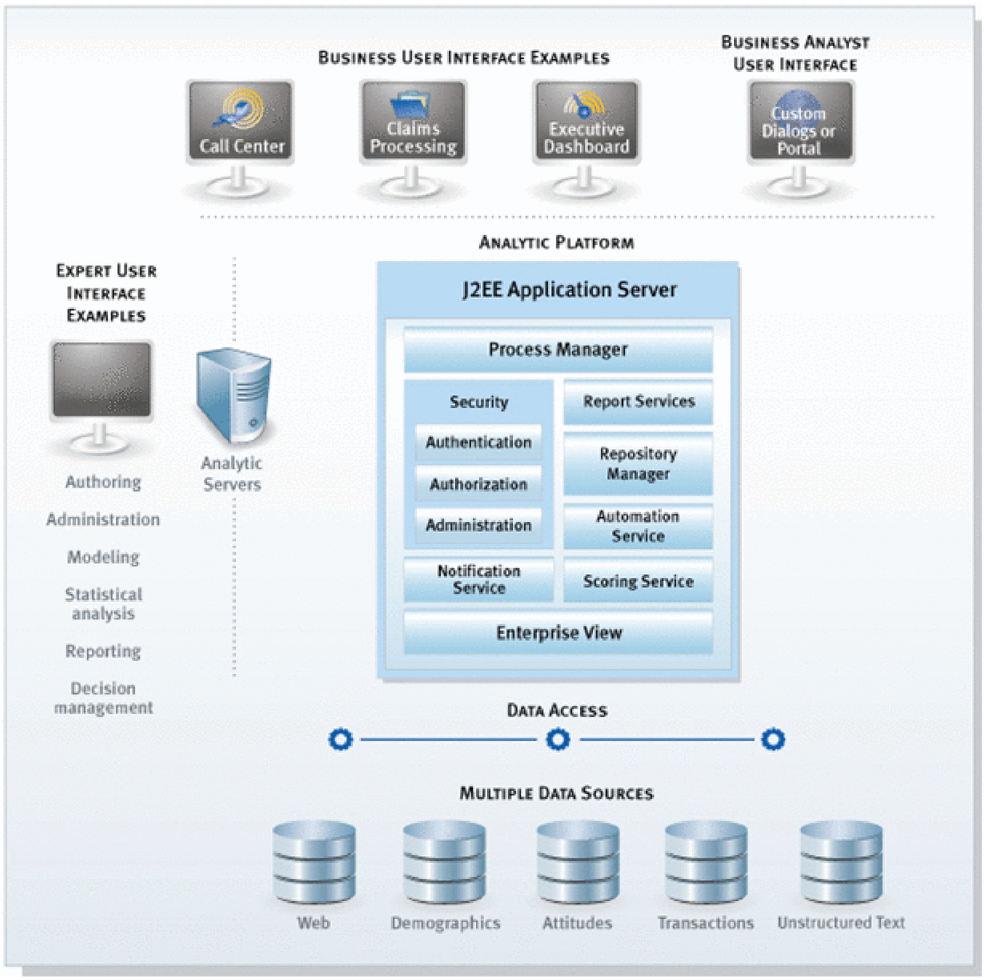
The primary difference between a Decision Tree model, as used in the data mining community, and a JRules decision tree is that the JRules decision tree has actions attached to the leaf nodes while the Decision Tree model usually has some sort of predicted variable or classification attribute specified for each node. In other words, a Decision Tree model can identify the business rules for classifying and predicting a specific variable, whereas the JRules decision tree can actually execute those business rules along with the appropriate actions at run time.

JRules runtime integration with the SPSS Scoring Service

This section describes JRules runtime integration with the SPSS Scoring Service. See *Invoking the SPSS Scoring Service* for specific instructions on how to invoke the scoring service.

The SPSS Scoring Service

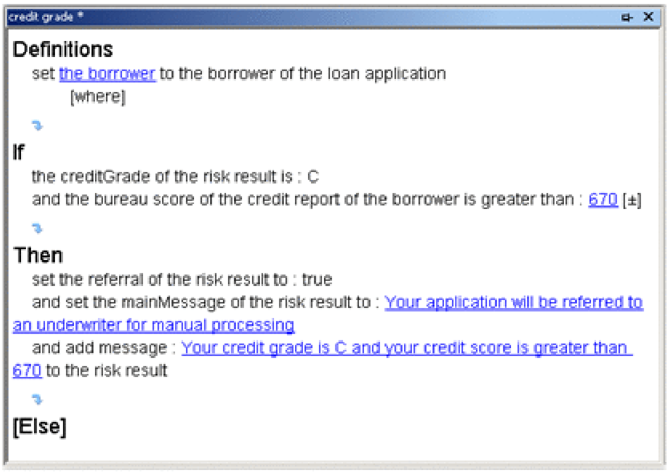
SPSS's Collaboration and Deployment Services (C&DS) product provides a platform for collaboration, management, and deployment of SPSS statistical and predictive models. Through its scoring service, C&DS can deploy predictive models as scores that can be used within key business processes to make better decisions. The scoring service is optimized for real time, and includes the ability to cache data and models and handle multiple sources of input data, including data gathered through real-time interactions. Using the scoring service, scoring may be applied at various points in business processes, and scores can easily be integrated with existing applications using web service calls.



Writing business rules that reference predictive scores

Leveraging the SPSS Scoring Service, JRules users can author business rules that reference predictive scores. These scores are then retrieved at runtime, when the business rules execute.

The following image shows a business rule referencing a predictive score:



credit grade *

Definitions

set the borrower to the borrower of the loan application
[where]

If

the creditGrade of the risk result is : C
and the bureau score of the credit report of the borrower is greater than : 670 [±]

Then

set the referral of the risk result to : true
and set the mainMessage of the risk result to : Your application will be referred to an underwriter for manual processing
and add message : Your credit grade is C and your credit score is greater than 670 to the risk result

[Else]

Installing

To correctly install and configure the SupportPac:

1. Verify that your environment corresponds to the supported platforms for the SupportPac.
2. Unzip the SupportPac deliverable in the JRules installation directory.
3. Install the predictive analytics features from Rule Studio.

Supported platforms

The SupportPac supports the following platforms:

- JRules 7.1.1.3
- JDK 6.0 or higher
- WebSphere Application Server 7.0, fixpack 11 or higher
- SPSS Collaboration and Deployment Services v 4.2
- Platforms: Windows, Linux.
- PMML version 3.2 and 4.0 TreeModels (with some limitations), as generated by SPSS PASW Modeler V13 or V14. See *PMML support* for more details on the limitations.

Unzipping the SupportPac deliverable

To begin your installation of the SupportPac, unzip `SupportPacLB02v2.0.zip` in the JRules 7.1.1.3 installation directory (`<JRulesInstallDir>`), typically `C:\Program Files\IBM\WebSphereILOGJRules711`.

Warning: The SupportPac requires JRules 7.1.1.3 or higher.

This adds `<JRulesInstallDir>/SupportPacs/SupportPacLB02/` in the distribution structure, with the following folders and files:

- `readme.html`: Readme file.
- `lib/`: Runtime JAR files required for execution.
  - `jrules-analytics-runtime.jar`
  - `commons-collections-3.1.jar`

    - `commons-lang-2.1.jar`
    - `velocity-1.5.jar`
- `license/`: License files.
- `eclipse/`
    - `features/`
    - `plugins/`
- `samples/`: Sample files.
    - `import`: PMML TreeModel sample files used in decision tree import:
        - `iris-tree-spss-3.2–chaid.xml`: Decision tree for the classification of iris trees, in PMML 3.2.
        - `samplePMML-3.2.xml`: Decision tree for determining whether to play tennis, according to temperature, wind, humidity, and so on, in PMML 3.2.
        - `newschan-spss-3.2.xml`: Prediction for a consumer to subscribe to a news channel, using a decision tree in PMML 3.2.
    - `scoring service`: PMML or SPSS Modeler Stream sample files to deploy to SPSS C&DS 4.2, for runtime calls:
        - `KMeans.xml`: Medical analysis clustering model using the KMeans method. Prediction of the patient's category according to some indicators in the blood analysis.
        - `Gender.xml`: Prediction of the gender of the patient, according to some indicators (age, blood pressure, cholesterol level).
        - `BP-Bayes.str`: SPSS Modeler 14.1 stream file, with a Bayesian network. Prediction of the blood pressure according to age, sex, and cholesterol level.

Installing the Predictive Analytics features

To install the SupportPac features in Eclipse, run the Eclipse Update Site:

1. In Rule Studio, click **Help** > **Software Updates**.
2. In the **Software Updates and Add-ons** dialog, select the **Available Software** tab.
3. Click **Add Site**, then click **Local**, and then browse to select the directory `<JRulesInstallDir>/SupportPacs/SupportPacLB02/eclipse`.
4. Click **OK**, and then **OK** again to add the new update site. It should now appear in your list of update sites.

5. Select **IBM WebSphere ILOG JRules integration for the SPSS Predictive Analytics Suite (LB02)**, and then click **Install**.
6. Confirm by clicking **Next** and accepting the license conditions, and then click **Finish**.
7. Click **Yes** to restart Eclipse.

To uninstall the SupportPac features:

1. In Rule Studio, click **Help** > **Software Updates**.
2. In the **Software Updates and Add-ons** dialog, select the **Installed Software** tab.
3. Select **IBM WebSphere ILOG JRules integration for the SPSS Predictive Analytics Suite (LB02)**, and click **Uninstall**.
4. Confirm and click **Finish**.
5. Click **Yes** to restart Eclipse.

Importing PMML in JRules

You import a PMML TreeModel into a JRules decision tree to visualize and understand its content, and then modify, manage, and deploy the resulting artifact as you would with any other decision tree.

In this section

Overview
After you export a Decision Tree model from a modeling tool to a PMML file, you can import it into a JRules decision tree.

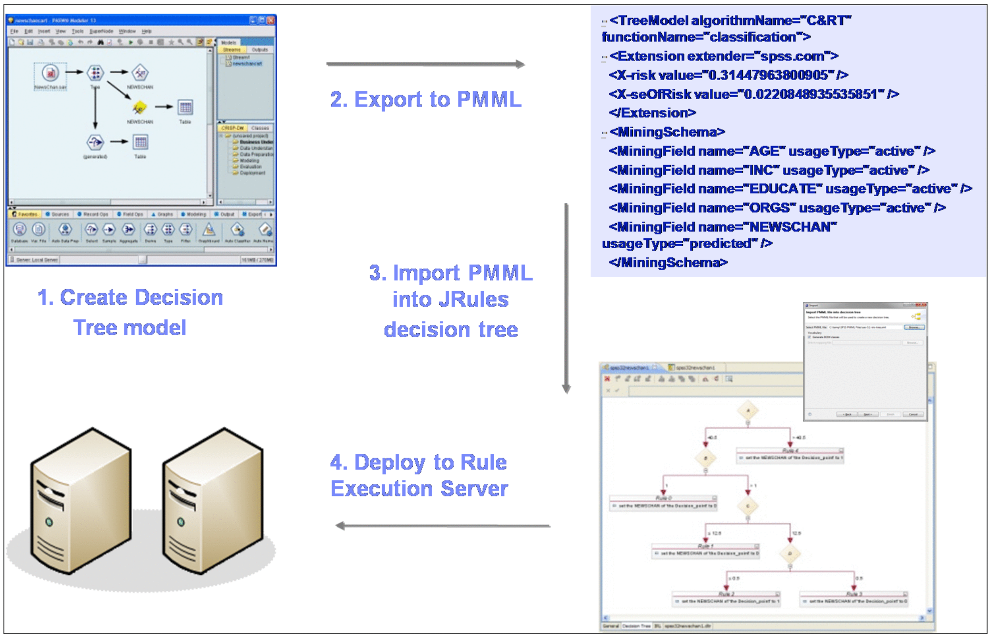
PMML support
PMML import in JRules can be done on files that meet certain requirements and limitations.

Importing a PMML file to a decision tree
You use the Import PMML Decision Tree wizard to import a PMML file into a decision tree.

Integrating the decision tree into the rule project
When importing a PMML file into a decision tree, create a XOM class corresponding to the BOM entry.

Overview

The JRules PMML import feature focuses on the **Decision Tree model**. After your Decision Tree model has been exported from a modeling tool to a PMML file, you can import it into a **JRules decision tree**. The following image shows the workflow:



1. Create Decision Tree model
2. Export to PMML
3. Import PMML into JRules decision tree
4. Deploy to Rule Execution Server

```
<TreeModel algorithmName="C&RT"
functionName="classification">
<Extension extender="spss.com">
  <X-risk value="0.31447963800905" />
  <X-seOfRisk value="0.0220848935535851" />
  </Extension>
<MiningSchema>
  <MiningField name="AGE" usageType="active" />
  <MiningField name="INC" usageType="active" />
  <MiningField name="EDUCATE" usageType="active" />
  <MiningField name="ORGS" usageType="active" />
  <MiningField name="NEWSCHAN"
usageType="predicted" />
  </MiningSchema>
```

Note: PMML supports a wide array of data mining models in addition to decision tree models. If you have other types of models that produce scores that you would like to leverage within your JRules application, you may want to invoke the scoring service instead, as described in *Invoking the SPSS Scoring Service*.

To correctly import a PMML file into a JRules decision tree:

- Make sure that your PMML file meets the requirements and limitations (see *PMML support*).
- Follow the step-by-step instructions to importing (see *Importing a PMML file to a decision tree*).
- Complete the integration of your decision tree into the rule project (see *Integrating the decision tree into the rule project*).

After your PMML TreeModel file has been successfully imported, it becomes a standard JRules decision tree artifact and can be edited, managed, and deployed like any other JRules decision tree. However, the import of PMML TreeModels containing compound or simple set predicates may result in irrelevant messages being produced by the overlap and gap checker in the JRules decision tree editor. To remedy this, turn off overlap and gap checking at either the node or tree level.

Note: An imported decision tree currently has no life-cycle link to the PMML file. Consequently, if you change the PMML model itself, you will have to repeat the import/modification process.

See also

Product overview

PMML support

The PMML import feature supports PMML 3.2 and 4.0 TreeModels generated by IBM SPSS Modeler V13 or V14. PMML 3.2 TreeModels generated by other tools or by previous versions of IBM SPSS Modeler may also work with this feature.

Tip: You can validate your PMML file on the DMG.org website (*www.dmg.org/converter*). This utility provides validation of PMML files, as well as conversion from older versions.

The PMML file must be a well-formed PMML 3.2 or 4.0 document consisting of at least the following elements:

- `Header`
- `DataDictionary`
- `TreeModel`
- `MiningSchema`: Must contain a `MiningField` element for each field referenced in the generated JRules decision tree. There must be one (and only one) field with a `FIELD-USAGE-TYPE` of `predicted`.
- `MiningField`
- `Node`: Each node in the PMML TreeModel must contain a value for the `Score` attribute.

Note: General information on PMML can be found here: *http://public.dhe.ibm.com/software/dw/industry/ind-PMML1/ind-PMML1-pdf.pdf*

The JRules PMML import feature does not execute PMML. Rather, it uses the PMML TreeModel as a basis to generate business rules in the form of a JRules decision tree artifact.

After the TreeModel has been imported, the generated JRules decision tree may be manually transformed by the user, if necessary, to produce a decision tree that makes sense within the context of the deployed business application. In other words, you can import a design time artifact that will be transformed into executable business rules that can be managed and deployed in the BRMS.

JRules decision trees are not currently able to persist all of the metadata contained in a PMML TreeModel. In the generated JRules decision tree, you will find:

- On each node: The name of the attribute being tested.

- On each branch: The test conditions being applied to the attribute in the node where the branch originates.
- On each leaf node: The value of the `Score` attribute from the corresponding leaf node in the PMML TreeModel.

Additional metadata that may be associated with the PMML TreeModel node and displayed in SPSS Modeler and other Data Mining tools will not be displayed in the JRules decision tree. The most common examples of this kind of metadata are: the Score Distribution value, the Score Distribution RecordCount, and the ScoreDistribution Confidence.

Assumptions

Many of the PMML elements that you may commonly encounter in a PMML TreeModel are not relevant in the proposed design-time approach, and will simply be ignored.

These elements are listed under the following sections:

- Data manipulation
- Unknown and missing values
- Presence of a value for the Score attribute

Data manipulation

The following elements are used to manipulate input fields, predicted values, and output fields. Since the import feature imports the PMML rather than executing it, these elements are not used during decision tree generation:

Data transformations

PMML import ignores data transformations (`TransformationDictionary`, `LocalTransformations`, `DerivedField`). This information can be useful to the JRules application developer when configuring the JRules BOM and XOM, to help ensure that the right data will be provided to JRules at runtime.

Target

Ignored by PMML import. They are typically used to manipulate the predicted value generated by a model.

Output

The additional output fields that might be generated when a PMML file is executed are not relevant when authoring a JRules decision tree. However, if these additional output fields are important to your application processing, you should consider executing your Decision Tree model using a Scoring Service, rather than transforming the model into a JRules decision tree for execution.

Unknown and missing values

There are a number of constructs within PMML that are used to specify how unknown and missing input values are handled during PMML execution of the TreeModel. These constructs are not supported when importing PMML TreeModels into JRules, because JRules does not support the notion of missing values. Rather, it is the responsibility of the JRules developer to ensure that the JRules data model is properly configured and that the necessary data is made available to the JRules Decision Service at runtime.

As a result, the following elements are handled in the following ways:

- `missingValueStrategy`: Ignored.
- `missingValuePenalty`: Ignored.
- `Surrogates`: Ignored. If the surrogate operator is used within a compound predicate, the first predicate in the sequence will be selected by default.
- `missingValueReplacement`: Ignored.
- `missingValueTreatment`: Ignored.

Presence of a value for the Score attribute

Since there will always be a `score` attribute for a given node, `noTrueChildStrategy` and the `ScoreDistribution` element are ignored.

Limitations

The following PMML constructs are not supported, ignored, or otherwise handled by PMML import:

- Vendor-specific extensions: Any occurrences of the PMML `Extension` element will be ignored during import.
- Multiple models: If there is more than one TreeModel in the PMML file, only the first one will be processed, and the rest will be ignored. If there are other (non-TreeModel) model elements in the PMML file, they will also be ignored. Model composition (`EmbeddedModel` element), and segmentation (`Segment` or `Segmentation` elements) are not supported. Also, the `MiningModel` element is not supported, so Decision Tree Ensembles cannot be processed.
- DataFields, Datatypes, and Optypes.
  - There are several PMML date formats that are simply treated as `int` datatypes (see *Reference* for a full listing).
  - Intervals: Continuous `Datafields` with defined intervals will be treated like regular continuous fields.

- Taxonomy: If a categorical `Datafield` references a Taxonomy, it will be treated like a categorical field without declared Values.
- Cyclic elements: The Cyclic designation will be ignored on continuous and ordinal DataFields. This should not pose a problem, as this element is used in Clustering models, not Decision Tree models).

◆ Operators

- The `LessThan` and `GreaterThan` operators on Ordinal Optypes are not supported.
- The boolean operator `XOR` is not supported.
- The boolean operator `Surrogate` is not supported.
- The `isMissing` and `isNotMissing` operators are not supported.

See also

Overview
Type support for PMML import
Importing a PMML file to a decision tree

Importing a PMML file to a decision tree

When you use PMML TreeModels in JRules decision trees, the JRules rule project must contain:

- ♦ A BOM class on which the PMML TreeModel elements are mapped. This class must have one attribute per PMML field and a default verbalization for each of these elements.
- ♦ An instance of the above BOM class as ruleset parameter. This ruleset parameter must also have a verbalization.

There are two possible cases to consider before importing:

The BOM class for the PMML TreeModel does not exist.
This typically occurs:

- ♦ The first time you import a PMML file into a given rule project.
- ♦ When the PMML fields of the file you want to import are completely different from the BOM class you generated in a previous import, so you need a new BOM class.

In either of these cases, you want the Import PMML Decision Tree wizard to generate both the BOM class and the corresponding ruleset parameter, define a default verbalization for them, and then create the decision tree using the default verbalizations.

In the case of subsequent imports requiring a different BOM class, the Import PMML Decision Tree wizard can generate a different BOM class type and verbalization, as well as a different ruleset parameter name and verbalization to avoid conflicts with PMML TreeModels that you previously imported.

The BOM class for the PMML TreeModel already exists.
If the PMML file you want to import has all or most of the fields already found in the BOM class, do not generate another BOM class and ruleset parameter. Instead, use the PMML Verbalization editor in the Import PMML Decision Tree Wizard to associate the PMML elements of the new file with the corresponding members of the existing BOM class. This is facilitated by the use of the mapping file.

After your PMML TreeModel file has been successfully imported, it becomes a standard JRules decision tree artifact and can be edited, managed, and deployed like any other JRules decision tree. An imported decision tree currently has no life-cycle link to the PMML file. Consequently, if you change the PMML model itself, you have to repeat the import/modification process.

Note: The import of PMML TreeModels containing compound or simple set predicates may result in irrelevant messages being produced by the overlap and gap checker in the

JRules decision tree editor. To remedy this, turn off overlap and gap checking at either the node or tree level.

Importing when the BOM class does not exist

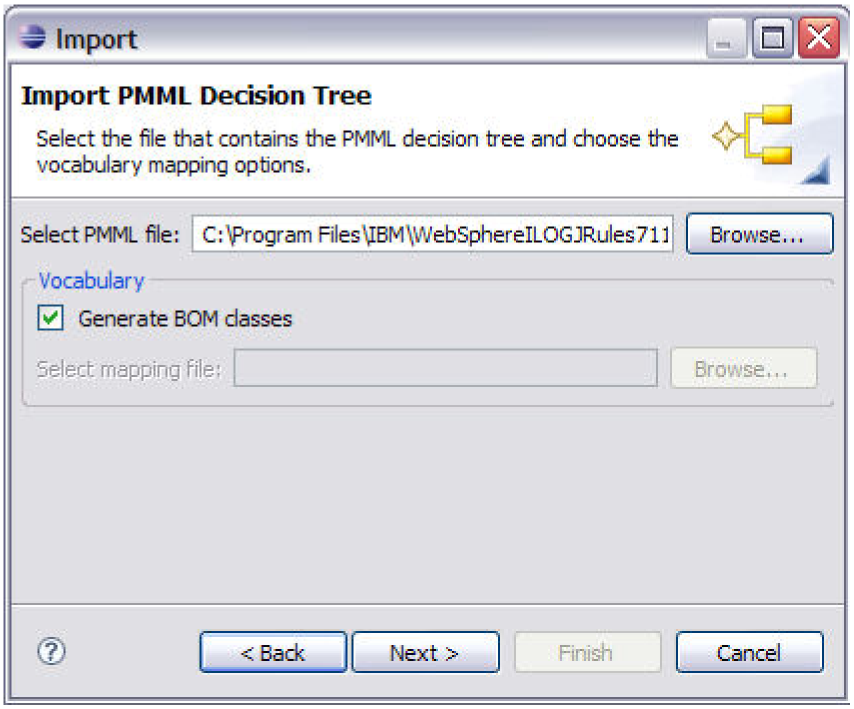
The first time you import a PMML TreeModel file into a given rule project, there is no corresponding BOM class and ruleset parameter. Similarly, when the PMML fields of the file you want to import are completely different from the BOM class you generated in a previous import, you need a new BOM class.

In either of these cases, let the Import PMML Decision Tree wizard generate a BOM class and ruleset parameter by keeping the **Generate BOM classes** option enabled.

When you keep this option enabled, the Import PMML Decision Tree wizard creates a BOM entry with a class containing all the members of the PMML file, creates a default verbalization for each member, creates an instance of this class as ruleset parameter with a default verbalization, and then uses this information to create the decision tree.

To import a PMML file as a decision tree when the required BOM class does not exist:

1. In Rule Studio, switch to the Rule perspective, and select the rule project in which you want to import the PMML file.
2. Click **File** > **Import**, select **Rule Studio** > **Predictive Analytics** > **PMML Decision Tree**, and click **Next**.
3. In the **Import PMML Decision Tree** page, enter or browse to a valid PMML file.


Import

Import PMML Decision Tree

Select the file that contains the PMML decision tree and choose the vocabulary mapping options.

Select PMML file: C:\Program Files\IBM\WebSphereILOGJRules711

Vocabulary

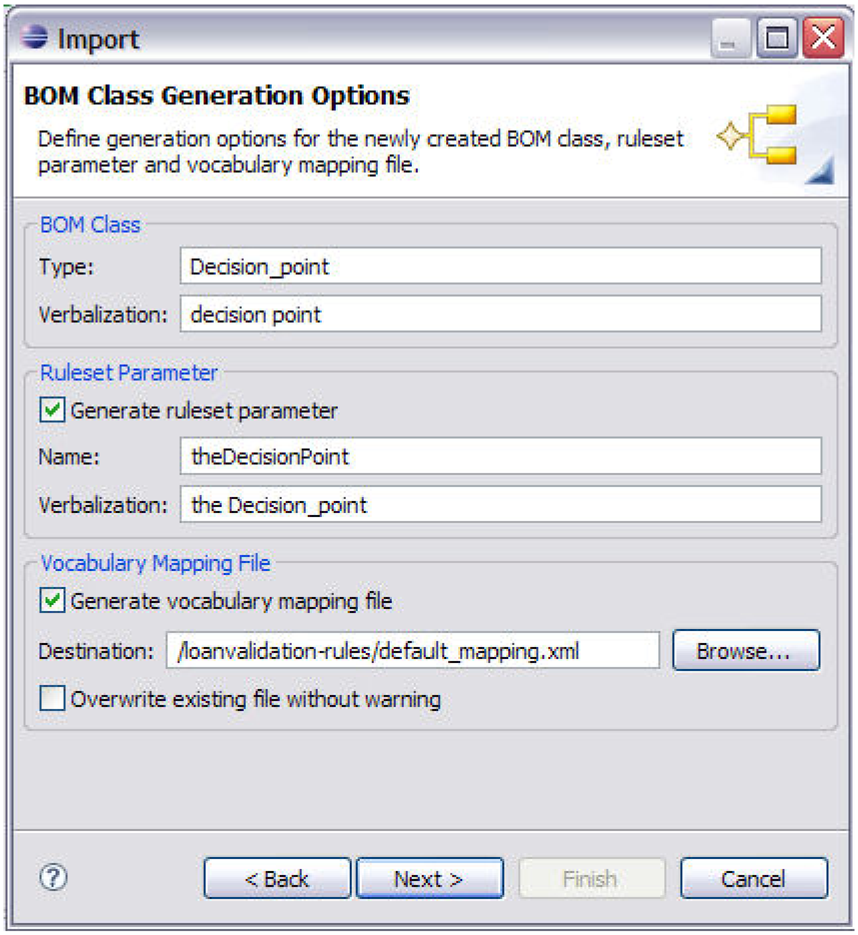
[✓] Generate BOM classes

Select mapping file:

Important: Make sure the PMML file conforms to the requirements and limitations described in *PMML support*.

4. Make sure that the **Generate BOM classes** option is enabled and click **Next**.

   The **BOM Class Generation Options** panel is displayed.



BOM Class Generation Options

Define generation options for the newly created BOM class, ruleset parameter and vocabulary mapping file.

BOM Class

Type: Decision_point

Verbalization: decision point

Ruleset Parameter

[✓] Generate ruleset parameter

Name: theDecisionPoint

Verbalization: the Decision_point

Vocabulary Mapping File

[✓] Generate vocabulary mapping file

Destination: /loanvalidation-rules/default_mapping.xml

[ ] Overwrite existing file without warning

The wizard proposes default values for the generated BOM class type and verbalization, and the generated ruleset parameter name and verbalization. You can edit these if needed.

5. Keep **Generate vocabulary mapping file** and click **Next**.

   With this option checked, the wizard also creates a vocabulary mapping file containing the default verbalization associated to each PMML element. This mapping file can later be reused when importing similar PMML TreeModel files.

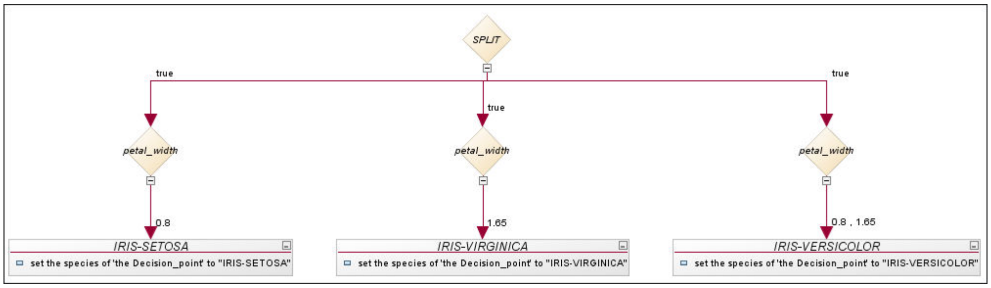
6. Enter the name and location for the new BOM entry in which to store the PMML members, and then click **Next**.

Note: The **folder** field defines in which bom sub-folder the BOM entry is created. A blank value means that the entry is generated directly under the bom top-level folder.

7. Enter the name and location of the new decision tree, and then click **Finish**.

Note: You can change the Type field if you have defined custom decision tree extension classes and wish to have the new decision tree instance inherit from one of these custom decision tree classes.

Your new decision tree is displayed:


SPLIT
true
true
true
petal_width
petal_width
petal_width
0.8
1.65
0.8 , 1.65
IRIS-SETOSA
set the species of 'the Decision_point' to "IRIS-SETOSA"
IRIS-VIRGINICA
set the species of 'the Decision_point' to "IRIS-VIRGINICA"
IRIS-VERSICOLOR
set the species of 'the Decision_point' to "IRIS-VERSICOLOR"

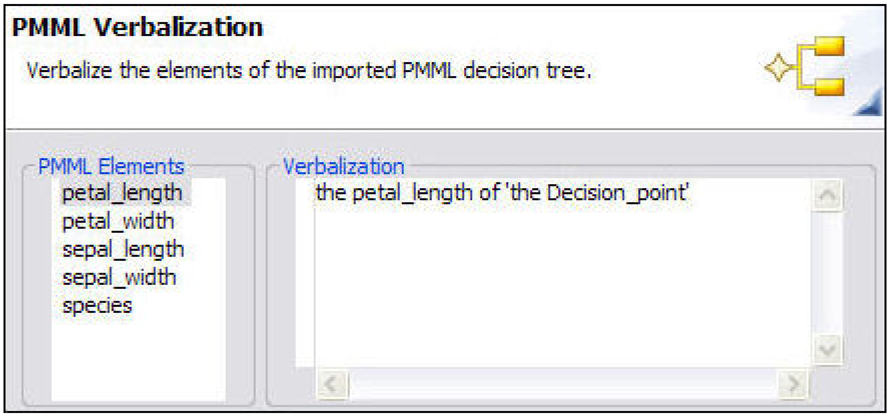
You may get B2X errors and warnings until you create the corresponding XOM class, as described in *Integrating the decision tree into the rule project*. Also, you can remove irrelevant overlap and gap checker messages generated by the decision tree editor by turning off overlap and gap checking at either the node or tree level.

Note: Other messages are described in section *Reference*.

Importing when the BOM class already exists

When a BOM class already contains all or most of the PMML TreeModel elements of the file you want to import, do not generate another BOM class. Otherwise you will get redundant entries in your drop-down lists.

Instead, uncheck the **Generate BOM classes** option in the import wizard and use the PMML Verbalization editor in the Import PMML Decision Tree Wizard to associate the PMML elements of the new file with the corresponding members of the existing BOM class:


PMML Verbalization

Verbalize the elements of the imported PMML decision tree.

the petal_length of 'the Decision_point'

In this editor, all the PMML elements of the new file are displayed in the **PMML Elements** pane. Then, in the **Verbalization** pane, enter the verbalization for each field.

This step is facilitated by the use of a **mapping file**, which you create when generating the original BOM class, and then specify as a starting point for PMML verbalization. When you specify a mapping file, you only need to provide a verbalization for PMML elements not present in the original PMML file.

If you do not specify a mapping file, the PMML Verbalization editor suggests a default verbalization for all the elements instead.

In either case, this verbalization of each PMML element is then used by the wizard to generate the decision tree, but the verbalization information is not stored in the BOM entry, which remains unchanged.

In a mapping file, each `mapping` field maps the PMML element with the element of the vocabulary of the BOM entry. Here is an example:

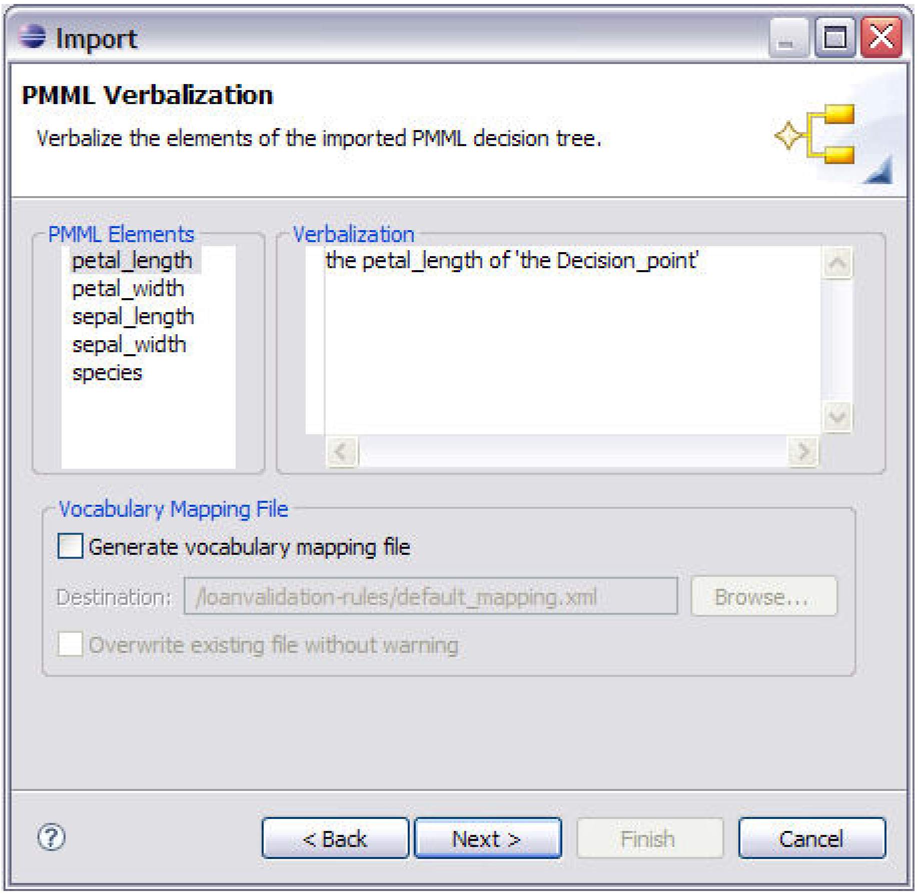
```
<?xml version="1.0" encoding="UTF-8"?>
<tns:mappings xmlns:tns="http://www.ibm.com/rules/analytics/mapping" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ibm.com/rules/analytics/mapping mapping.xsd ">
   <tns:locale>en-US</tns:locale>
   <tns:mapping>
      <tns:PMMLField>petal_width</tns:PMMLField>
      <tns:NavigationSentence>the width of 'the Flower'</tns:NavigationSentence>
   </tns:mapping>
   <tns:mapping>
      <tns:PMMLField>petal_length</tns:PMMLField>
      <tns:NavigationSentence>the length of 'the Flower'</tns:NavigationSentence>
   </tns:mapping>
   <tns:mapping>
      <tns:PMMLField>species</tns:PMMLField>
      <tns:NavigationSentence>the type of 'the Flower'</tns:NavigationSentence>
   </tns:mapping>
</tns:mappings>
```

To import a PMML file as a decision tree when the BOM class already exists:

1. In Rule Studio, switch to the Rule perspective, and select the rule project in which you want to import the PMML file.
2. Edit the BOM class to add, remove, or modify any members and their verbalization so that it matches the PMML file you want to import.
3. Click **File** > **Import**, select **Rule Studio** > **Predictive Analytics** > **PMML Decision Tree**, and click **Next**.
4. In the **Import PMML Decision Tree** page, enter or browse to a valid PMML file.

   Important: Make sure the PMML file conforms to the requirements and limitations described in *PMML support*.

5. Uncheck **Generate BOM classes**, and select the mapping file you created when generating the BOM class. This may take a moment as the wizard retrieves model elements and their verbalization. Then click **Next** to display the PMML Verbalization page.


Import
PMML Verbalization
Verbalize the elements of the imported PMML decision tree.
PMML Elements
petal_length
petal_width
sepal_length
sepal_width
species
Verbalization
the petal_length of 'the Decision_point'
Vocabulary Mapping File
Generate vocabulary mapping file
Destination: /loanvalidation-rules/default_mapping.xml
Browse...
Overwrite existing file without warning
< Back
Next >
Finish
Cancel

6. Verbalize all the elements found in the PMML file by selecting them one-by-one in the PMML Elements pane and editing the Verbalization pane for each entry. The Verbalization pane makes use of Content Completion to help you access any existing verbalizations found in the BOM, typically the ones you created before importing.

Note: If you specified a mapping file as a starting point to PMML verbalization, the Verbalization pane automatically proposes the verbalization for PMML elements found in both the new PMML file and the mapping file.

7. Save the verbalization information in a mapping file if required, by checking **Generate vocabulary mapping file** and specifying the destination. Then click **Next**.

Important: Make sure you correctly verbalize all the elements before clicking Next. The wizard does not advise you of any missing verbalizations, which can produce incorrect decision tree actions.

8. Enter the name and location of the new decision tree, and then click **Finish**.

You may get B2X errors and warnings until you create the corresponding XOM class, as described in *Integrating the decision tree into the rule project*. Also, you can remove irrelevant overlap and gap checker messages generated by the decision tree editor by turning off overlap and gap checking at either the node or tree level.

See also

Overview
PMML support
Integrating the decision tree into the rule project

Integrating the decision tree into the rule project

When you import a PMML file into a JRules decision tree with the option of generating the BOM classes, Rule Studio generates the BOM class, which contains one attribute per PMML field.

However, Rule Studio does not create a XOM class corresponding to the newly created BOM entry. You must do this manually. Ideally you should do this before importing the PMML file to avoid B2X errors, but you can also do it after.

To add the XOM class:

1. In the `/src` package of your XOM, create a class with the same name as the BOM class, for example `Decision_point.java`.
2. In this class, add a public attribute for each PMML field contained in your generated BOM class. For example:

```
public class Decision_point {
 public double petal_length;
 public double petal_width;
 public double sepal_length;
 public double sepal_width;
 public String species;
}
```

3. Save your work.

Note: Best practices around ruleflow orchestration suggest that each decision tree should be contained within its own rule task (see the JRules documentation for additional information).

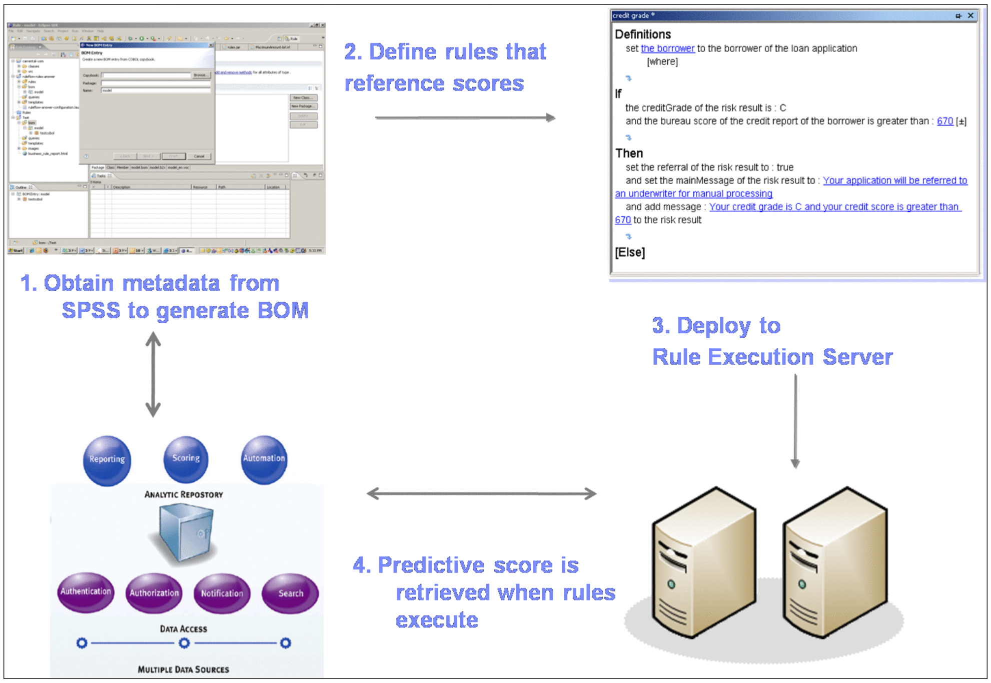
See also

Importing a PMML file to a decision tree
PMML support
Overview

Invoking the SPSS Scoring Service

In JRules, you can author business rules that reference predictive scores, which are then retrieved from a scoring service when the business rules execute. Using a scoring service is recommended when your analytic models change frequently, or when using models that cannot be represented in rule artifacts.

The following image shows how JRules integrates with the SPSS scoring service:


credit grade *
1. Obtain metadata from SPSS to generate BOM
2. Define rules that reference scores
3. Deploy to Rule Execution Server
4. Predictive score is retrieved when rules execute
Reporting
Scoring
Automation
Analytic Repository
Authentication
Authorization
Notification
Search
Data Access
Multiple Data Sources

Definitions
set the borrower to the borrower of the loan application
[where]

If
the creditGrade of the risk result is : C
and the bureau score of the credit report of the borrower is greater than : 670 [±]

Then
set the referral of the risk result to : true
and set the mainMessage of the risk result to : Your application will be referred to an underwriter for manual processing
and add message : Your credit grade is C and your credit score is greater than 670 to the risk result

[Else]

Retrieving models from the scoring service

To invoke a scoring service from a business rule, you first need to generate a XOM in JRules corresponding to the analytic models that you retrieve from the scoring service. A XOM class for an analytic model is a proxy class usable by the JRules engine to invoke the model, by passing the right input data.

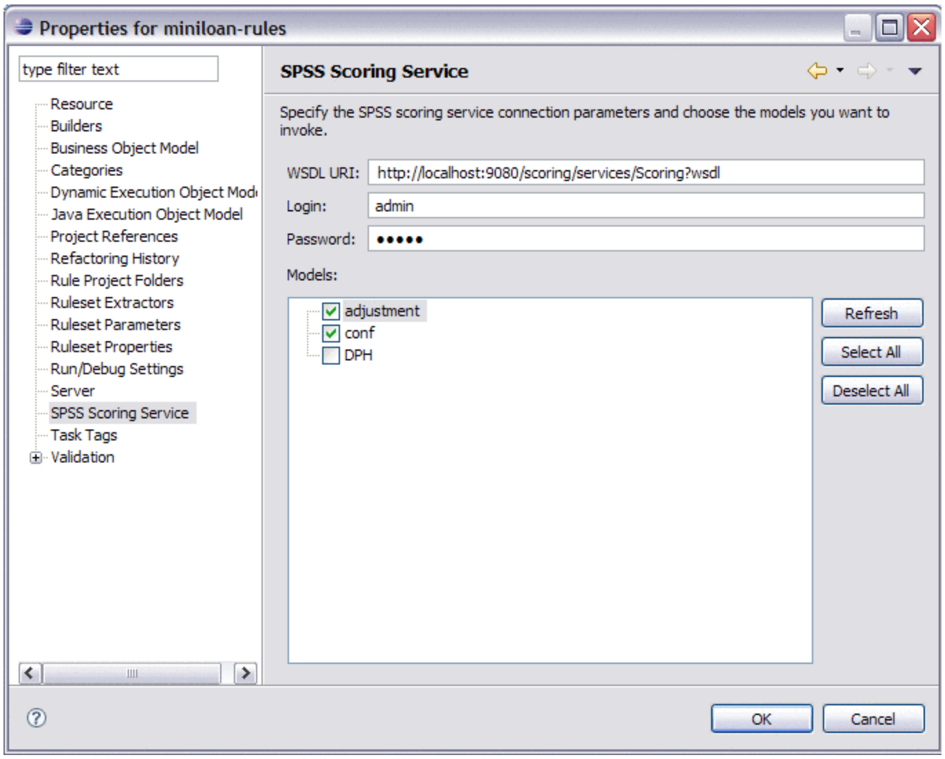
Retrieving models from the scoring service assumes that you have deployed and configured the models on the scoring service. The SPSS Scoring Service is available in SPSS Collaboration and Deployments Services (C&DS), where you deploy analytic models using the Deployment Manager. To retrieve the models in JRules, you need the URL of the scoring service (for example, `http://somehost:port/scoring/services/Scoring`), as well as the login/password created in C&DS. See its documentation for more information.

Retrieving the models from the scoring service must be done for each rule project where invocation is used, as follows:

1. In Rule Studio, right-click the rule project and click **Properties**.
2. Select **SPSS Scoring Service**.
3. Specify the connection parameters to the SPSS Scoring Service:
    - ♦ The URL of the scoring service or the URI of the WSDL.
    - ♦ The login/password.

    Warning: The login and password are encrypted and stored in the XOM header and in the Eclipse workspace metadata. The key used to encrypt the data cannot be changed.

4. Click **Refresh**. The available models are displayed.
5. In the Models section, select the models you wish to import:



SPSS Scoring Service

Specify the SPSS scoring service connection parameters and choose the models you want to invoke.

WSDL URI: http://localhost:9080/scoring/services/Scoring?wsdl

Login: admin

Password: •••••

Models:

- [✓] adjustment
- [✓] conf
- [ ] DPH

6. Click **OK**.

   All the models are now available in package `com.ibm.rules.analytics/` in the XOM file `spss.scoringservice.models.xom`.

Note: The XOM is not published to Rule Team Server. If in another workspace you import a project from Rule Team Server that contains rules that invoke a scoring service, you must reset the scoring service parameters and regenerate the XOM.

In addition, the following JAR files are referenced in the rule project as XOM entries:

- `<JRulesInstallDir>/SupportPacs/SupportPacLB02/lib/velocity-1.5.jar`
- `<JRulesInstallDir>/SupportPacs/SupportPacLB02/lib/commons-collections-3.1.jar`

- `<JRulesInstallDir>/SupportPacs/SupportPacLB02/lib/commons-lang-2.1.jar`
- `<JRulesInstallDir>/SupportPacs/SupportPacLB02/lib/jrules-analytics-runtime.jar`

Understanding the generated XOM

The generated XOM class is a representation of the SPSS models in a model executable by JRules.

These XOM classes contain:

- A constructor with no arguments.
- Some fields: Each field has a name, a type, and some properties: (Note: Do not change these).
  - The `InitialName` property: Contains the name of the field in the analytic model. The name of the field changes if the initial name contains spaces or special characters.
  - The `Direction` properties can be one or more of the following:
    - **Required**: A required input field.
    - **Optional**: An optional input field.
    - **Returned**: An output field.
- A `getScore()` method: All the required fields must have a value when calling `getScore()`, otherwise an exception is thrown. The `getScore()` method invokes the scoring service, and then sets a value on all the returned fields. You can choose which fields to make use of.
- A `reset()` method: Returns the object to its initial state.

The following example shows a typical XOM class:

```
package com.ibm.rules.analytics.spss.scoringservice.models;

public class PatientKMeans
property InitialName "PatientKMeans"
property "ilog.rules.engine.driver" "com.ibm.rules.analytics.bom.ScoringDriver"

{
    public PatientKMeans();

    public double Age
                property Direction "required"
                property InitialName "Age";
    public string BP
                property Direction "required"
                property InitialName "BP";
    public string Cholesterol
                property Direction "required"
                property InitialName "Cholesterol";
    public string Drug
                property Direction "required"
                property InitialName "Drug";
    public double K
                property Direction "required"
                property InitialName "K";
    public double Na
                property Direction "required"
                property InitialName "Na";
    public string Sex
                property Direction "required"
                property InitialName "Sex";
    public string Prediction
                property Direction "returned"
                property InitialName "Prediction";

    public void getScore();

    public void reset();
}
```

Creating methods to invoke the scoring service

After you have generated the XOM corresponding to the analytic models retrieved from the scoring service, you must create BOM methods that invoke the scoring service, so that they are usable in your business rules after verbalization.

To start this, you can either:

- Create a new BOM entry from your generated XOM, and then create a virtual BOM method for the invocation.
- Create a virtual BOM method in an existing BOM class.

Then, add code to the **BOM to XOM Mapping** section of the virtual BOM method so that it sets the required fields and calls the `getScore()` method.

The following example shows a `getPrediction()` method, added to an existing BOM class `Patient`, that defines the same attributes as the `PatientKMeans` BOM class (Age, Sex, BP, and so on). This `getPrediction()` method is defined as follows:

1. In the **Edit the imports** section:

   ```
   // Imports the XOM classes from the scoring service package.
   import com.ibm.rules.analytics.scoringservice.models.PatientKMeans();
   ```

2. In the **Body** section:

```
// Creates a new object to invoke the analytic model
PatientKMeans obj = new PatientKMeans();

// Set the fields. All the required fields must be set.
obj.Age = this.Age;
obj.Sex = this.Sex;
obj.BP = this.BP;
obj.Cholesterol = this.Cholesterol;
obj.Na = this.Na;
obj.K = this.K;
obj.Drug = this.Drug;

// Invoke the analytic model, now that the inputs are filled.
obj.getScore();

// Return one of the computed result (the score).
String res = obj.Prediction;
return res;
```

Note: Typically you write this code in the B2X body of the BOM method, but it can be written anywhere you write XOM-based IRL.

Invoking a scoring service

You can invoke the scoring service from Rule Studio or from an Enterprise Application.

Note: You can also use Decision Validation Services to test the invocation of the scoring service. See the **Testing and simulating rulesets** section of the JRules documentation.

Invoking a scoring service from Rule Studio

You can test that the invocation of the scoring service is working by executing the rule project in Rule Studio.

To execute in Rule Studio using the SPSS Scoring Service:

1. In Rule Studio, select the rule project and click **Run** > **Run Configurations**.
2. In the **Run Configurations** panel, right-click the **Rule Project** category and choose **New** to create a new launch configuration.
3. Click **Browse** in the Rule Project tab and select the rule project.
4. Click **Apply**, and then **Run**.

Invoking a scoring service from an enterprise application

To use the scoring service invocation from enterprise applications for WebSphere Application Server 7, you must package the four runtime JAR files located in `<JRulesInstallDir>/SupportPacs/SupportPacLB02/lib/` to `myWar.war/WEB-INF/lib`.

If you have several WAR files within a same EAR, and some of the JAR files are shared among two or more WAR files, you can move some of the JAR files one level up, for example:

- `META-INF`
- `shared.jar`
- `w1.war`
- `w2.war`
- `w3.war`

Here, `w1.war`, `w2.war`, and `w3.war` share the same JAR file called `shared.jar`. Therefore, you can move `shared.jar` up one level, directly under the EAR.

Also, reverse the class loaders when you package the EAR file and deploy WebSphere Application Server 7. Reverse the class loaders for all the WAR modules, and for the EAR itself. For more details, consult the WebSphere Application Server 7 documentation.

See also

Product overview
Type support for scoring service invocation

Reference

Reference information on error messages and supported data types.

In this section

PMML import messages
Description of warning or error messages displayed during PMML import.

Type support for PMML import
Supported data types for PMML import.

Type support for scoring service invocation
Supported data types for scoring service invocation.

PMML import messages

The following table lists warning or error messages displayed during PMML import:

ID	Text	Description
P2J0008	File does not exist.	Wizard points to a PMML or mapping file that does not exist.
P2J0020	Could not read PMML file.	`IOException` while reading the contents of a PMML file.
P2J0021	Could not detect PMML version.	PMML version not detected.
P2J0022	Unsupported PMML version.	PMML version not supported.
P2J0023	Could not parse PMML.	Error while parsing the PMML file.
P2J0024	Unsupported PMML feature.	PMML file contains a `XOR`.
P2J0027	No TreeModel found in PMML file.	Only TreeModel is supported.
P2J0028	The imported PMML file contains surrogates. The first predicate of each surrogate was included in the tree.	See limitations on surrogates in *PMML support*.
P2J0029	The imported PMML file contains date or time types that are treated as integers.	See limitations on dates and times in *PMML support*.
P2J0040	Could not read mapping file	`IOException` while reading the contents of a mapping file.
P2J0041	Could not parse mapping	`XMLException` while parsing the mapping file.
P2J0042	Can only convert temporal types to date or time, not <X>.	Code asked the `DataTypeMapper` class to convert a temporal type

ID	Text	Description
		besides date, time, or dateTime to a BAL string.
P2J0043	All children of the same node in the PMML tree must test the same field (i.e. only univariate split are supported).	See limitations in *PMML support*.
P2J0050	A field named <X> was used in a PMML predicate, but does not exist in the PMML DataDictionary	Field must exist in DataDictionary.

Type support for PMML import

The following table lists the supported data types for PMML import into JRules decision trees and the mapped Java type:

Type Name	Description	Mapped Java Type
string	A string	java.lang.String
integer	An int	int
float	A float	float
double	A double	double
date	XML compliant date type	java.util.Date
time	XML compliant date type (time only)	java.util.Date
dateTime	XML compliant date type (a date + a time)	java.util.Date
dateDaysSince[0]	Date, represented as number of days since year 0 (for ex: 1980 is 1980)	int
dateDaysSince[1960]	Date, represented as number of days since year 1960	int
dateDaysSince[1970]	Date, represented as number of days since year 1970	int
dateDaysSince[1980]	Date, represented as number of days since year 1980	int
timeSeconds	PMML specific date type	int
dateTimeSecondsSince[0]	Date expressed in elapsed seconds since year 0	int
dateTimeSecondsSince[1960]	Date expressed in elapsed seconds since year 1960	int
dateTimeSecondsSince[1970]	Date expressed in elapsed seconds since year 1970	int
dateTimeSecondsSince[1980]	Date expressed in elapsed seconds since year 1980	int

Type support for scoring service invocation

The following table lists the supported data types for scoring service invocation from JRules and the mapped Java type:

Type Name	Description	Mapped Java Type
boolean	A boolean	boolean
integer	An int	int
long	A long	long
float	A float	float
double	A double	double
decimal	A decimal	double
date	A date (W3C compliant)	XMLGregorianCalendar
daytime	A time (W3C compliant)	XMLGregorianCalendar
timestamp	A date and a time (W3C compliant)	XMLGregorianCalendar

Glossary

Decision Tree model

The model in the Predictive Modeling tool (SPSS Modeler, SAS Enterprise Miner) where these models are usually created. They are represented and stored in a proprietary format.

JRules decision tree

The JRules artifact created from the imported PMML file.

PMML TreeModel

A Decision Tree model once it has been exported from the modeling tool and is in a PMML file.