WebSphere® Process Server v6.1

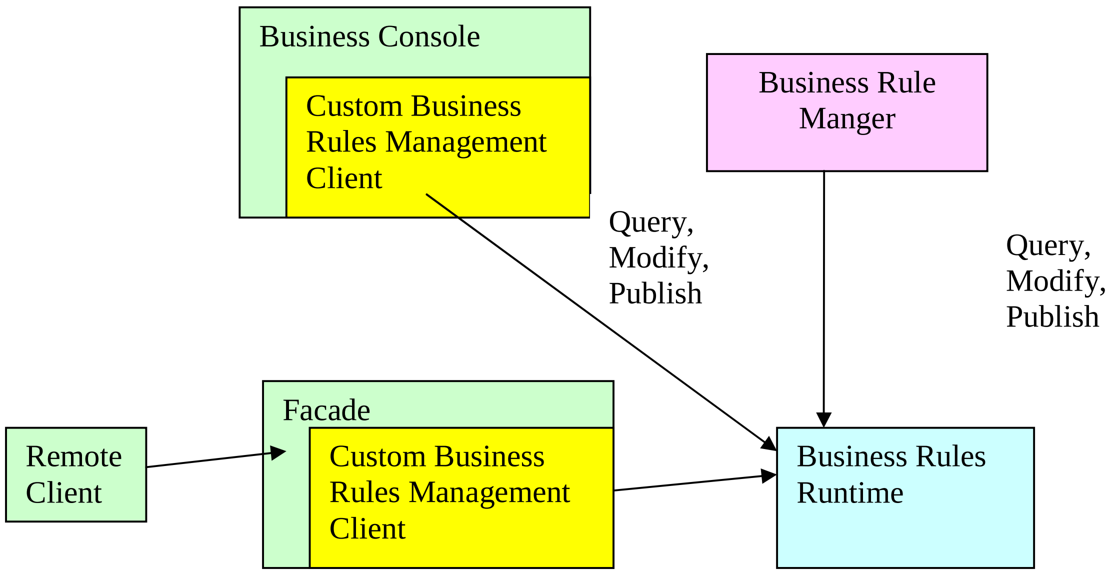
Business Rule Management Programming Guide

Table of Contents

1.0 Introduction

Public business rule management classes are provided for the building of custom management clients or to automate changes to business rules. The classes could be used in a Web application where they are combined with other management capabilities for things such as business process or human tasks in order to manage all components from a single client. Any custom management clients can be used along side the Business Rule Manager Web application included with WebSphere Process Server. The classes could also be used to automate changes to business rules within an application. For example business rules could be changed as the results of a business process that is using the business rules exceeds some threshold or limit.

The business rule management classes must be used in an application installed on WebSphere Process Server. The classes do not provide a remote interface, however they can be wrapped in a facade which is then exposed over a specific protocol for remote execution.



This programming guide is composed of two main sections and an appendix. The first section explains the programming model and how to use the different classes. Class diagrams are provided to show the relationship between classes. The second section provides examples on using the classes to perform actions such as searching for business rule groups, scheduling a new rule destination, and modifying a rule set or decision table. The appendix contains additional classes that were used in the examples to simplify common operations and additional examples of creating complex queries for searching for business rule groups using wildcards.

Besides this programming guide information on the classes is also available in Javadoc HTML format included with both the WebSphere Process Server v6.1 and test environment included with WebSphere Integration Developer v6.1. This Javadoc documentation is available at ${WebSphere Process Server Install Directory}\web\apidocs or ${WebSphere Integration Developer Install Directory}\runtimes\bi_v61\web\apidocs. The packages com.ibm.wbiserver.brules.mgmt.* contain all of the information.

2.0 Programming Model

WebSphere Business Integration Business Rules are authored with two different authoring tools and issued by the rule runtime. All three share the same model for the business rule artifacts. Sharing of the model was deemed critical for not only ease of future maintenance, but for a consistent programming model for the end user. Sharing this model required compromises between the needs of desktop tooling and runtime execution and authoring – all have clear sets of requirements to meet for their respective environments and these requirements at times conflicted with each other. The artifacts described below as part of the overall programming model represent a balance in meeting the requirements of these different environments.

Modification of business rules is limited to only those items that are defined with templates in the rule sets and decision tables as well as the operation selection table (effective dates and targets). Creation of new rule sets and decision tables is only supported through the copy of an existing rule set or decision table. The business rule group component itself is not eligible for dynamic authoring in the runtime with the exception of the user defined properties and description values. Changes that need to be made to the component (for example, adding a new operation) must be done using WebSphere Integration Developer and then redeployed or reinstalled in the server.

2.1 Business Rule Group

The `BusinessRuleGroup` class represents the business rule group component. The `BusinessRuleGroup` class can be considered the root object which contains rule sets and decision tables. Rule sets and decision tables can only be reached through the business rule group that they are associated with. Methods are provided on the class to retrieve information about the business rule group and to reach the rule sets and decision tables. Through the methods the following information can be retrieved:

- Target name space
- Name of business rule group
- Display name
- Name/Display name synchronization
- Description
- Presentation time zone which indicates whether dates should be displayed in UTC format or local to the system
- Operations defined in the interface associated with the business rule group
- Custom properties defined on the business rule group

The different rule sets and decision tables associated with the business rule group can be reached through the business rule group's operation.

There are also methods that allow for information to be updated on the business rule group. Through the methods, the following information can be updated:

- Description
- Display name
- Name/Display name synchronization
- Custom properties defined on the business rule group

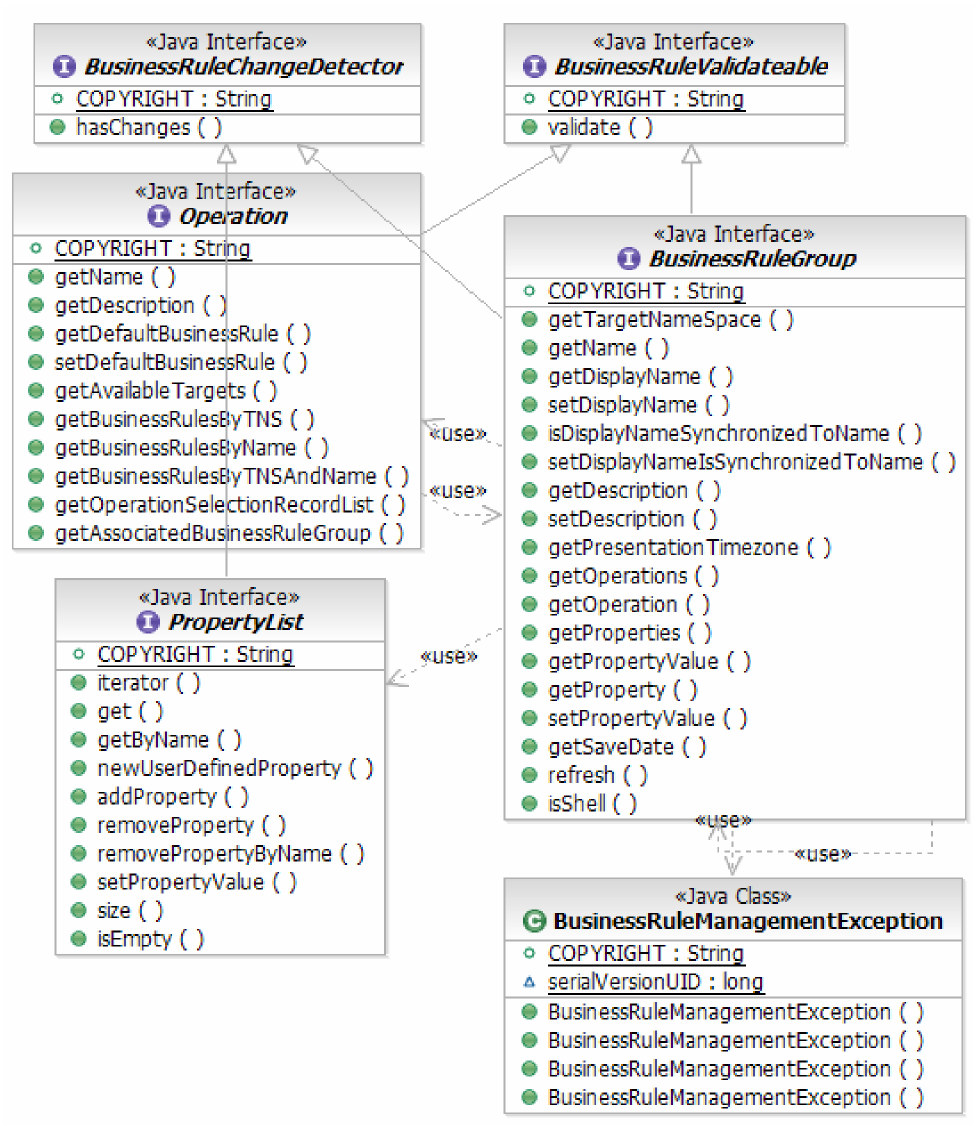
The Display name for the business rule group can be set explicitly or it can be set to the value of the Name using the `setDisplayNameIsSynchronizedToName` method.

Other values cannot be modified as these are part of the business rule group component definition and changes to these values would require a redeploy as well as reinstallation.

The business rule group class also provides a `refresh` method. This method will make a call to the persistent storage or repository where the business rules are stored and return the business rule group and all of the associated rule sets and decision tables with the persisted information. The returned business rule group is the latest copy and the previous object is obsolete.

The `isShell` method can be used to tell if a business rule group instance is of a version that is not supported by the current runtime. For example, if a web client was created with the current business rule management classes, and in the future new capabilities are added to the business rule group that are not supported by the classes, a shell business rule group will be created when the business rule group is retrieved. This allows the web client to continue to work with business rules that are supported and still retrieve business rule groups with limited attributes and capabilities. When `isShell` is true, only the methods `getName`, `getTargetNameSpace`, `getProperties`, `getPropertyValue`, and `getProperty` will return values. All other methods will result in an `UnsupportedOperationException`. Besides using the `isShell` method, the type of the `BusinessRuleGroup` can also be checked if it is an instance of `BusinessRuleGroupShell` in order to determine if it is of a supported version.

Figure 1. Class diagram of BusinessRuleGroup and related classes



2.2 Business Rule Group Properties

The properties on business rule groups are intended to be used for management of business rule groups. Properties set on business rule groups can be used in queries to return only a subset of business rule groups which are to be displayed and then modified. All properties are of type string and defined as name-value pairs. Each property can only be defined once in a business rule group. For each property defined, it must have a value also defined. The property value can be an empty string or zero in length, but not null. Setting a property to null is the same as deleting the property.

The properties on a business rule group can also be accessed in a rule set or decision table at runtime. This allows a single value to be set at the business rule group to be used within multiple rule sets or decision tables in the business rule group. Only those properties defined on the business rule group are available to enclosed rule sets and decision tables.

There are two types of properties, system and user-defined. The number of system or user-defined properties is not limited on a business rule group. The system properties are used to hold specific component information such as the version of the rule model used in defining the rule logic. This system information is exposed in properties to allow for query across these fields. The system properties begin with a prefix IBMSystem and are read-only through the business rule group and property classes. System properties can not be added, changed or deleted. Example of system property is:

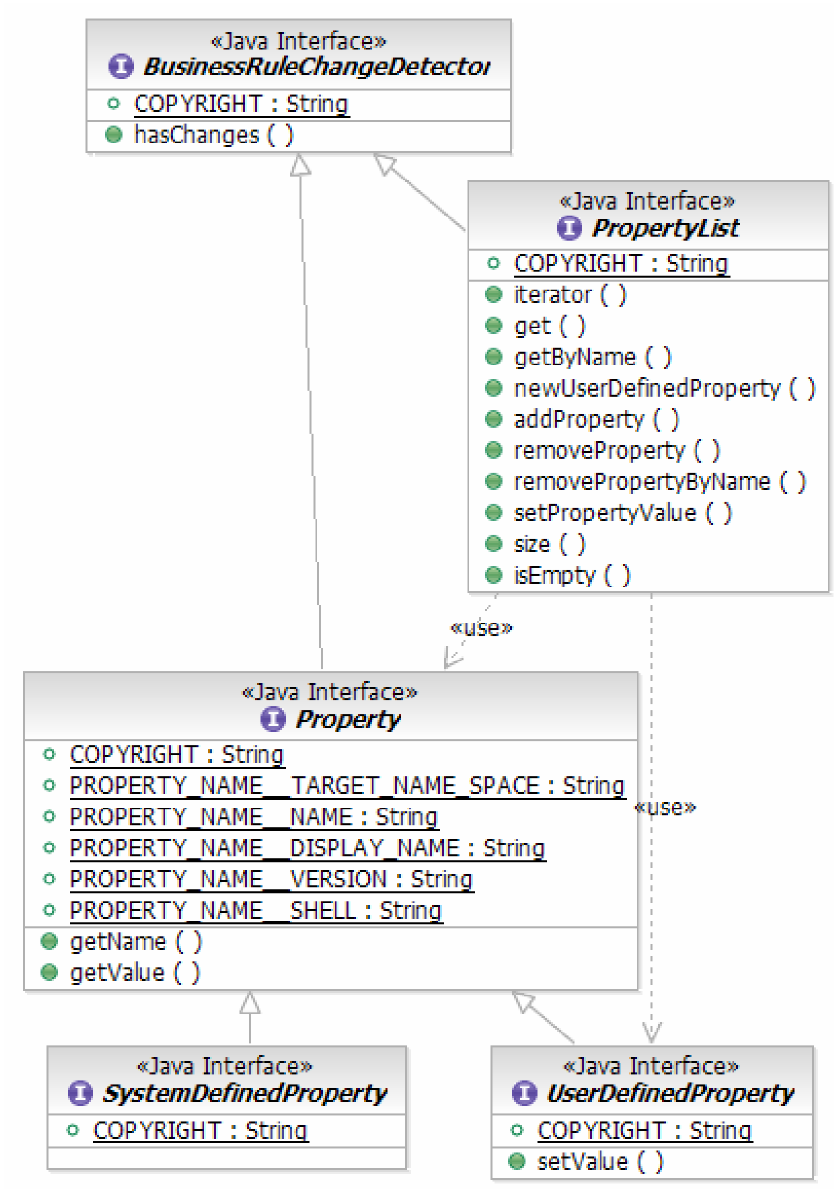
Property Name	**Property Value**
IBMSystemVersion	6.1.0

Note: The values of name, namespace and display name for a business rule group are treated as system properties for query purposes, and will be part of the list of properties that can be retrieved for a business rule group with the `getProperties` method. These properties are not however, defined as actual property elements in the business rule group artifact and are not seen as properties in WebSphere Integration Developer as they are defined with separate and unique elements on the business rule group. They are solely provided to offer more query options.

User-defined properties are available to be used for holding any customer specific information and can also be used in queries for business rule groups. User-defined properties are available for read-write.

The properties for a business rule group can be retrieved either individually or as a list (PropertyList object). With the PropertyList, methods are provided for retrieving individual properties and adding and removing user-defined properties.

Figure 2. Class diagram of Property and related classes



2.3 Operation

Operations are starting points for reaching individual rule sets and decision tables to modify. The operations of a business rule group match the operations listed in the WSDL which is associated with the business rule group component. For each operation, there are different targets, each of which is a business rule (rule set or decision table):

- Default target (optional)
- List of targets scheduled by date/time ranges (OperationSelectionRecord)
- List of all available targets that can be used for that operation

Each operation must have at least one business rule target specified. This target can be an `OperationSelectionRecord` with a specific start date and end date when the target should be scheduled to be active. The operation can also have a single default target set which is used during execution when no matching scheduled business rule target is found. The `Operation` class provides methods for retrieving and setting the default business rule target as well as retrieving the list (`OperationSelectionRecordList`) of scheduled business rule targets. Besides the default business rule target and the scheduled business rule targets, there is a list of all available business rule targets for the operation. This list will include those business rules targets which are scheduled and the default business rule target as well as any other rule sets or decision tables which are not scheduled for this operation. An unscheduled rule set or decision table is associated with the operation through the available target list by the fact that it implicitly shares the operation information. All business rule targets must support the input and output messages for their operation. With each operation unique on an interface, the rule sets and decision tables for an operation are unique from those rule sets and decision tables of another operation.

Any of the different rule sets and decision tables in the available targets list can be scheduled to be active through the creation of an `OperationSelectionRecord`. Along with the particular rule set or decision table from the available targets list, a start date and end date must be specified. The start date must be before the end date. The dates can be for a time which covers the current date as well as the past and the future. The time span of the dates cannot overlap with any other `OperationSelectionRecords` once it is added to the `OperationSelectionRecordList` and published. The start date and end date values are of type java.util.Date. Any values which are specified will be treated as UTC values according to the java.util.Date class. With the `OperationSelectionRecord` complete, it can be added to the `OperationSelectionRecordList` to be scheduled along with other business rule targets. Gaps may exist between the time spans of different `OperationSelectionRecords`. When a gap is encountered during execution, the default target is used. If no default target has been specified, an exception will be thrown. As a best practice, it is recommended to always specify a default business rule target.

A scheduled business rule target can be removed from the list of scheduled targets by removing the `OperationSelectionRecord` from the `OperationSelectionRecordList`. Removing an `OperationSelectionRecord` will not remove the business rule target from the list of available business rule targets and it will not remove any other `OperationSelectionRecords` which have the same business rule target scheduled.

Besides retrieving a rule set or decision table through the `OperationSelectionRecordList` or list of available targets, the `Operation` class also allows for business rule targets to be retrieved by name and target namespace property values. Through the methods on the `Operation` class, those rule sets and decision tables which are listed in the available targets for that operation can be queried. Rule sets and decision tables which might have matching name and target namespace values, but are part of the available target lists of other operations, will not be included in the result set. As a convenience, the `getBusinessRulesByName`, `getBusinessRulesByTNS`, and `getBusinessRulesByTNSAndName` methods are provided to simplify retrieving specific rule sets and decision tables.

The `Operation` class provides methods that support the following:

- Retrieve the operation name
- Retrieve the operation description
- Retrieve and set the default business rule target
- Retrieve the scheduled business rule targets (`OperationSelectionRecordList`)
- Retrieve the list of all available business rule targets
- Retrieve a rule set or decision table from the list of all available targets by name or target namespace
- Retrieve the business rule group with which the operation is associated

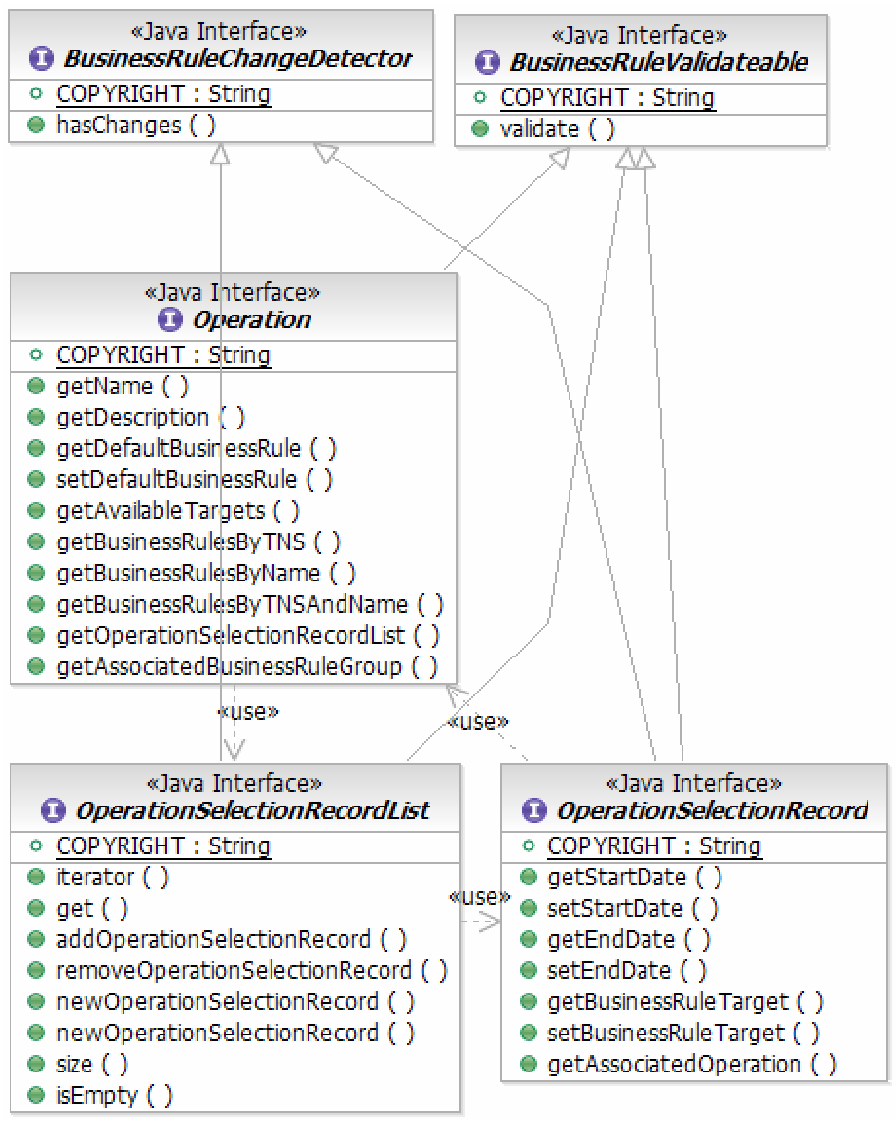
The `OperationSelectionRecordList` class provides methods that support the following:

- Retrieve a specific `OperationSelectionRecord` by index value
- Remove a specific `OperationSelectionRecord` by index value
- Add a new `OperationSelectionRecord` to the list

The `OperationSelectionRecord` class provides methods that support the following:

- Retrieve and set the start date
- Retrieve and set the end date
- Retrieve and set the business rule target
- Retrieve the operation with which the `OperationSelectionRecord` is associated

Figure 3. Class diagram for `Operation` and related classes


«Java Interface»
BusinessRuleChangeDetector
COPYRIGHT : String
hasChanges ( )
«Java Interface»
BusinessRuleValidateable
COPYRIGHT : String
validate ( )
«Java Interface»
Operation
COPYRIGHT : String
getName ( )
getDescription ( )
getDefaultBusinessRule ( )
setDefaultBusinessRule ( )
getAvailableTargets ( )
getBusinessRulesByTNS ( )
getBusinessRulesByName ( )
getBusinessRulesByTNSAndName ( )
getOperationSelectionRecordList ( )
getAssociatedBusinessRuleGroup ( )
«use»
«use»
«Java Interface»
OperationSelectionRecordList
COPYRIGHT : String
iterator ( )
get ( )
addOperationSelectionRecord ( )
removeOperationSelectionRecord ( )
newOperationSelectionRecord ( )
newOperationSelectionRecord ( )
size ( )
isEmpty ( )
«use»
«Java Interface»
OperationSelectionRecord
COPYRIGHT : String
getStartDate ( )
setStartDate ( )
getEndDate ( )
setEndDate ( )
getBusinessRuleTarget ( )
setBusinessRuleTarget ( )
getAssociatedOperation ( )

2.4 Business Rule

The `RuleSet` and `DecisionTable` classes are based off a generic `BusinessRule` class with methods that provide information that is available on both rule sets and decision tables.

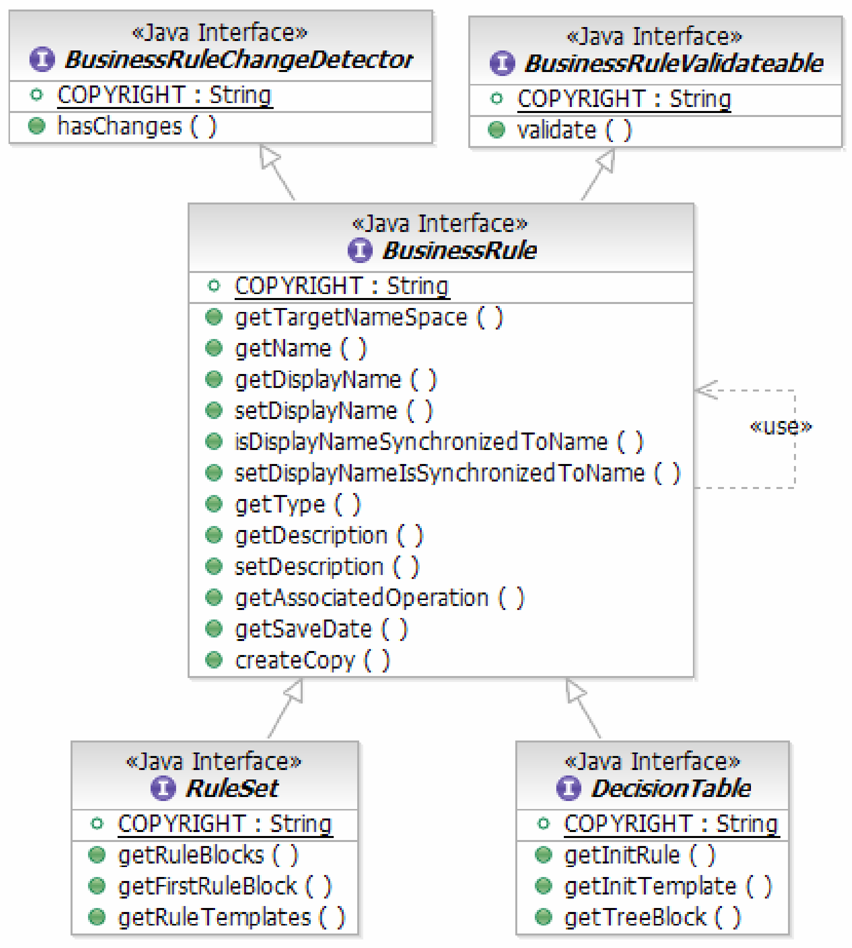
Similar to business rule group artifacts, rule sets and decision tables have a name and a target namespace. The combination of these values must be unique when compared to other rule sets and decision tables. For example, two rule sets can share the same target namespace value, but must have different names or a rule set and decision table could have the same name, but have different target namespace values.

A copy of a business rule can be made from an existing business rule for situations where a similar rule is required to be scheduled at a specific time with different parameter values for rules constructed from templates. New rules cannot be fully created from scratch as there must be a backing Java™ class to provide the implementation for the business rule. The backing Java class is only created at deploy time. When a new rule is created, it is added to the list of available targets for the operation which is associated with the original rule. The additional rule is not persisted however, until the business rule group with which the operation is associated is published. The new business rule must have either a different target namespace or name from the original rule. The display name for the new business rule can remain the same as the original rule as the combination of the name and namespace provide a key value to identify the business rule. Within the business rule, the different parameter values which have been defined with a template can be modified. Scheduling the business rule at a certain time can be done with the `OperationSelectionRecordList` or as a default destination with the `Operation` associated with the business rule.

The `BusinessRule` class provides methods that support the following:

- Retrieve the target namespace
- Retrieve the name of the rule set or decision table
- Retrieve and set the display name of the rule set or decision table
- Retrieve the type of the business rule, either rule set or decision table
- Retrieve and set the description for the business rule
- Retrieve the operation that the business rule is associated with.
- Create a copy of the business rule with a different name and/or target namespace

Figure 4. Class diagram of `BusinessRule` and related classes



2.5 Rule set

A rule set is one type of business rule. Rule sets are typically used when multiple rules may need to be executed based on different conditional values. Rule sets are composed of a rule block and rule templates. The rule block (`RuleBlock`) contains the different if-then and action rules which make up the logic of the rule set.

The `RuleSet` class provides methods that support the following:

- Retrieve a list of rule blocks for the rule set
- Retrieve a list of rule templates defined in the rules set

Currently each rule set can only have one rule block, while there can be multiple rule templates defined in the rule set. The rule block contains the set of rules that will be executed when the rule set is invoked. The rule block allows for the order of the rules to be modified. A rule block must have at least one rule defined. The rules (`Rule`) can be defined as template instance rules (`TemplateInstanceRule`) or hard-coded. If an if-then or action rule has been defined with a template, it can be removed from the rule block. If a new instance of rule was created with a template, it can be added to the rule block.

If a rule is hard-coded and was not defined with a template, it cannot be modified or removed from the rule block. The expectation with these rules is that they have been designed to always be part of the rule set logic and are not to be changed or repeated within the logic.

When a new rule is created with a template, it must have a unique name value. The list of existing rules can be retrieved and checked first before creating the rule.

For hard-coded if-then and action rules, only the name and presentation can be retrieved. The presentation is a string which can be used to display information about the rule in client applications. For if-then or action rules that are defined with a template, the name and presentation can be retrieved as well as additional information. Specific parameter values can be retrieved and changed. With a template (`RuleSetRuleTemplate`) defined in the rule set, another instance of the rule can be created within the rule set and parameter values can be set. For example, if you have a rule saying that a customer of a particular status level receives a discount of a specific amount. This logic could be defined with a single rule template and then repeated with parameter values changed for the status level (gold, silver, bronze, and so on) and the discount amount (15%, 10%, 5%, and so on).

The parameters for a rule defined with a template are specific to the instance of the rule. The template only defines a standard presentation and the number of parameters for the rule. Each rule defined with a template can have different values as explained in the example on discounts for different customer status.

The `RuleBlock` class provides methods that support the following:

- Retrieve a rule by index
- Add a rule that was defined with a template
- Remove a rule defined with a template
- Modify the order of a rule by one place or to a specific index location

The `RuleSetRule` class provides methods that support the following:

- Retrieve the name of the rule
- Retrieve the display name of the rule
- Retrieve the user presentation
- Retrieve the rule block

The `RuleSetRuleTemplate` class provides methods that support the following:

- Create a rule template instance from this template definition
- Retrieve the parent rule set

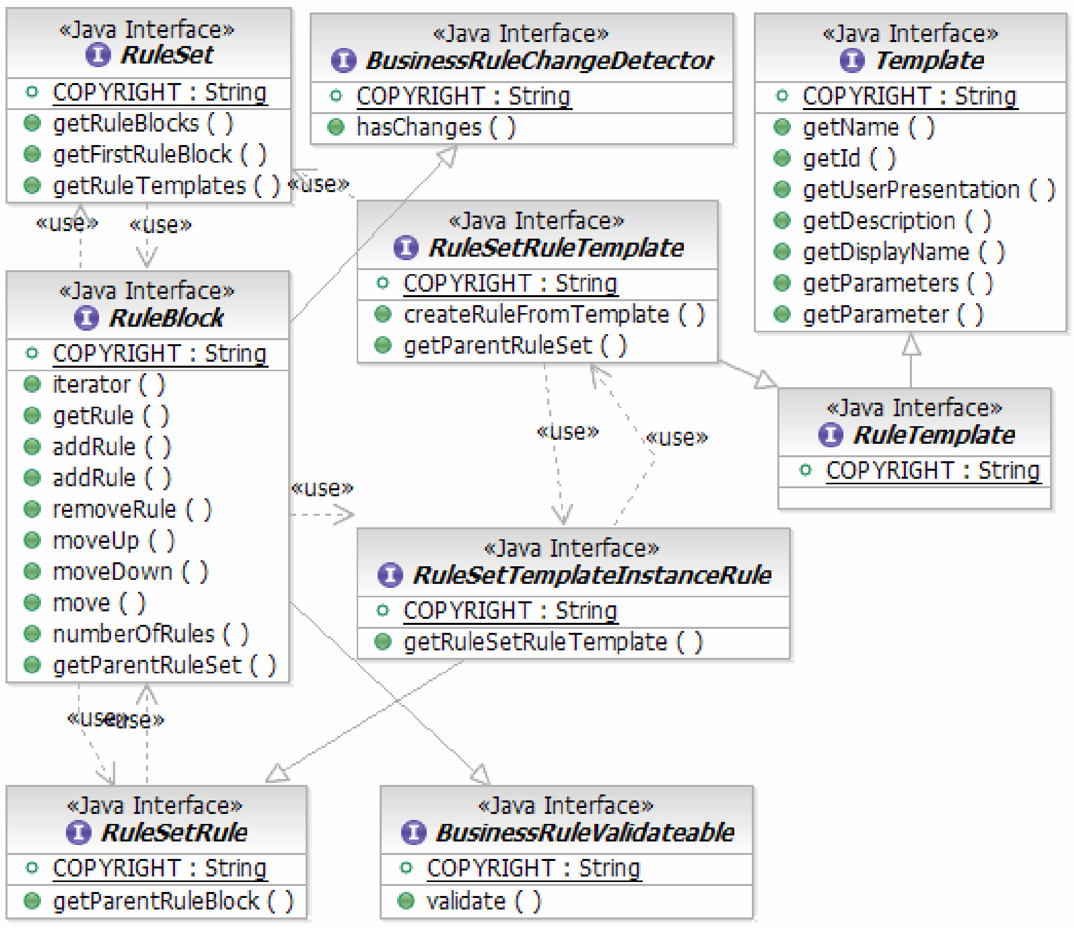
The `TemplateInstanceRule` class provides methods that support the following:

- Retrieve the parameters for the rule
- Retrieve the template definition which defined the rule

The `Template` class provides methods that support the following:

- Retrieve the template ID
- Retrieve the name
- Retrieve and set the display name
- Retrieve and set the description
- Retrieve the parameters for this template ~~values~~
- Retrieve the user presentation

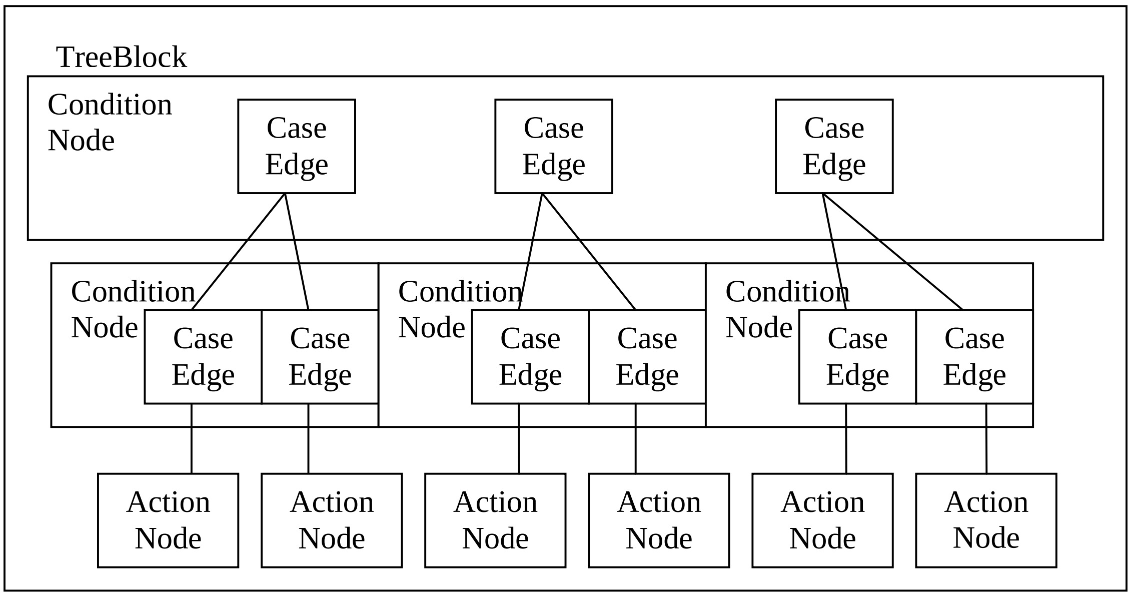
Figure 5. Class diagram of `RuleSet` and related classes



2.6 Decision Table

Decision tables are another type of business rule which can be managed and modified. Decision tables are typically used when there are a consistent number of conditions which must be evaluated and a specific set of actions to be issued when the conditions are met. Decision tables are similar to decision trees, however they are balanced. Decision tables always have the same number of conditions to be evaluated and actions to be performed no matter what set of branches are resolved to true. A decision tree may have one branch with more conditions to evaluate than another branch.

Decision tables are structured as a tree of nodes and defined by a `TreeBlock`. There are different `TreeNodes` which make up the `TreeBlock`. `TreeNodes` can be condition nodes or action nodes. Condition nodes are the evaluation branches. At the end of branches, there are action nodes that have the appropriate tree actions to issue should all of the conditions evaluate to true. There can be any number of levels of condition nodes, but only one level of action nodes.

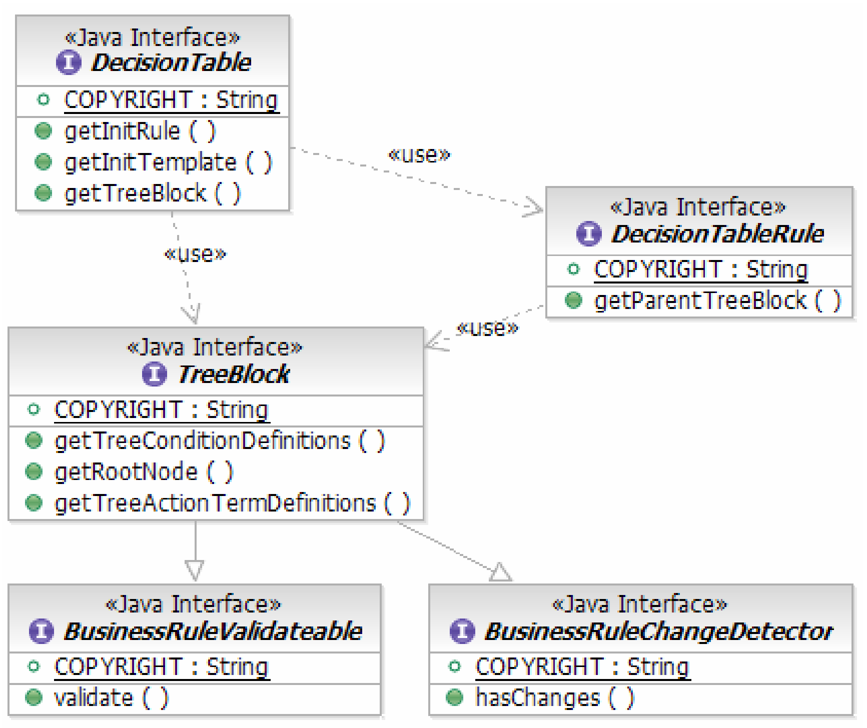


Decision tables might also have an initialization rule (init rule) which can be issued before the conditions in the table are checked.

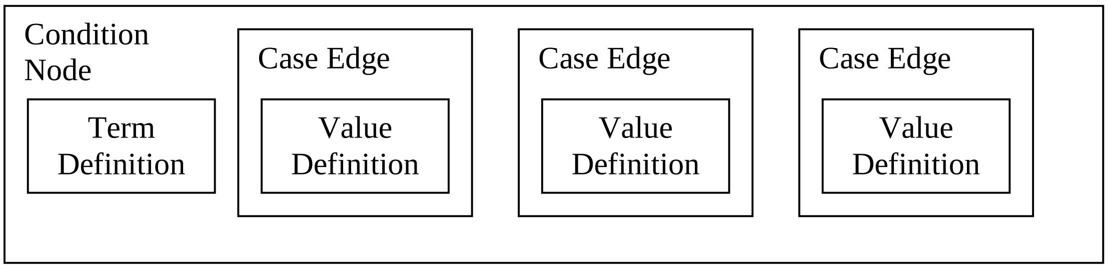
The `DecisionTable` class provides methods that support the following:

- Retrieve the tree block of tree nodes (condition and action nodes)
- Retrieve the init rule instance
- Retrieve the init rule template if defined

Figure 6. Class diagram of `DecisionTable` and related classes



The `TreeBlock` of a decision table contains the different condition and action nodes. Each condition node (`ConditionNode`) has a term definition (`TreeConditionTermDefinition`) and one to *n* case edges (`CaseEdge`). The term definition contains the left-hand operand for the condition expression. The case edges contain the value definitions which are the different right-hand operands to be used in the condition expression. For example, in the expression (status == "gold") the term definition would be "status" and "gold" would be the value definition in the case edge. For all of the case edges in a condition node, they share the term definition and are only different by the value (`TreeConditionValueDefinition`). Continuing with the example, another case edge in the condition node could have a value "silver". This would be used in an expression too (status == "silver"). The only exception to this behavior is if an otherwise has been defined for the condition node. With an otherwise, there is no value definition as it is used if all other case edges within the condition node evaluate to false. While an otherwise is not a case edge, it does have a `TreeNode` that can be retrieved.



For the term definition, the user presentation can be retrieved and used in client applications. The presentation for the term definition is typically only a representation of the left-hand operand (status in our example) and does not contain any placeholders. For the case edges, a template can be used to define the value definition (`TreeConditionValueTemplate`). A template value definition instance (`TemplateInstanceExpression`) actually holds the parameter values which are used for execution and can be modified. If an attempt is made to retrieve the value template definition for a `TreeConditionValueDefinition` that was not defined with a template, a null value will be returned. If a template has not been used to define the value condition, a user presentation can still be retrieved and used in client applications if it was specified at authoring time.

The `TreeBlock` class provides methods that support the following:

- Retrieve the root node of the tree
- Retrieve the condition term definitions for the tree block
- Retrieve the action term definitions for the tree block

The root node of the tree is of type `TreeNode` and from here, navigation of the decision table can occur. The `TreeNode` class provides methods that support the following:

- Determine if a node is an otherwise clause
- Retrieve the parent node for the current tree node (condition or action node)
- Retrieve the root node of the tree containing the current tree node

The `ConditionNode` class provides methods that support the following:

- Retrieve the case edges
- Retrieve the term definition
- Retrieve the otherwise case
- Retrieve the templates for the value conditions of the case edges for the condition node
- Add a condition value based on a template to the node
- Remove a condition value based on a template

The `CaseEdge` class provides methods that support the following:

- Retrieve the list of value templates which are available for the value definition
- Retrieve the child node (condition or action node)
- Retrieve the instance of the template definition associated with the value definition
- Retrieve the value definition directly without retrieving the template
- Set the value for the definition to use a specific template instance definition

The `TreeConditionTermDefinition` class provides methods that support the following:

- Retrieve the value definition templates defined for the condition node
- Retrieve the user presentation of the condition term

The `TreeConditionDefinition` class provides methods that support the following:

- Retrieve the term definition for the condition node
- Retrieve the condition value definitions for the condition node from all of the case edges
- Retrieve the orientation (row or column)

The `TreeConditionValueDefinition` class provides methods that support the following:

- Retrieve the specific template instance expression defined for the value
- Retrieve the user

The `Template` class provides methods that support the following:

- Retrieve the system ID for the template
- Retrieve the name of the template
- Retrieve the parameters defined for the template
- Retrieve the presentation for the template

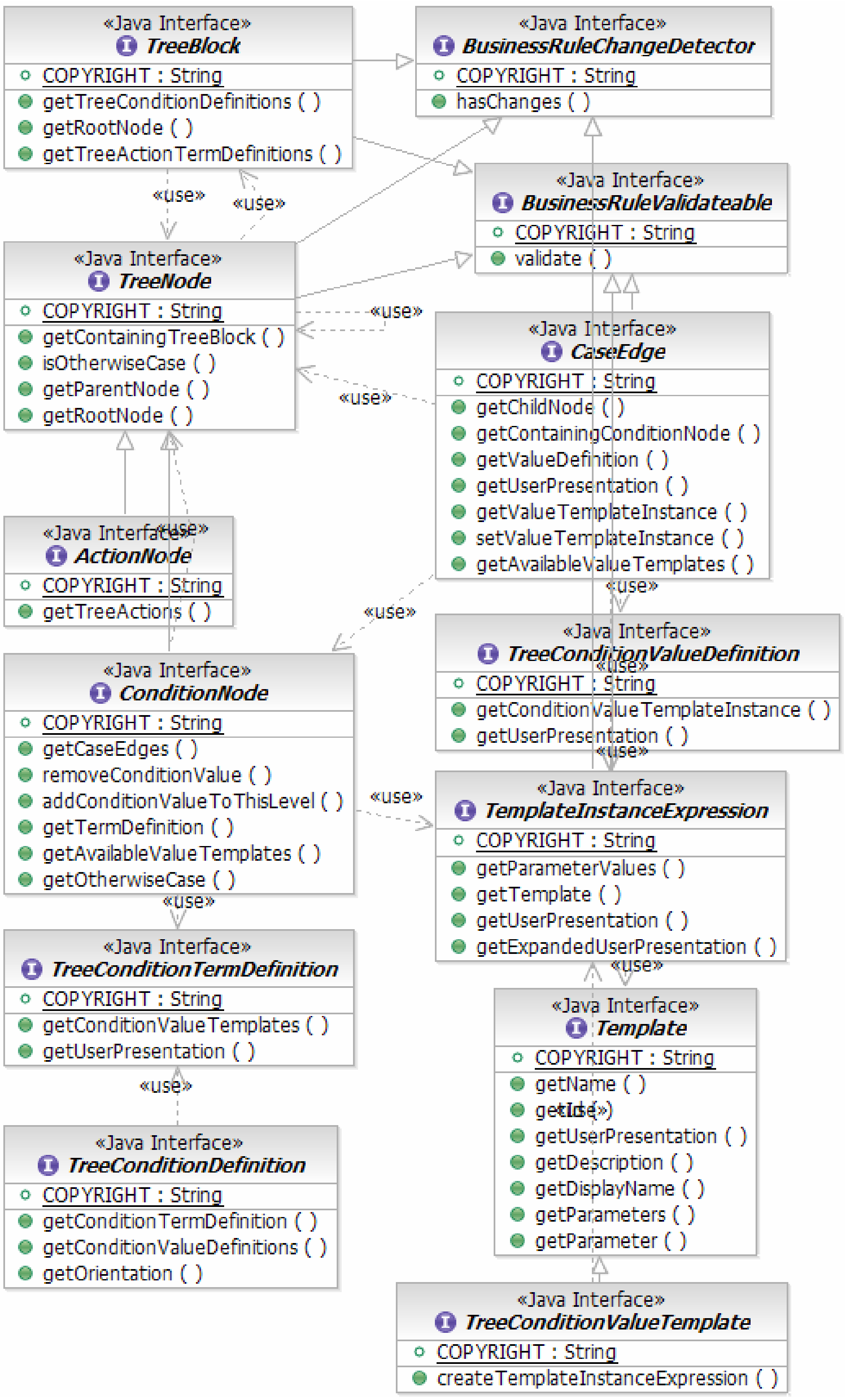
The `TreeConditionValueTemplate` class provides a method that supports the following:

- Create a new template condition value instance

The `TemplateInstanceExpression` class provides methods that support the following:

- Retrieve the parameters for the template instance
- Retrieve the template (`TreeConditionValueTemplate` in the case of a case edge in a decision table) that was used to define the instance

Figure 7. Class diagram for `TreeNode` and related classes


«Java Interface»
TreeBlock
COPYRIGHT : String
getTreeConditionDefinitions ( )
getRootNode ( )
getTreeActionTermDefinitions ( )
«Java Interface»
BusinessRuleChangeDetector
COPYRIGHT : String
hasChanges ( )
«Java Interface»
BusinessRuleValidateable
COPYRIGHT : String
validate ( )
«Java Interface»
TreeNode
COPYRIGHT : String
getContainingTreeBlock ( )
isOtherwiseCase ( )
getParentNode ( )
getRootNode ( )
«Java Interface»
CaseEdge
COPYRIGHT : String
getChildNode ( )
getContainingConditionNode ( )
getValueDefinition ( )
getUserPresentation ( )
getValueTemplateInstance ( )
setValueTemplateInstance ( )
getAvailableValueTemplates ( )
«Java Interface»
ActionNode
COPYRIGHT : String
getTreeActions ( )
«Java Interface»
TreeConditionValueDefinition
COPYRIGHT : String
getConditionValueTemplateInstance ( )
getUserPresentation ( )
«Java Interface»
ConditionNode
COPYRIGHT : String
getCaseEdges ( )
removeConditionValue ( )
addConditionValueToThisLevel ( )
getTermDefinition ( )
getAvailableValueTemplates ( )
getOtherwiseCase ( )
«Java Interface»
TemplateInstanceExpression
COPYRIGHT : String
getParameterValues ( )
getTemplate ( )
getUserPresentation ( )
getExpandedUserPresentation ( )
«Java Interface»
TreeConditionTermDefinition
COPYRIGHT : String
getConditionValueTemplates ( )
getUserPresentation ( )
«Java Interface»
Template
COPYRIGHT : String
getName ( )
getId ( )
getUserPresentation ( )
getDescription ( )
getDisplayName ( )
getParameters ( )
getParameter ( )
«Java Interface»
TreeConditionDefinition
COPYRIGHT : String
getConditionTermDefinition ( )
getConditionValueDefinitions ( )
getOrientation ( )
«Java Interface»
TreeConditionValueTemplate
COPYRIGHT : String
createTemplateInstanceExpression ( )
«use»

When a new case edge is added to a condition node, the new case edge must use a template to define the value. For example if a new case edge of “bronze” was to be added for checking ‘status’, the appropriate template (`TreeConditionValueTemplate`) would need to be used to create a new TemplateInstanceExpression, setting the parameter value to “bronze”.

When a new case edge is added, it will also have a child condition node added to it automatically. This child condition node will contain case edges which are based on the case edge definitions that have been defined for condition nodes at that same level. If templates or hard-coded values are used in case edges, they will then be used in the child condition node’s case edges as well. The child condition node that is added automatically will also have its own child condition nodes created automatically. These child condition nodes will also have child condition nodes and so on until all levels of condition nodes have been recreated.

Besides the condition nodes, a decision table and more specifically tree block, also contains a level of action nodes (`ActionNode`). The action nodes are leaf nodes and reside at the end of the branch of condition nodes and the case edges. Should all of the condition values in a line of case edges resolve to true, an action node is reached. The action node will have at least one action (`TreeAction`) defined. For the action, there will be a term definition and value definition. Just as with the condition nodes, the term definition (`TreeActionTermDefinition`) is the left-hand side of the expression and the value definition (`TemplateInstanceExpression`) is the right-hand side of the expression. For example, for the different condition nodes which were checking on the status, there might be actions to define the discount. If the condition was (status == “gold”), the action can be (discountValue = 0.90). For the action the “discountValue” would be the term definition and the “= 0.90” would be the value definition.

The term definition of a tree action is actually shared with other tree actions in other action nodes. Since every branch of case edges reaches an action, the same term definitions are used. The value definitions however, can be different per tree action and action node. For example the discountValue for a status of “gold” can be “0.90”; however the “discountValue” for a status of “silver” can be “0.95”.

Action nodes can have multiple tree actions which have a separate term definition and separate value definition. For example, if the discount was being determined for a rental car, besides setting the discountValue, you can also want to assign a specific level of car. Another tree action could be created to set the “carSize” term to “full size” if the status was “gold” as well as set the “discountValue” to “0.90”.

The value definition in a tree action can be created from a template (`TreeActionValueTemplate`). The template definition contains an expression (`TemplateInstanceExpression`) which has the parameters for the expression.

Besides changing the parameters, the entire value definition can be modified with a new value definition instance which is created with another template which was defined for the tree action.

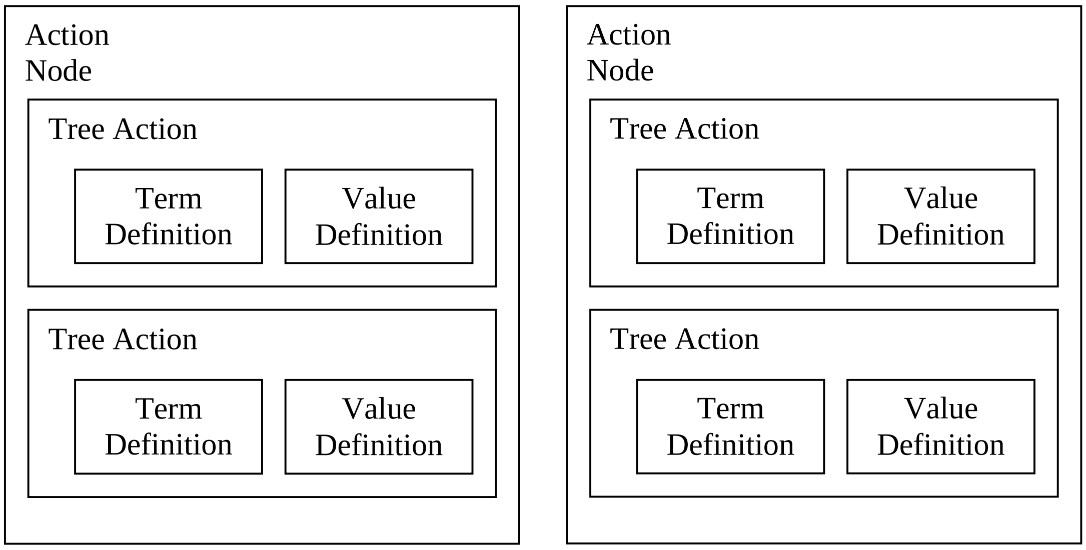
If a value definition is not created with a template, it cannot be changed. For client applications, the user presentation can be used in display if it was specified at author time.

For term definitions in tree actions, if a user presentation has been specified, it can also be used by client applications.

When a new case edge is added to a condition node and the different child condition nodes are created, action nodes will also be created. Unlike the child condition nodes and case edges which are created based on the definition of the case edges already defined for that level, action nodes do not automatically inherit an existing design. Only empty placeholder TreeActions are created in the action node. A template (`TreeActionValueTemplate`) must be used to complete the action definition by creating a TemplateInstanceExpression for at least one term definition for the action node. Until the tree action is set with a `TemplateInstanceExpression`, the tree action will have null values specified for the user presentation value and template instance value.

When creating a new condition that results in new `ActionNodes`, the action nodes will be added to the right of existing actions for the immediate parent condition node. For example if a status of "ruby" is added to the decision table and should have a specific discount, the condition to check the status is added at the right of "gold", "silver", and "bronze". The action node for the discount for "ruby" will be added to the right of the action nodes that correspond to the "gold", "silver" and "bronze" case edges.

When setting new tree actions for action nodes, an algorithm that looks to the right-most action node at the lowest case edge will return the action node with an empty tree action. The tree action can also be checked that it has null values for the user presentation value and `template instance value`. Once the tree action is obtained, it can be set with the correct instance of a `TreeActionValueTemplate.`



The `ActionNode` class provides a method that supports the following:

- Retrieve a list of the defined tree actions

The `TreeAction` class provides methods that support the following:

- Retrieve a list of the available value templates defined for the tree action
- Retrieve the term definition
- Retrieve the value template instance defined for the tree action
- Retrieve the user presentation for the value if a value template was not used
- Check if the action is a SCA service invocation (`isValueNotApplicable` method)
- Replace the value template instance with a new instance

The `TreeActionTermDefinition` class provides methods that support the following:

- Retrieve the user presentation for the term value definition
- Retrieve a list of the value templates available for the tree action
- Check if the action is a SCA service invocation (`isTermNotApplicable` method)

The `Template` class provides methods that support the following:

- Retrieve the system ID for the template
- Retrieve the name of the template
- Retrieve the parameters defined for the template

- Retrieve the presentation for the template

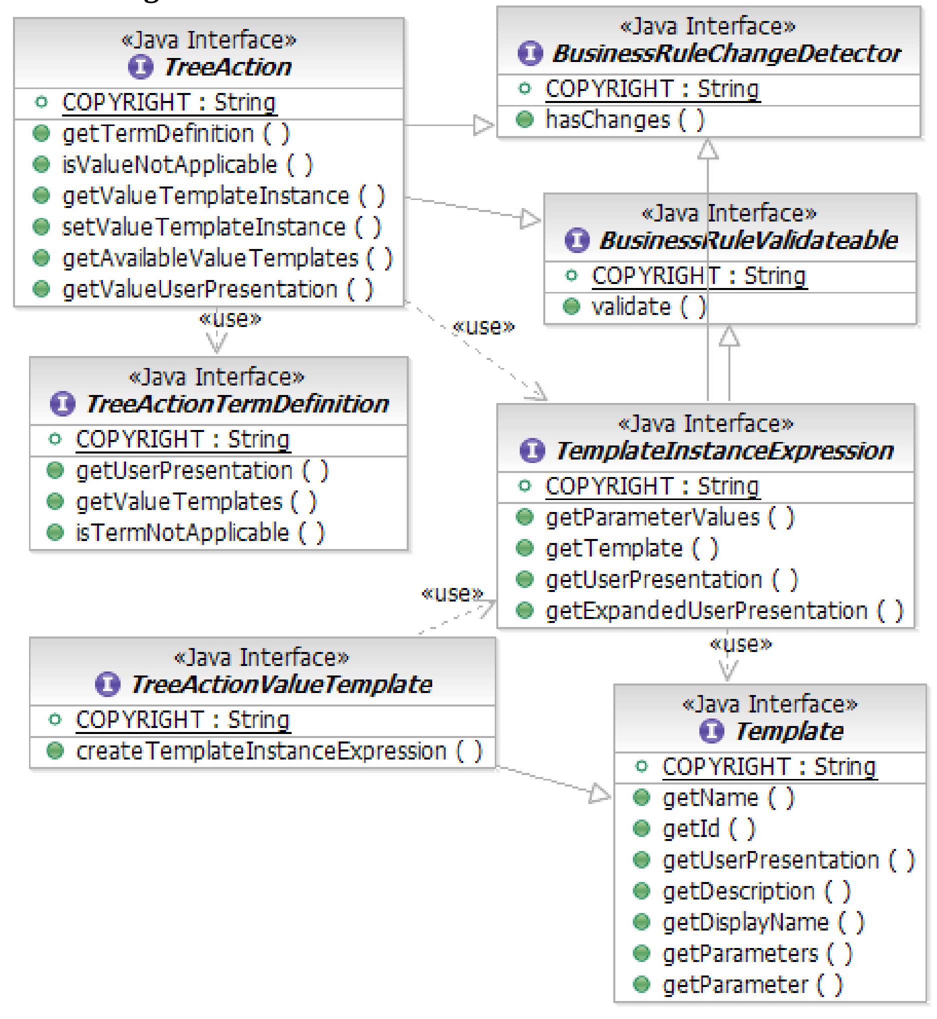
The `TreeActionValueTemplate` class provides a method that supports the following:

- Create a new value template instance from the template definition

The `TemplateInstanceExpression` class provides methods that support the following:

- Retrieve the parameters for the template instance
- Retrieve the template (`TreeActionValueTemplate` in the case of a tree action in a decision table) which was used to define the instance

Figure 8. Class diagram for `TreeAction` and related classes



The definition of an init rule for a decision table follows the same structure as a rule in a rule set. The init rule can be defined with a template (`DecisionTableRuleTemplate`).

If an init rule was not created at authoring time, it can not be added once the rule is deployed.

The `Rule` class provides methods that support the following:

- Retrieve the name of the rule
- Retrieve the user presentation for the rule
- Retrieve the user presentation for the rule with the different parameters for the rule filled in

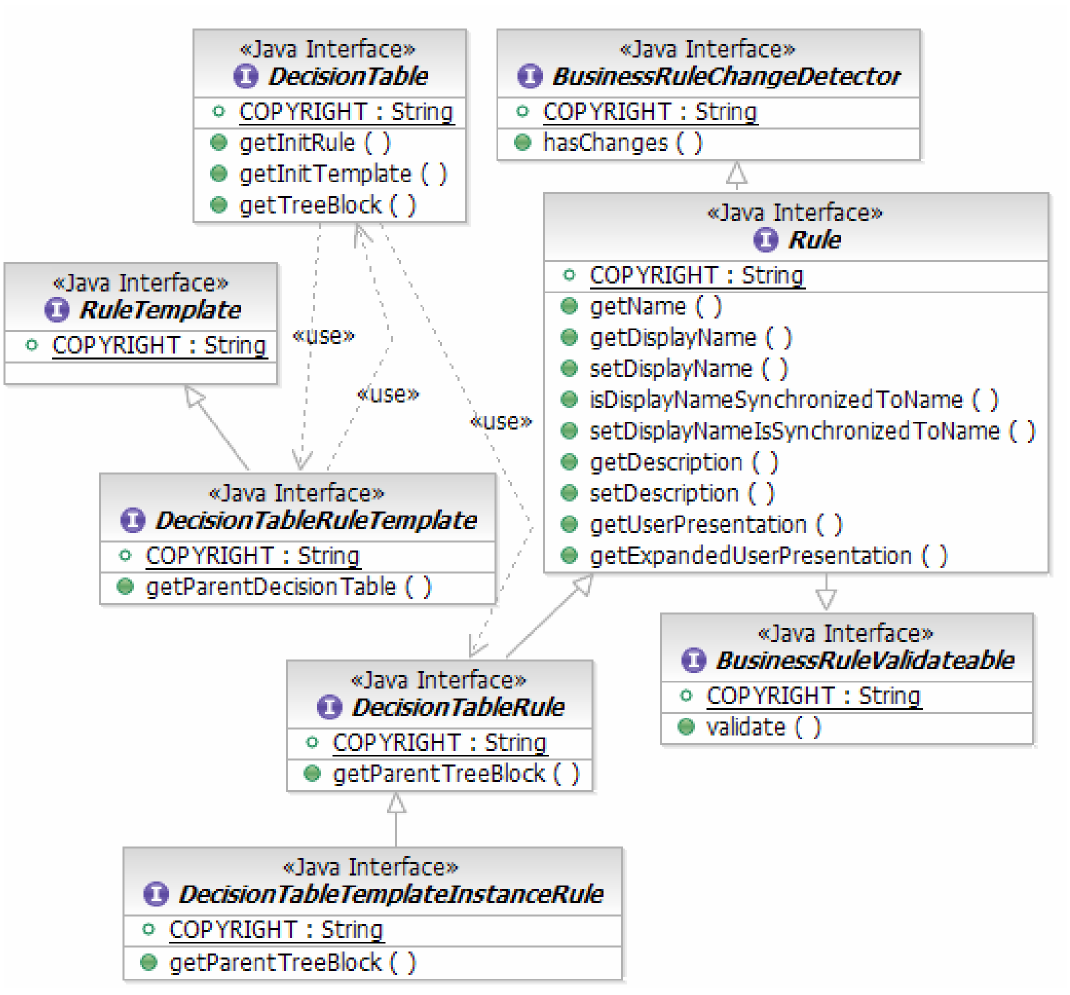
The `DecisionTableRule` class provides a method that supports the following:

- Retrieve the tree block containing the init rule

The `DecisionTableRuleTemplate` class provides a method that supports the following:

- Retrieve the decision table containing the template

Figure 9. Class diagram for `DecisionTableRule` and related classes



2.7 Templates and Parameters

Templates in rule sets and decision tables are based off of a common definition. Templates have parameters and a user presentation. The template parameter values are defined to allow for changes to be made to the rule once it has been deployed. The user presentation value provides a string value which can be used for displaying the rule and different parameters in a user-friendly manner. The user presentation, which is a string, has placeholders to allow for the different parameter values to be replaced and displayed correctly. The placeholders are in the format {<parameter index>}. For example, if the presentation string for the init rule is “Base discount is {0} %”, the placeholder {0} could be substituted with the parameter value. The presentation string cannot be changed for the rule or the template definition. The placeholder values however, can be modified with the parameter values in a client application per the definition of the template. The different templates include a convenience method (getExpandedUserPresentation) that returns a string which has all of the parameter values correctly placed in the string.

All parameter values have a specific data type, however when retrieving and setting a parameter value, a string object is used. The parameter value can be treated as string when substituting the value into the user presentation and also when setting the parameter with a new value. The parameter is converted to the correct data type at runtime in order to correctly issue the rule at execution time. During validation, the parameter value will be compared to the data type to insure it is correct. For example, if a parameter is of type `boolean` and is set to “T”, validation will not recognize this value and will return a problem.

In the template definition, the parameter values can be restricted by constraints. The constraints can be defined as a range or an enumeration. The constraints for the parameter will be enforced when the rule is validated. If a template was not used to define the value definition, only a user presentation will be available. A value definition can not have both a template and a user presentation. Should a template be used, the presentation from the template definition is the only presentation which is available.

The `Template` class provides methods that support the following:

- Retrieve the template ID
- Retrieve the name
- Retrieve the parameters
- Retrieve the user presentation

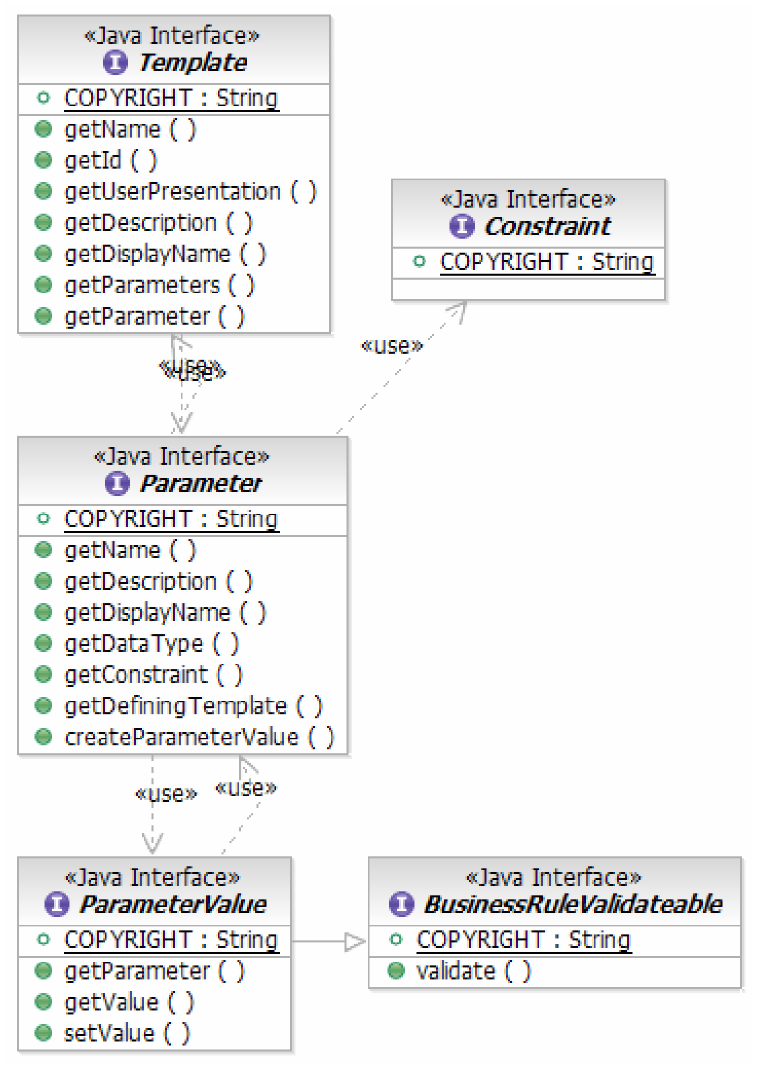
The `Parameter` class provides methods that support the following:

- Retrieve the parameter name
- Retrieve the parameter data type
- Retrieve the constraint for the parameter
- Retrieve the template defining the parameter
- Create a parameter value

The `ParameterValue` class provides methods that support the following:

- Retrieve the parameter name
- Retrieve the parameter value
- Set the parameter value

Figure 10. Class diagram for `Template` and `Parameter` and related classes



2.8 Validation

Many of the main objects have a validate method which allows for the artifacts to be checked for correctness and completeness prior to publishing the artifacts. The validation that occurs when making changes through the API classes is only a proper subset of the overall validation that occurs during serviceDeploy or when editing the artifacts in WebSphere Integration Developer. This is due to the constraints that are already placed on the business rule group in limiting which aspects are editable at runtime. The user of the classes can validate the business rule group selection table, rule set or decision table whenever it is needed (the rule group component itself is not editable at runtime). When a business rule group is published the rule group selection table, rule sets and decision tables will be validated before being published to the repository.

If the artifacts are invalid, a ValidationException will be thrown with a list of the validation problems. The different validation problems are documented in the Exception Handling section.

2.9 Tracking Changes

For all objects, a `hasChanges` method is available to check if there have been any modifications which have occurred to the object and any containing objects. This method can be used to check for changes and only publish a business rule group if it has items which changed.

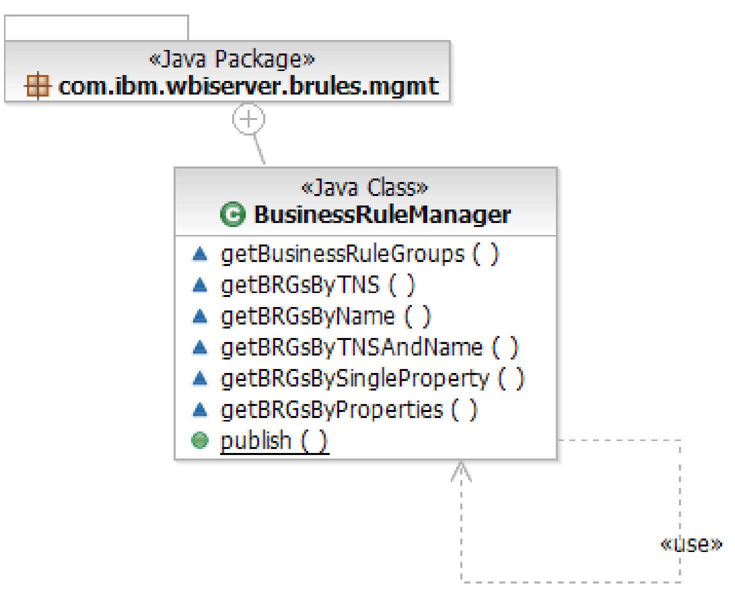
2.10 BusinessRuleManager

The `BusinessRuleManager` class is the main class for working with the business rule groups, rule sets and decision tables. The `BusinessRuleManager` has methods which allow for retrieving business rule groups by name, target namespace, or custom properties. It also has a method for publishing changes which have been made to business rule groups, rule sets, or decision tables.

The `BusinessRuleManager` class provides methods that support the following:

- Retrieve all of the business rule groups
- Retrieve business rule groups of a specific target namespace
- Retrieve business rule groups of a specific name
- Retrieve business rule groups of a specific name and target namespace
- Retrieve business rule groups which contain a specific property
- Retrieve business rule groups which contain specific properties
- Publish business rule groups

Figure 11. Class diagram for `BusinessRuleManager` and package



2.10.1 Rule Group Component Query

The rule group component can have user defined properties (name/value pairs) that can be used to narrow the list of business rule groups returned from the class. The fields that can be used in the query and in any combination are as follows:

- Business rule group component target name space
- Business rule group component name
- Property name
- Property value

Each property name can only be defined once per business rule group component.

The query function supported by this class is a small subset of the full SQL language. The user does not provide the SQL statement, but rather provides the values as parameters for a single property or a tree structure containing the information for a multi-property query in the form of nodes. There are logical operator nodes and property query nodes which all implement the `QueryNode` interface. The logical operator nodes specify the boolean operators (AND, OR, NOT). These are created through the `QueryNodeFactory`. As part of the creation of these logical operator nodes, the left and right hand sides of the operator must be specified with additional `QueryNode` classes. These nodes can either be a property query node or another logical operator node. If a property query node is passed, it will contain the property name, value and operator (EQUAL (==), NOT_EQUAL (!=), LIKE, or NOTLIKE). The overall QueryNode is parsed by the class and a query is performed on the underlying data in persistent storage.

Wildcard searches are supported when the LIKE and NOTLIKE operators are used. Both the '%' and '_' characters are supported in wildcard searches.

The ‘%’ character is used when there are an infinite number of characters which are not known or should not be considered when searching. For example if a search was to be performed for all business rule groups that have a property with a name of Department and value that begins with “North”, the value would be specified as “North%”. Another example, suppose that all Departments with a value ending in “Region” was desired. The value would be “%Region”. The ‘%’ character can also be used in the middle of a string. For example, if there were business rule groups with properties that had values of “NorthCentralRegion”, “NorthEastRegion”, and “NorthWestRegion”, a value of “North%Region” could be specified.

The ‘_’ character is used when there is a single character which is unknown or should not be considered when searching. For example, if a search for all business rule groups with Department properties with values of “Dept1North”, “Dept2North”, “Dept3North”, and “Dept4North” was desired, a value of “Dept_North” could be specified and all 4 of the business rule groups with these properties will be returned. The ‘_’ character can be used multiple times in a search value with each instance indicating a single character to ignore. The ‘_’ character can be used at the beginning or end of a value. For example if two characters were to be ignored in a value, two ‘_’ could be used such as “Dept__outh”.

In order to treat ‘%’ and ‘_’ as literal characters and not wildcards a ‘\’ escape character must be specified preceding the ‘%’ or ‘_’. For example if the property name was “%Discount”, in order to use this in a query, “\%Discount” would need to be specified. If the ‘\’ character is to be used as a literal character, another ‘\’ escape character must be used such as “Orders\\Customer”. If a single ‘\’ character is found without a following ‘%’, ‘_’, or ‘\’, an IllegalArgumentException will be thrown.

Wildcard characters can only be used on the left-hand side (property value). Wildcard characters can not be used in property name.

During searches on the value of a specific property or a search for values which do not match a property, the absence of a property causes the artifact to be ignored from consideration in the search. For example, if there are 3 business rule groups (A, B, and C) and only two (A and B) have a property named “Department” with different values (“Accounting” and “Shipping” respectively) a search for all business rule groups which do not have a “Department” property of “Accounting” will only return the business rule group which has the “Department” property defined but does not equal “Accounting” (business rule group B). The business rule group (C) which does not have the “Department” property, will not be returned as it does not have the property defined in any way.

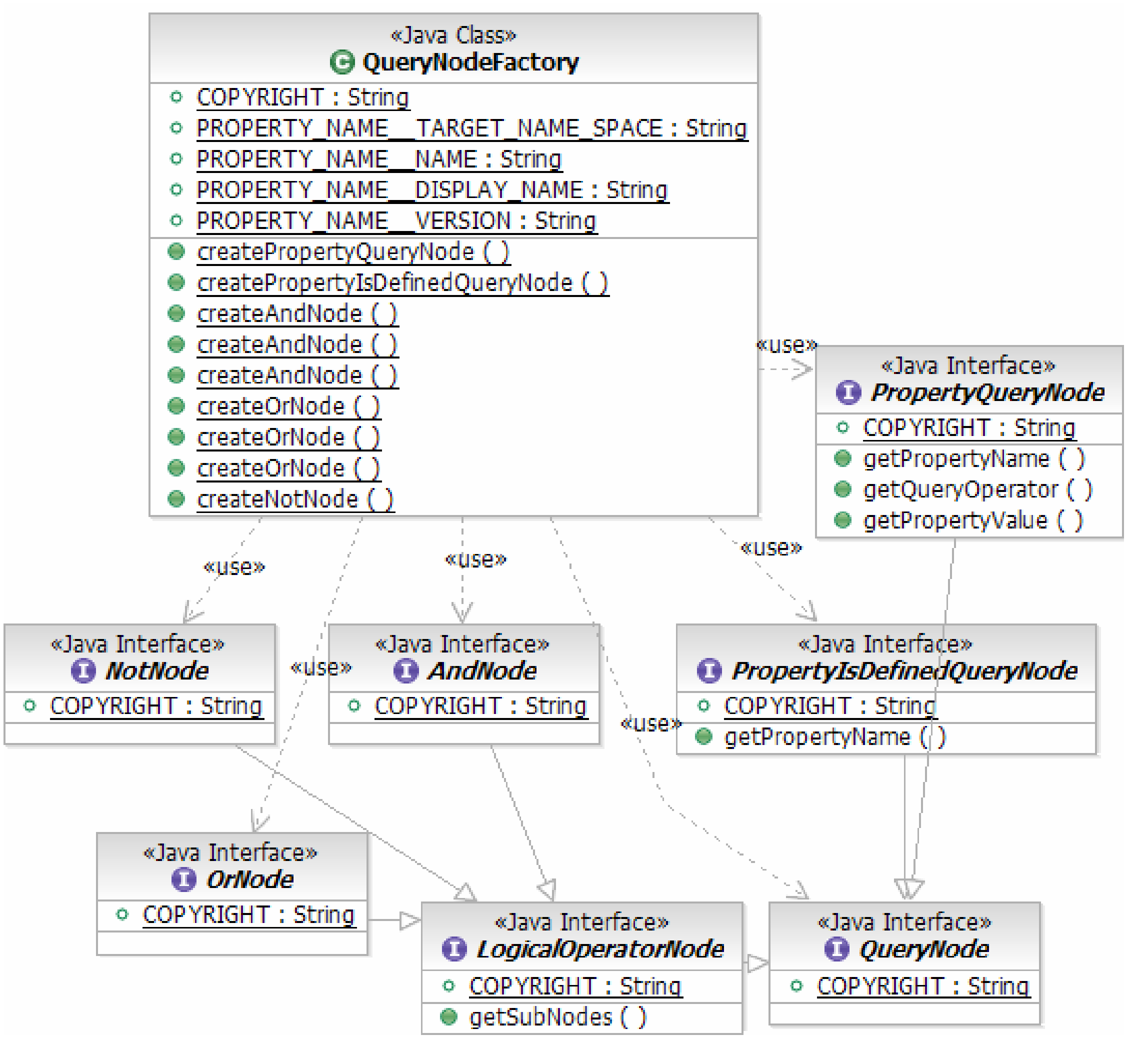
When using properties for searching, there are two special properties named *IBMSystemName* and *IBMSystemTargetNameSpace* which can be used for searching based on the name and namespace of an artifact. These values can also be retrieved with the `getName` and `getTargetNameSpace` methods.

The class supports the following methods for query:

```
List getBRGsByTNS (string tNSName, Operator op, int skip, int
threshold)

List getBRGByName(string Name, Operator op, int skip, int threshold)

List getBRGsByTNSAndName (string tNSName, Operator, tNSOp, string
name,                                                 Operator
nameOp, int skip, int threshold)

List getBRGsBySingleProperty (string propertyName, string
propertyValue, Operator op, int skip, int threshold)

List getBRGsByProperties (QueryNode queryTree, int skip, int
threshold)
```

The 'skip' and 'threshold' parameters provide the user the capability of fetching a partial result list up to the specified threshold. A value of zero for both of these parameters will return the full result list. The cursor is not maintained in the result set from a query call. If a skip value is used, it is possible that additions or deletions could have been made to the result set such that a subsequent request will return business rule groups which were in an earlier result set.

Figure 12. Class diagram for `QueryNodeFactory` and related classes



The nodes in the tree allow the user to specify a search expression using the boolean operators, wild cards (% and escape) and the property/value pair. The operator is only valid for the values, the operator for the property is always equals (==).

2.10.2 Publishing

Publishing of business rule changes is done at the business rule group component level. The user can publish 1…n business rule group components. Before a publish operation is performed, a validate action is performed on the business rule group and the different objects contained in the business rule group (operation selection table, rule sets, decision table, and so on). Each publish request will occur within a single transaction and if any exceptions are encountered during validation or database publishing the transaction is rolled back and no changes for any business rule group are published to the repository. This allows changes that are dependent on each other within a single component (for example, operation selection table and a rule set) or dependencies between components to occur within one atomic operation.

At publishing time, a check will be made to insure that the items which are to be published have not been changed by another transaction. To reduce the possibilities of a conflict, the publish method will give the user the ability to choose to publish all artifacts whether they are changed or not or only those artifacts that were changed within the business rule group. The default behavior will be to publish all artifacts. If the option is set to publish all artifacts and another transaction had changed the artifacts in the meantime, a `ChangeConflictException` will be thrown. Specifying to only publish those artifacts which have changed will reduce the chance of conflict. Publishing only those artifacts that were changed could result in two users pushing changes to the repository for two different artifacts within a business rule group (for example, two rule sets) which could introduce incompatible changes within the business rule group. Because this potential situation, this option should be used with caution.

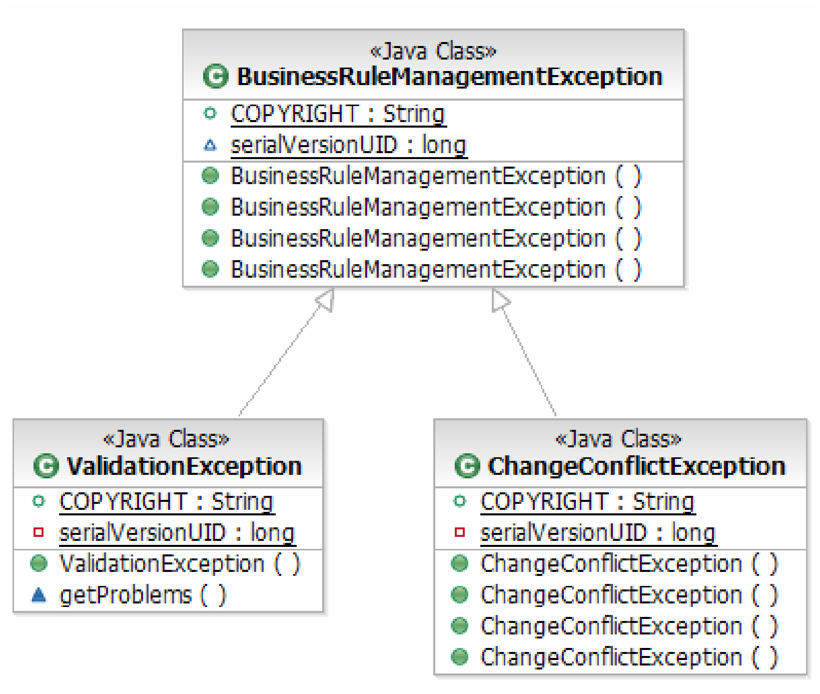
2.11 Exception Handling

Exceptions can occur when validation is called on an artifact or when an artifact is published. When a validation error occurs, the `ValidationException` is thrown with a list of problems. If there is a problem during publishing due to another transaction publishing the same artifacts, a `ChangeConflictException` is thrown. Anytime another transaction is detected as changing an artifact, the `ChangeConflictException` exception is thrown.

There is a `SystemPropertyNotChangeableException` which is thrown if a property which duplicates a system property name is attempted to be changed. System properties cannot be changed.

There is a `ChangesNotAllowedException` which is thrown if a set operation is attempted on an artifact as it is being published.

Figure 13. Class diagram for `BusinessRuleManagementException` and related classes


«Java Class»
BusinessRuleManagementException
COPYRIGHT : String
serialVersionUID : long
BusinessRuleManagementException ( )
BusinessRuleManagementException ( )
BusinessRuleManagementException ( )
BusinessRuleManagementException ( )
«Java Class»
ValidationException
COPYRIGHT : String
serialVersionUID : long
ValidationException ( )
getProblems ( )
«Java Class»
ChangeConflictException
COPYRIGHT : String
serialVersionUID : long
ChangeConflictException ( )
ChangeConflictException ( )
ChangeConflictException ( )
ChangeConflictException ( )

2.11.1 Business Rule Group Problems

Problems which can occur when a business rule group is validated or an attempt to publish the business rule group is made and a portion of the business rule group is not valid.

2.11.1.1 ProblemBusRuleNotInAvailTargetList

Problem that occurs when a rule is specified as the default business rule for an operation selection table but the rule artifact is not in the list of available targets for that operation. A valid business rule from the list of available targets on the operation should be specified to avoid this problem.

2.11.1.2 ProblemDuplicatePropertyName

This problem occurs when a property is attempted to be created which is a duplicate of a system or user-defined property on a business rule group. A unique property name should be used to avoid this problem.

2.11.1.3 ProblemOperationContainsNoTargets

Problem that occurs when an operation does not have a default rule destination or any scheduled rule destinations set. The operation should be set with at least one rule destination as either the default or at a scheduled time to avoid this problem.

2.11.1.4 ProblemOverlappingRanges

Problem that occurs when an operation selection record has a start date or end date which overlaps with the range of another operation selection record start date and end date. This overlap in date ranges prevents the business rule runtime from finding the correct rule destination to invoke. The start date or end date of the other operation selection records for an operation should be checked to insure there is not an overlap to avoid this problem.

2.11.1.5 ProblemStartDateAfterEndDate

This problem occurs when the start date for an operation selection record is after the end date for that selection record. This problem can occur for any operation selection record except the default record which does not have a start date or end date. Specify a start date after the end date on an operation selection record to avoid this problem

2.11.1.6 ProblemTargetBusRuleNotSet

Problem that occurs when an operation selection record has a rule specified which is not in the list of available target rules. A rule from the available target list should be specified to avoid this problem.

2.11.1.7 ProblemTNSAndNameAlreadyInUse

Problem that occurs when a new business rule is created with a target name space and name which is already in use by a rule set or decision table. A check is done on all rule sets and decision tables associated with the current business rule group as well as any rule artifact stored in the repository. A different target name space or name should be used to avoid this problem.

2.11.1.8 ProblemWrongOperationForOpSelectionRecord

Problem that occurs when a new operation selection record is added to an operation selection record list and the operation of the new record does not match the operation of the records in the list. A new operation should be created using the `newOperationSelectionRecord` method on the correct operation selection record list object to avoid this problem.

2.11.2 Rule set and Decision Table Problems

2.11.2.1 ProblemInvalidBooleanValue

Problem that occurs when a parameter for a rule template in a rule set or an action value or condition value in decision table receives a different value than “true” or “false” for a parameter of type Boolean. Examples of incorrect parameter values would be “T” or “F”. Use values of “true” or “false” when working with a parameter of type Boolean to avoid this problem.

2.11.2.2 ProblemParmNotDefinedInTemplate

Problem that occurs when a value is specified for a template parameter and the parameter is not defined in the list of valid parameters for the template. The parameters should be checked prior setting in the template. It can occur for RuleTemplate, TreeActionValueTemplate, or TreeConditionValueTemplate templates.

2.11.2.3 ProblemParmValueListContainsUnexpectedValue

Problem that occurs when valid parameters are passed with a template, however, there are too many parameters for the parameter. The number of parameters should be reduced. It can occur for RuleTemplate, TreeActionValueTemplate, or TreeConditionValueTemplate templates.

2.11.2.4 ProblemRuleBlockContainsNoRules

This problem occurs when all rules in a rule block in a rule set are removed and the rule set is attempted to be validated or published. The rule block in a rule set must have at least one rule.

2.11.2.5 ProblemTemplateNotAssociatedWithRuleSet

Problem that occurs when a rule is attempted to be added to a rule set and the rule was created with a template which is not defined with that rule set. When creating a new rule, a template which has been defined in the rule set should be used to avoid this problem.

2.11.2.6 ProblemRuleNameAlreadyInUse

Problem that occurs when a rule is attempted to be added to a rule block in a rule set and it has the same name as an existing rule in the rule block. The names of the rules should be checked prior to adding a new rule to avoid this problem.

2.11.2.7 ProblemTemplateParameterNotSpecified

This problem occurs when a parameter is not included when a template is updated for a rule in a rule set or action or condition value in a decision table. All parameters for a template should be specified to avoid this problem.

2.11.2.8 ProblemTypeConversionError

This problem occurs when a parameter for a template cannot be converted to the appropriate type All parameters are treated as string objects and then converted to the parameter type (boolean, byte, short, int, long, float, and double). If the parameter value string cannot be converted to the specified type for this parameter, then this error occurs. To avoid this problem, a string that can be converted to the parameter’s type (boolean, byte, short, int, long, float, and double) should be specified.

2.11.2.9 ProblemValueViolatesParmConstraints

This problem occurs when a parameter is not within the enumeration or range of values which have been defined within the template for that parameter. This problem can occur for parameters restricted with enumerations or ranges in rule templates in a rule set or action value or condition value templates in a decision table. A value which is within the enumeration should be used to avoid this problem.

2.11.2.10 ProblemInvalidActionValueTemplate

Problem that occurs when a template instance is attempted to be set on the value definition in a tree action but the corresponding template is not available to that tree action. Use the correct template to create a value definition in a tree action in order to avoid this problem.

2.11.2.11 ProblemInvalidConditionValueTemplate

Problem that occurs when a template instance is attempted to be set on the condition definition in a case edge but the corresponding template is not available to that case edge. Use the correct template to create a condition definition in a case edge in order to avoid this problem.

2.11.2.12 ProblemTreeActionIsNull

This problem occurs when a new condition value is created and an action is not set with a template instance. Using a template from the ActionNode, create a new template instance and set it in the list of `TreeAction`s.

2.12 Authorization

The classes do not support any level of authorization. It is up to the client application using the classes to add its own form of authorization.

3.0 Examples

A number of examples are provided that show how the different classes can be used to retrieve business rule groups and to make modifications to rule sets and decision tables. The examples are provided in a project interchange file (ZIP) that can be imported into WebSphere Integration Developer where they can be browsed and reused.

The project interchange contains a number of projects.

BRMgmtExamples – Module project with business rules artifacts that are used in the various examples.
BRMgmt – Java project with the examples located in the com.ibm.websphere.sample.brules.mgmt package.
BRMgmtDriverWeb – Web project with interface for executing the samples.

The examples are also provided as an EAR file (BRMgmtExamples.ear) that can be issued once after installed into WebSphere Process Server. A Web interface is provided with the examples. The Web interface is purposely simple as the examples focus on using the classes to retrieve artifacts, make modifications, and publish changes. It is not meant to be a high-functioning Web interface. The classes can however, be easily used to build robust Web interfaces or used in other Java applications focused on modifying the business rules.

The example application can be installed on WebSphere Process Server v6.1 and the index page can be accessed at:

http://<hostname>:<port>/BRMgmtDriverWeb/ (For example, http://localhost:9080/BRMgmtDriverWeb/)

As the examples are issued, changes will be made to the rule artifacts. If all examples are issued, the application will need to be reinstalled to see the same results for all examples again.

The examples are explained in detail with complete code samples as well as the result as displayed in a Web browser.

A number of additional classes were created in order to perform common operations and assist with displaying the information within the example Web application. See the appendix for more information on the `Formatter` and `RuleArtifactUtility` classes.

To fully understand these examples, a study of the different artifacts within WebSphere Integration Developer will greatly help.

3.1 Example 1: Retrieve and print all business rule groups

This example will retrieve all business rule groups and print out the attributes, the properties, and the operations for each business rule group.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.Iterator;
import java.util.List;
```

For the business rules management classes, be sure to use those classes in the com.ibm.wbiserver.brules.mgmt package and not the com.ibm.wbiserver.brules package or other package. These other packages are for IBM internal classes.

```
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import
com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.Property;
import com.ibm.wbiserver.brules.mgmt.PropertyList;

public class Example1 {

static Formatter out = new Formatter();
static public String executeExample1()
{
      try
      {
            out.clear();
```

The BusinessRuleManager class is the main class to retrieve business rule groups and to publish changes to business rule groups. This includes working with and changing any rule artifacts such as rule sets and decision tables. There are a number of methods on the BusinessRuleManager class that simplify the retrieval of specific business rule groups by name and namespace and properties.

```
            // Retrieve all business rule groups
            List<BusinessRuleGroup> brgList = BusinessRuleManager
                        .getBusinessRuleGroups(0, 0);

            Iterator<BusinessRuleGroup> iterator = brgList.iterator();

            BusinessRuleGroup brg = null;

            // Iterate through the list of business rule groups
            while (iterator.hasNext())
            {
                  brg = iterator.next();
                  // Output attributes for each business rule group
                  out.printlnBold("Business Rule Group");
```

The basic attributes of the business rule group can be retrieved and displayed.

```
                out.println("Name: " + brg.getName());
                out.println("Namespace: " +
                brg.getTargetNameSpace());
                out.println("Display Name: " +
                brg.getDisplayName());
                out.println("Description: " + brg.getDescription());
                out.println("Presentation Time zone: "
                            + brg.getPresentationTimezone());
                out.println("Save Date: " + brg.getSaveDate());
```

The properties for the business rule group can also be retrieved and modified.

```
                PropertyList propList = brg.getProperties();

                Iterator<Property> propIterator =
                propList.iterator();
                Property prop = null;
                // Output property names and values
                while (propIterator.hasNext())
                {
                      prop = propIterator.next();
                      out.println("Property Name: " +
                      prop.getName());
                      out.println("Property Value: " +
                      prop.getValue());
                }
```

The operations for the business rule group are also available and are the way to retrieve the business rule artifacts such as rule sets and decision tables.

```
                List<Operation> opList = brg.getOperations();

                Iterator<Operation> opIterator = opList.iterator();
                Operation op = null;
                // Output operations for the business rule group
                while (opIterator.hasNext())
                {
                      op = opIterator.next();
                      out.println("Operation: " + op.getName());
                }
                out.println("");
          }
    } catch (BusinessRuleManagementException e)
    {
          e.printStackTrace();
          out.println(e.getMessage());
    }
    return out.toString();
  }
}
```

Web browser output for example 1.

Executing example1

Business Rule Group
Name: ApprovalValues
Namespace: http://BRSamples/com/ibm/websphere/sample/brules
Display Name: ApprovalValues
Description: null
Presentation Time zone: LOCAL
Save Date: Sun Jan 06 17:56:51 CST 2008
Property Name: IBMSystemVersion
Property Value: 6.1.0
Property Name: Department
Property Value: Accounting
Property Name: RuleType
Property Value: regulatory
Property Name: IBMSystemTargetNameSpace
Property Value: http://BRSamples/com/ibm/websphere/sample/brules
Property Name: IBMSystemName
Property Value: ApprovalValues
Property Name: IBMSystemDisplayName
Property Value: ApprovalValues
Operation: getApprover

Business Rule Group
Name: ConfigurationValues
Namespace: http://BRSamples/com/ibm/websphere/sample/brules
Display Name: ConfigurationValues
Description: null
Presentation Time zone: LOCAL
Save Date: Sun Jan 06 17:56:51 CST 2008
Property Name: IBMSystemVersion
Property Value: 6.1.0
Property Name: Department
Property Value: General
Property Name: RuleType
Property Value: messages
Property Name: IBMSystemTargetNameSpace
Property Value: http://BRSamples/com/ibm/websphere/sample/brules
Property Name: IBMSystemName
Property Value: ConfigurationValues
Property Name: IBMSystemDisplayName
Property Value: ConfigurationValues
Operation: getMessages

Business Rule Group
Name: DiscountRules
Namespace: http://BRSamples/com/ibm/websphere/sample/brules
Display Name: DiscountRules
Description: null
Presentation Time zone: LOCAL
Save Date: Sun Jan 06 17:56:51 CST 2008
Property Name: Department
Property Value: Accounting
Property Name: IBMSystemVersion
Property Value: 6.1.0
Property Name: RuleType
Property Value: monetary
Property Name: IBMSystemTargetNameSpace
Property Value: http://BRSamples/com/ibm/websphere/sample/brules
Property Name: IBMSystemName
Property Value: DiscountRules
Property Name: IBMSystemDisplayName
Property Value: DiscountRules
Operation: calculateOrderDiscount
Operation: calculateShippingDiscount

3.2 Example 2: Retrieve and print business rule groups, rule sets and decision tables

Besides the function in example 1, this example will print out the selection table for each operation and then the default business rule destination (either rule set or decision table) and the other business rules scheduled for the operation. It prints out both rule sets and decision tables.

The majority of the example is the same, but provided for completeness.

```
import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.OperationSelectionRecord;
import com.ibm.wbiserver.brules.mgmt.OperationSelectionRecordList;
import com.ibm.wbiserver.brules.mgmt.Property;
import com.ibm.wbiserver.brules.mgmt.PropertyList;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSet;

public class Example2
{
     static Formatter out = new Formatter();

     static public String executeExample2()
     {
           try
           {
                 out.clear();
```

A specific business rule group is retrieved by name for this example.

```
                 // Retrieve all business rule groups
                 List<BusinessRuleGroup> brgList = BusinessRuleManager
                             .getBRGsByName("DiscountRules",
                             QueryOperator.EQUAL, 0, 0);

                 Iterator<BusinessRuleGroup> iterator = brgList.iterator();

                 BusinessRuleGroup brg = null;
                 // Iterate through the list of business rule groups
                 while (iterator.hasNext())
                 {
                       brg = iterator.next();
                       // Output attributes for each business rule group
                       out.printlnBold("Business Rule Group");
```

```
out.println("Name: " + brg.getName());
out.println("Namespace: " +
brg.getTargetNameSpace());
out.println("Display Name: " +
brg.getDisplayName());
out.println("Description: " + brg.getDescription());
out.println("Presentation Time zone: "
            + brg.getPresentationTimezone());
out.println("Save Date: " + brg.getSaveDate());

PropertyList propList = brg.getProperties();

Iterator<Property> propIterator =
propList.iterator();
Property prop = null;
// Output property names and values
while (propIterator.hasNext())
{
      prop = propIterator.next();
      out.println("Property Name: " +
      prop.getName());
      out.println("Property Value: " +
      prop.getValue());
}
```

For each operation, a selection table has a list of the different rule artifacts and the schedule on when they are active. A default business rule can be specified for each operation. There is no requirement that a default business rule be specified or that there is a scheduled business rule, however there must be at least a default business rule or one scheduled business rule. Because of this support, it is best to check for `null` before using the default business rule or check the size of the `OperationSelectionRecordList`.

```
List<Operation> opList = brg.getOperations();

Iterator<Operation> opIterator = opList.iterator();
Operation op = null;
out.println("");
out.printlnBold("Operations");
// Output operations for the business rule group
while (opIterator.hasNext())
{
      op = opIterator.next();
      out.printBold("Operation: ");
      out.println(op.getName());


      // Retrieve the default business rule for the
      operation
      BusinessRule defaultRule =
      op.getDefaultBusinessRule();
      // If the default rule is found, print out the
      business rule
      // using the appropriate method for rule type
      if (defaultRule != null)
```

```
                        {
                              out.printlnBold("Default Destination:");
```

The default business rule is of type `RuleSet` or `DecisionTable` and can be cast to the correct type in order to process the rule artifact.

```
                              if (defaultRule instanceof RuleSet)
                                    out.println(RuleArtifactUtility.pr
                                    intRuleSet(defaultRule));
                              else
                                    out.print(RuleArtifactUtility.prin
                                    tDecisionTable(defaultRule));
                        }

                        OperationSelectionRecordList
                        opSelectionRecordList = op
                                    .getOperationSelectionRecordList()
                                    ;

                        Iterator<OperationSelectionRecord>
                        opSelRecordIterator = opSelectionRecordList
                                    .iterator();

                        OperationSelectionRecord record = null;
```

The OperationSelectionRecord is composed of the rule artifact and the schedule on when the rule artifact is active.

```
                        while (opSelRecordIterator.hasNext())
                        {
                              out.printlnBold("Scheduled
                              Destination:");
                              record = opSelRecordIterator.next();

                              out.println("Start Date: " +
                              record.getStartDate()
                                          + " - End Date: " +
                                          record.getEndDate());
                              BusinessRule ruleArtifact = record
                                          .getBusinessRuleTarget();

                              if (ruleArtifact instanceof RuleSet)
                                    out.println(RuleArtifactUtility.pr
                                    intRuleSet(ruleArtifact));
                              else
                                    out.print(RuleArtifactUtility.prin
                                    tDecisionTable(ruleArtifact));
                        }
                  }
            }
            out.println("");
      } catch (BusinessRuleManagementException e)
      {
            e.printStackTrace();
            out.println(e.getMessage());
      }
```

```
        return out.toString();
    }
}
```

Web browser output for example 2.

Executing example2

Business Rule Group
Name: DiscountRules
Namespace: http://BRSamples/com/ibm/websphere/sample/brules
Display Name: DiscountRules
Description: null
Presentation Time zone: LOCAL
Save Date: Sun Jan 06 17:56:51 CST 2008
Property Name: Department
Property Value: Accounting
Property Name: IBMSystemVersion
Property Value: 6.1.0
Property Name: RuleType
Property Value: monetary
Property Name: IBMSystemTargetNameSpace
Property Value: http://BRSamples/com/ibm/websphere/sample/brules
Property Name: IBMSystemName
Property Value: DiscountRules
Property Name: IBMSystemDisplayName
Property Value: DiscountRules

Operations
Operation: calculateOrderDiscount
Default Destination:
Rule Set
Name: calculateOrderDiscount
Namespace: http://BRSamples/com/ibm/websphere/sample/brules
Rule: CopyOrder
Display Name: CopyOrder
Description: null
Expanded User Presentation: null
User Presentation: null
Rule: FreeGiftInitialization
Display Name: FreeGiftInitialization
Description: null
Expanded User Presentation: Product ID for Free Gift = 5001AE80 Quantity = 1 Cost = 0.0 Description = Free gift for discounted order
User Presentation: Product ID for Free Gift = {0} Quantity = {1} Cost = {2} Description = {3}

Parameter Name: param0
Parameter Value: 5001AE80
Parameter Name: param1
Parameter Value: 1
Parameter Name: param2
Parameter Value: 0.0
Parameter Name: param3
Parameter Value: Free gift for discounted order
Rule: Rule1
Display Name: Rule1
Description: null
Expanded User Presentation: If customer is gold status, then apply a discount of 20.0 and include a free gift
User Presentation: If customer is {0} status, then apply a discount of {1} and include a free gift
Parameter Name: param0
Parameter Value: gold
Parameter Name: param1
Parameter Value: 20.0
Rule: Rule2
Display Name: Rule2
Description: null
Expanded User Presentation: If customer.status == silver, then provide a discount of 15.0
User Presentation: If customer.status == {0}, then provide a discount of {1}
Parameter Name: param0
Parameter Value: silver
Parameter Name: param1
Parameter Value: 15.0
Rule: Rule3
Display Name: Rule3
Description: Template for non-gold customers
Expanded User Presentation: If customer.status == bronze, then provide a discount of 10.0
User Presentation: If customer.status == {0}, then provide a discount of {1}
Parameter Name: param0
Parameter Value: bronze
Parameter Name: param1
Parameter Value: 10.0

Operation: calculateShippingDiscount
Default Destination:
Decision Table
Name: calculateShippingDiscount
Namespace: http://BRSamples/com/ibm/websphere/sample/brules

Init Rule: Rule1
Display Name: Rule1
Description: null
Extended User Presentation: null
User Presentation: null

Order Sub-Total	<= 5000 <= {0} Parameter Name: param0 Parameter Value: 5000	Customer Postal Code	Between 0 and 2000 Between {0} and {1} Parameter Name: param0 Parameter Value: 0 Parameter Name: param1 Parameter Value: 2000	Order Shipping Amount	10.0 Parameter Name: param0 Parameter Value: 10.0
			Between 20001 and 40000 Between {0} and {1} Parameter Name: param0 Parameter Value: 20001 Parameter Name: param1 Parameter Value: 40000	Order Shipping Amount	15.0 Parameter Name: param0 Parameter Value: 15.0
			Between 40001 and 60000	Order Shipping	25.0 Parameter Name:

			Between {0} and {1} Parameter Name: param0 Parameter Value: 40001 Parameter Name: param1 Parameter Value: 60000	Amount	param0 Parameter Value: 25.0
			Between 60001 and 80000 Between {0} and {1} Parameter Name: param0 Parameter Value: 60001 Parameter Name: param1 Parameter Value: 80000	Order Shipping Amount	25.0 Parameter Name: param0 Parameter Value: 25.0
			Between 80001 and 10000 Between {0} and {1} Parameter Name: param0 Parameter	Order Shipping Amount	30.0 Parameter Name: param0 Parameter Value: 30.0

			Value: 80001 Parameter Name: param1 Parameter Value: 10000		
	> 5000 > {0} Parameter Name: param0 Parameter Value: 5000	Customer Postal Code	Between 0 and 2000 Between {0} and {1} Parameter Name: param0 Parameter Value: 0 Parameter Name: param1 Parameter Value: 2000	Order Shipping Amount	5.0 Parameter Name: param0 Parameter Value: 5.0
			Between 20001 and 40000 Between {0} and {1} Parameter Name: param0 Parameter Value: 20001 Parameter Name: param1 Parameter Value: 40000	Order Shipping Amount	10.0 Parameter Name: param0 Parameter Value: 10.0

			Between 40001 and 60000 Between {0} and {1} Parameter Name: param0 Parameter Value: 40001 Parameter Name: param1 Parameter Value: 60000	Order Shipping Amount	15.0 Parameter Name: param0 Parameter Value: 15.0
			Between 60001 and 80000 Between {0} and {1} Parameter Name: param0 Parameter Value: 60001 Parameter Name: param1 Parameter Value: 80000	Order Shipping Amount	20.0 Parameter Name: param0 Parameter Value: 20.0

			Between 80001 and 10000 Between {0} and {1} Parameter Name: param0 Parameter Value: 80001 Parameter Name: param1 Parameter Value: 10000	Order Shipping Amount	25.0 Parameter Name: param0 Parameter Value: 25.0

Scheduled Destination:
Start Date: Thu Dec 01 00:00:00 CST 2005 - End Date: Sun Dec 25 00:00:00 CST 2005
Decision Table
Name: calculateShippingDiscountHoliday
Namespace: http://BRSamples/com/ibm/websphere/sample/brules
Init Rule: Rule1
Display Name: Rule1
Description: null
Extended User Presentation: null
User Presentation: null

Order Sub-Total	<= 7500 <= {0} Parameter Name: param0 Parameter Value: 7500	Customer Postal Code	Between 0 and 2000 Between {0} and {1} Parameter Name: param0 Parameter Value: 0 Parameter Name: param1 Parameter	Order Shipping Amount	10.0 Parameter Name: param0 Parameter Value: 10.0

			Value: 2000	
			Between 20001 and 40000 Between {0} and {1} Parameter Name: param0 Parameter Value: 20001 Parameter Name: param1 Parameter Value: 40000	Order Shipping Amount | 15.0 Parameter Name: param0 Parameter Value: 15.0
			Between 40001 and 60000 Between {0} and {1} Parameter Name: param0 Parameter Value: 40001 Parameter Name: param1 Parameter Value: 60000	Order Shipping Amount | 25.0 Parameter Name: param0 Parameter Value: 25.0
			Between 60001 and 80000 Between	Order Shipping Amo | 25.0 Parameter Name: param0

			{0} and {1} Parameter Name: param0 Parameter Value: 60001 Parameter Name: param1 Parameter Value: 80000	unt	Parameter Value: 25.0
			Between 80001 and 10000 Between {0} and {1} Parameter Name: param0 Parameter Value: 80001 Parameter Name: param1 Parameter Value: 10000	Order Ship ping Amo unt	30.0 Parameter Name: param0 Parameter Value: 30.0
	> 7500 > {0} Parameter Name: param0 Parameter Value: 7500	Customer Postal Code	Between 0 and 2000 Between {0} and {1} Parameter Name: param0 Parameter Value: 0	Order Ship ping Amo unt	5.0 Parameter Name: param0 Parameter Value: 5.0

			Parameter Name: param1 Parameter Value: 2000		
			Between 20001 and 40000 Between {0} and {1} Parameter Name: param0 Parameter Value: 20001 Parameter Name: param1 Parameter Value: 40000	Order Shipping Amount	10.0 Parameter Name: param0 Parameter Value: 10.0
			Between 40001 and 60000 Between {0} and {1} Parameter Name: param0 Parameter Value: 40001 Parameter Name: param1 Parameter Value: 60000	Order Shipping Amount	15.0 Parameter Name: param0 Parameter Value: 15.0

			Between 60001 and 80000 Between {0} and {1} Parameter Name: param0 Parameter Value: 60001 Parameter Name: param1 Parameter Value: 80000	Order Shipping Amount	20.0 Parameter Name: param0 Parameter Value: 20.0
			Between 80001 and 10000 Between {0} and {1} Parameter Name: param0 Parameter Value: 80001 Parameter Name: param1 Parameter Value: 10000	Order Shipping Amount	25.0 Parameter Name: param0 Parameter Value: 25.0

3.3 Example 3: Retrieve business rule groups by multiple properties with AND

This example is also similar to example 1, but will only retrieve those business rule groups which have a property named Department and a value of “accounting” and a property named RuleType and a value of “regulatory”. More examples of business rule group queries are available in the appendix.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Property;
import com.ibm.wbiserver.brules.mgmt.PropertyList;
import com.ibm.wbiserver.brules.mgmt.query.AndNode;
import com.ibm.wbiserver.brules.mgmt.query.PropertyQueryNode;
import com.ibm.wbiserver.brules.mgmt.query.QueryNodeFactory;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;

public class Example3
{
static Formatter out = new Formatter();

static public String executeExample3()
{
      try
      {
            out.clear();
```

Queries for business rule groups are composed of query nodes that follow a tree structure. Each query node has a left-hand term and a right-hand term and condition. Each left-hand term and right-hand term can be another query node. For this example the business rule group is retrieved by the combination of two property values.

```
            // Retrieve business rule groups based on two conditions
            // Create PropertyQueryNodes for each one condition
            PropertyQueryNode propertyNode1 = QueryNodeFactory
                        .createPropertyQueryNode("Department",
                        QueryOperator.EQUAL,
                                    "Accounting");
            PropertyQueryNode propertyNode2 = QueryNodeFactory
                        .createPropertyQueryNode("RuleType",
                        QueryOperator.EQUAL,
                                    "regulatory");
            // Combine the two PropertyQueryNodes with an AND node
            AndNode andNode =
            QueryNodeFactory.createAndNode(propertyNode1,
                        propertyNode2);
            // Use andNode in search for business rule groups
            List<BusinessRuleGroup> brgList = BusinessRuleManager
                        .getBRGsByProperties(andNode, 0, 0);

            Iterator<BusinessRuleGroup> iterator = brgList.iterator();

            BusinessRuleGroup brg = null;
            // Iterate through the list of business rule groups
            while (iterator.hasNext())
            {
                  brg = iterator.next();
                  // Output attributes for each business rule group
                  out.printlnBold("Business Rule Group");
                  out.println("Name: " + brg.getName());
                  out.println("Namespace: " +
                  brg.getTargetNameSpace());
                  out.println("Display Name: " + brg.getDisplayName());
                  out.println("Description: " + brg.getDescription());
                  out.println("Presentation Time zone: "
                              + brg.getPresentationTimezone());
                  out.println("Save Date: " + brg.getSaveDate());

                  PropertyList propList = brg.getProperties();

                  Iterator<Property> propIterator =
                  propList.iterator();
                  Property prop = null;
                  // Output property names and values
                  while (propIterator.hasNext())
                  {
                        prop = propIterator.next();
                        out.println("\t Property Name: " +
                        prop.getName());
                        out.println("\t Property Value: " +
                        prop.getValue());
                  }
            }
      } catch (BusinessRuleManagementException e)
      {
            e.printStackTrace();
            out.println(e.getMessage());
      }
      return out.toString();
   }
}
```

Web browser output for example 3.

Executing example3

Business Rule Group
Name: ApprovalValues
Namespace: http://BRSamples/com/ibm/websphere/sample/brules
Display Name: ApprovalValues
Description: null

Presentation Time zone: LOCAL
Save Date: Sun Jan 06 17:56:51 CST 2008
Property Name: IBMSystemVersion
Property Value: 6.1.0
Property Name: Department
Property Value: Accounting
Property Name: RuleType
Property Value: regulatory
Property Name: IBMSystemTargetNameSpace
Property Value: http://BRSamples/com/ibm/websphere/sample/brules
Property Name: IBMSystemName
Property Value: ApprovalValues
Property Name: IBMSystemDisplayName
Property Value: ApprovalValues

3.4 Example 4: Retrieve business rule groups by multiple properties with OR

This example is similar to example 3; however it will only retrieve those business rule groups which have a property named Department and a value of "accounting" or a property named RuleType and a value of "monetary". More examples of business rule group queries are available in the appendix.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Property;
import com.ibm.wbiserver.brules.mgmt.PropertyList;
import com.ibm.wbiserver.brules.mgmt.query.OrNode;
import com.ibm.wbiserver.brules.mgmt.query.PropertyQueryNode;
import com.ibm.wbiserver.brules.mgmt.query.QueryNodeFactory;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;

public class Example4
{
static Formatter out = new Formatter();

static public String executeExample4()
{
      try
      {
            out.clear();
```

Different properties make up the query and return different business rule groups.

```
            // Retrieve business rule groups based on two conditions
            // Create PropertyQueryNodes for each one condition
            PropertyQueryNode propertyNode1 = QueryNodeFactory
                        .createPropertyQueryNode("Department",
                        QueryOperator.EQUAL,
                                    "Accounting");
            PropertyQueryNode propertyNode2 = QueryNodeFactory
                        .createPropertyQueryNode("RuleType",
                        QueryOperator.EQUAL,
                                    "monetary");
            // Combine the two PropertyQueryNodes with an OR node
            OrNode orNode =
            QueryNodeFactory.createOrNode(propertyNode1,
                        propertyNode2);
            // Use orNode in search for business rule groups
            List<BusinessRuleGroup> brgList = BusinessRuleManager
                        .getBRGsByProperties(orNode, 0, 0);
            Iterator<BusinessRuleGroup> iterator = brgList.iterator();

            BusinessRuleGroup brg = null;
            // Iterate through the list of business rule groups
            while (iterator.hasNext())
            {
                  brg = iterator.next();
                  // Output attributes for each business rule group
                  out.printlnBold("Business Rule Group");
                  out.println("Name: " + brg.getName());
                  out.println("Namespace: " +
                  brg.getTargetNameSpace());
                  out.println("Display Name: " + brg.getDisplayName());
                  out.println("Description: " + brg.getDescription());
                  out.println("Presentation Time zone: "
                              + brg.getPresentationTimezone());
                  out.println("Save Date: " + brg.getSaveDate());

                  PropertyList propList = brg.getProperties();

                  Iterator<Property> propIterator =
                  propList.iterator();
                  Property prop = null;
                  // Output property names and values
                  while (propIterator.hasNext())
                  {
                        prop = propIterator.next();
                        out.println("\t Property Name: " +
                        prop.getName());
                        out.println("\t Property Value: " +
                        prop.getValue());
                  }
                  out.println("");
            }
      } catch (BusinessRuleManagementException e)
      {
            e.printStackTrace();
            out.println(e.getMessage());
      }
      return out.toString();
   }
}
```

Web browser output for example 4.

Executing example4

Business Rule Group
Name: ApprovalValues
Namespace: http://BRSamples/com/ibm/websphere/sample/brules
Display Name: ApprovalValues
Description: null
Presentation Time zone: LOCAL

Save Date: Sun Jan 06 17:56:51 CST 2008
Property Name: IBMSystemVersion
Property Value: 6.1.0
Property Name: Department
Property Value: Accounting
Property Name: RuleType
Property Value: regulatory
Property Name: IBMSystemTargetNameSpace
Property Value: http://BRSamples/com/ibm/websphere/sample/brules
Property Name: IBMSystemName
Property Value: ApprovalValues
Property Name: IBMSystemDisplayName
Property Value: ApprovalValues

Business Rule Group
Name: DiscountRules
Namespace: http://BRSamples/com/ibm/websphere/sample/brules
Display Name: DiscountRules
Description: null
Presentation Time zone: LOCAL
Save Date: Sun Jan 06 17:56:51 CST 2008
Property Name: Department
Property Value: Accounting
Property Name: IBMSystemVersion
Property Value: 6.1.0
Property Name: RuleType
Property Value: monetary
Property Name: IBMSystemTargetNameSpace
Property Value: http://BRSamples/com/ibm/websphere/sample/brules
Property Name: IBMSystemName
Property Value: DiscountRules
Property Name: IBMSystemDisplayName
Property Value: DiscountRules

3.5 Example 5: Retrieve business rule groups with a complex query

This example is a combination of examples 3 and 4 and it is meant to show how more complex queries can be created. In this example a search is performed with a query that combines 2 query conditions. The first query condition is to retrieve those business rule groups which have a property named Department and a value of “General” or a property named MissingProperty and a value of “somevalue”. This query condition is then combined with an AND to a condition where the property is named RuleType and a value of “messages”. More examples of business rule group queries are available in the appendix.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Property;
import com.ibm.wbiserver.brules.mgmt.PropertyList;
import com.ibm.wbiserver.brules.mgmt.query.AndNode;
import com.ibm.wbiserver.brules.mgmt.query.OrNode;
import com.ibm.wbiserver.brules.mgmt.query.PropertyQueryNode;
import com.ibm.wbiserver.brules.mgmt.query.QueryNodeFactory;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;

public class Example5
{
static Formatter out = new Formatter();

static public String executeExample5()
{
      try
      {
            out.clear();

            // Retrieve business rule groups based on three conditions
            where
            // two of the conditions are combined in an OR node
            // Create PropertyQueryNodes for each condition for the OR
            node
            PropertyQueryNode propertyNode1 = QueryNodeFactory
                        .createPropertyQueryNode("Department",
                        QueryOperator.EQUAL,
                                    "General");
            PropertyQueryNode propertyNode2 = QueryNodeFactory
                        .createPropertyQueryNode("MissingProperty",
                                    QueryOperator.EQUAL, "SomeValue");
            // Combine the two PropertyQueryNodes with an OR node
            OrNode orNode =
            QueryNodeFactory.createOrNode(propertyNode1,
                        propertyNode2);
            // Create the third PropertyQueryNode
            PropertyQueryNode propertyNode3 = QueryNodeFactory
                        .createPropertyQueryNode("RuleType",
                        QueryOperator.EQUAL,
                                    "messages");
```

The left-hand side of the condition is combined to the right-hand with an AND node. The `AndNode` is the root of the query tree.

```
            // Combine OR node with third PropertyQueryNode with
            AndNode
            AndNode andNode =
            QueryNodeFactory.createAndNode(propertyNode3,
                        orNode);

            List<BusinessRuleGroup> brgList = BusinessRuleManager
                        .getBRGsByProperties(andNode, 0, 0);

            Iterator<BusinessRuleGroup> iterator = brgList.iterator();

            BusinessRuleGroup brg = null;
            // Iterate through the list of business rule groups
            while (iterator.hasNext())
            {
                  brg = iterator.next();
                  // Output attributes for each business rule group
                  out.printlnBold("Business Rule Group");
                  out.println("Name: " + brg.getName());
                  out.println("Namespace: " +
                  brg.getTargetNameSpace());
                  out.println("Display Name: " + brg.getDisplayName());
                  out.println("Description: " + brg.getDescription());
                  out.println("Presentation Time zone: "
                              + brg.getPresentationTimezone());
                  out.println("Save Date: " + brg.getSaveDate());

                  PropertyList propList = brg.getProperties();

                  Iterator<Property> propIterator =
                  propList.iterator();
                  Property prop = null;
                  // Output property names and values
                  while (propIterator.hasNext())
                  {
                        prop = propIterator.next();
                        out.println("\t Property Name: " +
                        prop.getName());
                        out.println("\t Property Value: " +
                        prop.getValue());
                  }
            }
      } catch (BusinessRuleManagementException e)
      {
            e.printStackTrace();
            out.println(e.getMessage());
        }
        return out.toString();
    }
}
```

Web browser output for example 5.

```
Executing example5

Business Rule Group
Name: ConfigurationValues
Namespace: http://BRSamples/com/ibm/websphere/sample/brules
Display Name: ConfigurationValues
Description: null
Presentation Time zone: LOCAL
Save Date: Sun Jan 06 17:56:51 CST 2008
Property Name: IBMSystemVersion
Property Value: 6.1.0
Property Name: Department
Property Value: General
Property Name: RuleType
Property Value: messages
Property Name: IBMSystemTargetNameSpace
Property Value: http://BRSamples/com/ibm/websphere/sample/brules
Property Name: IBMSystemName
Property Value: ConfigurationValues
Property Name: IBMSystemDisplayName
Property Value: ConfigurationValues
```

3.6 Example 6: Update a business rule group property and publish

In this example, a property in a business rule group is updated and then the business rule group is published.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.ArrayList;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.UserDefinedProperty;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;

public class Example6
{
static Formatter out = new Formatter();

static public String executeExample6()
{
      try
      {
            out.clear();
            out.printlnBold("Business Rule Group before publish:");
            // Retrieve business rule groups by a single property value
            List<BusinessRuleGroup> brgList = BusinessRuleManager
                        .getBRGsBySingleProperty("Department",
                        QueryOperator.EQUAL,
                                    "General", 0, 0);

            if (brgList.size() > 0)
            {
                  // Get the first business rule group from the list
                  BusinessRuleGroup brg = brgList.get(0);
                  // Retrieve the property from the business rule group
                  UserDefinedProperty userDefinedProperty =
                  (UserDefinedProperty) brg
                              .getProperty("Department");

                  out.println("Business Rule Group: " + brg.getName());
                  out.println("Department Property value: "
                  + brg.getProperty("Department").getValue());
```

The `getProperty` method returns a property by reference and changes made to the property are directly made to the business rule group.

```
                  // Modify the property value in the brg
                  // This updates the property value directly in the
                  brg object
                  userDefinedProperty.setValue("GeneralConfig");
                  // Use the original list or create a new list
                  // of business rule groups
                  List<BusinessRuleGroup> publishList = new
                  ArrayList<BusinessRuleGroup>();
                  // Add the changed business rule group to the list
                  publishList.add(brg);
```

The `BusinessRuleManager` class is used to publish the changes made to a business rule group. To publish the change, a list is passed to the `BusinessRuleManager` publish method even if there is only one item is being published.

```
                  // Publish the list with the updated business rule
                  group
                  BusinessRuleManager.publish(publishList, true);

                  out.println("");

                  // Retrieve the business rule group again to verify
                  the
                  // changes were published
                  out.printlnBold("Business Rule Group after
                  publish:");
                  brgList = BusinessRuleManager
                  .getBRGsBySingleProperty("Department",
                  QueryOperator.EQUAL,
                              "GeneralConfig", 0, 0);

                  brg = brgList.get(0);

                  out.println("Business Rule Group: " + brg.getName());
                  // Display the property value to show the change
                  out.println("Department Property value: "
                  + brg.getProperty("Department").getValue());
            }
      } catch (BusinessRuleManagementException e)
      {
            e.printStackTrace();
            out.println(e.getMessage());
      }
      return out.toString();
   }
}
```

Web browser output for example 6.

Executing example6

Business Rule Group before publish:
Business Rule Group: ConfigurationValues
Department Property value: General

Business Rule Group after publish:
Business Rule Group: ConfigurationValues

Department Property value: GeneralConfig

3.7 Example 7: Update properties in multiple business rule groups and publish

In this example, properties in multiple business rule groups are updated before publish.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.UserDefinedProperty;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;

public class Example7
{
static Formatter out = new Formatter();

static public String executeExample7()
{
      try
      {
            out.clear();
            out.printlnBold("Business Rule Group before publish:");
            // Retrieve business rule groups by a single property value
            List<BusinessRuleGroup> brgList = BusinessRuleManager
                        .getBRGsBySingleProperty("Department",
                        QueryOperator.EQUAL,
                                    "Accounting", 0, 0);

            Iterator<BusinessRuleGroup> iterator = brgList.iterator();

            BusinessRuleGroup brg = null;

            // Use the original list or create a new list
            // of business rule groups
            List<BusinessRuleGroup> publishList = new
            ArrayList<BusinessRuleGroup>();

            // Iterate through all of the business rule groups and
            // modify the property
            while (iterator.hasNext())
            {
                  // Retrieve the property from the business rule group
                  brg = iterator.next();

                  out.println("Business Rule Group: " + brg.getName());

                  // Retrieve the property from the business rule group
                  UserDefinedProperty prop = (UserDefinedProperty) brg
                              .getProperty("Department");
                  out.println("Department Property value: "
                              +
                              brg.getProperty("Department").getValue())
                              ;

                  // Modify the property value in the brg
                  // This updates the property value directly in the
                  brg object
                  prop.setValue("Finance");
```

Each changed business rule group is added to the list.

```
                  // Add the changed business rule group to the list
                  publishList.add(brg);
            }
                  // Publish the list with the updated business rule
                  group
                  BusinessRuleManager.publish(publishList, true);


                  out.println("");

                  // Retrieve the business rule groups again to verify
                  the
                  // changes were published
                  out.printlnBold("Business Rule Group after
                  publish:");

                  brgList = BusinessRuleManager
                              .getBRGsBySingleProperty("Department",
                              QueryOperator.EQUAL,
                                          "Finance", 0, 0);

                  iterator = brgList.iterator();

                  while (iterator.hasNext())
                  {

                        brg = iterator.next();
                        out.println("Business Rule Group: " +
                        brg.getName());
                        out.println("Department Property value: "
                                    +
                                    brg.getProperty("Department").getVa
                                    lue());
                  }


      } catch (BusinessRuleManagementException e)
      {
            e.printStackTrace();
            out.println(e.getMessage());
      }
      return out.toString();
```

```
}
}
```

Web browser output for example 7,

```
Executing example7

Business Rule Group before publish:
Business Rule Group: ApprovalValues
Department Property value: Accounting
Business Rule Group: DiscountRules
Department Property value: Accounting

Business Rule Group after publish:
Business Rule Group: ApprovalValues
Department Property value: Finance
Business Rule Group: DiscountRules
Department Property value: Finance
```

3.8 Example 8: Change the default business rule for a business rule group

In this example, the default business rule is changed with another business rule that is part of the available targets list for a specific operation.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;

public class Example8
{
static Formatter out = new Formatter();

static public String executeExample8()
{
      try
      {
            out.clear();

            // Retrieve a business rule group by target namespace and
            name
            List<BusinessRuleGroup> brgList = BusinessRuleManager
                        .getBRGsByTNSAndName(
                                    "http://BRSamples/com/ibm/websphere
                                    /sample/brules",
                                    QueryOperator.EQUAL,
                                    "DiscountRules",
                                    QueryOperator.EQUAL, 0, 0);

            if (brgList.size() > 0)
            {
                  out.printlnBold("Business Rule Group before
                  publish:");
                  // Get the first business rule group from the list
                  // This should be the only business rule group in the
                  list as
                  // the combination of target namespace and name are
                  unique
                  BusinessRuleGroup brg = brgList.get(0);

                  out.print("Business Rule Group: ");
                  out.println(brg.getName());
```

```
// Get the operation of the business rule group that
// will have its default business rule updated
Operation op =
brg.getOperation("calculateShippingDiscount");
```

The default business rule is retrieved before it is updated with another rule that is part of the available target list for the operation. Rule sets and decision tables are specific to operations and only those business rule artifacts that are for an operation can be set to be default or be scheduled for another time on the operation.

```
// Retrieve the default business rule for the
operation
BusinessRule defaultRule =
op.getDefaultBusinessRule();
out.print("Default Rule: ");
out.println(defaultRule.getName());

// Get the list of available business rules for this
operation
List<BusinessRule> ruleList =
op.getAvailableTargets();

Iterator<BusinessRule> iterator =
ruleList.iterator();
BusinessRule rule = null;

// Find a business rule that is different from the
current
// default
// business rule
while (iterator.hasNext())
{
      rule = iterator.next();
      if
      (!defaultRule.getName().equals(rule.getName()))
      {
```

The default business rule is set on the operation object. Setting the default business rule to null will remove any default business rule from the operation, however it is recommended that every operation have a default business rule specified.

```
            // Set the default business rule to be a
            different
            // business rule
            // This change is to the operation object
            // directly
            op.setDefaultBusinessRule(rule);
            break;
      }
}

// Use the original list or create a new list
// of business rule groups
```

```
                List<BusinessRuleGroup> publishList = new
                ArrayList<BusinessRuleGroup>();
                // Add the changed business rule group to the list
                publishList.add(brg);
                // Publish the list with the updated business rule
                group
                BusinessRuleManager.publish(publishList, true);

                out.println("");

                // Retrieve the business rule groups again to verify
                the
                // changes were published

                out.printlnBold("Business Rule Group after
                publish:");
                brgList = BusinessRuleManager
                .getBRGsByTNSAndName(
                        "http://BRSamples/com/ibm/websphere/sampl
                        e/brules",
                        QueryOperator.EQUAL, "DiscountRules",
                        QueryOperator.EQUAL, 0, 0);

                brg = brgList.get(0);

                out.println("Business Rule Group: " + brg.getName());
                op = brg.getOperation("calculateShippingDiscount");

                // Retrieve the default business rule for the
                operation
                defaultRule = op.getDefaultBusinessRule();
                out.print("Default Rule: ");
                out.println(defaultRule.getName());
            }
        } catch (BusinessRuleManagementException e)
        {
            e.printStackTrace();
            out.println(e.getMessage());
        }
        return out.toString();
    }
}
```

Web browser for example 8.

```
Executing example8

Business Rule Group before publish:
Business Rule Group: DiscountRules
Default Rule: calculateShippingDiscount

Business Rule Group after publish:
Business Rule Group: DiscountRules
Default Rule: calculateShippingDiscountHoliday
```

3.9 Example 9: Schedule another rule for an operation in a business rule group

In this example, a business rule is scheduled to be active for 1 hour from the time of publish for a specific operation.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.ArrayList;
import java.util.Date;
import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.OperationSelectionRecordList;
import com.ibm.wbiserver.brules.mgmt.OperationSelectionRecord;
import com.ibm.wbiserver.brules.mgmt.problem.Problem;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;

public class Example9 {
static Formatter out = new Formatter();

static public String executeExample9()
{
      try
      {
            out.clear();

            // Retrieve a business rule group by target namespace and
            name
            List<BusinessRuleGroup> brgList = BusinessRuleManager
                        .getBRGsByTNSAndName(
                                    "http://BRSamples/com/ibm/websphere
                                    /sample/brules",
                                    QueryOperator.EQUAL,
                                    "DiscountRules",
                                    QueryOperator.EQUAL, 0, 0);

            if (brgList.size() > 0)
            {
                  out.println("");
                  out.printlnBold("Business Rule Group before
                  publish:");
                  // Get the first business rule group from the list
                  // This should be the only business rule group in the
                  list as
                  // the combination of target namespace and name are
                  unique
```

```
BusinessRuleGroup brg = brgList.get(0);

// Get the operation of the business rule group that
// will have a new business rule scheduled
Operation op =
brg.getOperation("calculateShippingDiscount");

printOperationSelectionRecord(op);

// Get the list of available business rules for this
operation
List<BusinessRule> ruleList =
op.getAvailableTargets();

// Get the first rule in the list as this will be
scheduled
// for the operation
BusinessRule rule = ruleList.get(0);

// Get the list of scheduled business rules
OperationSelectionRecordList opList = op
            .getOperationSelectionRecordList();

// Create an end date in the future for the business
rule
Date future = new Date();
long futureTime = future.getTime() + 3600000;
```

For a new scheduled rule, a start date and end date can be specified along with the rule. If the start date is set to null, this indicates that the rule will be active immediate upon publish. If an end date is set to null, the rule will not have an end date. An overlap of schedules is not allowed and can be checked by calling the validate method on the operation.

```
// Create the new scheduled business rule with the
current
// date which means this rule will become active
immediately
// upon
// publish and the future date. Also specify the rule
to be
// scheduled
OperationSelectionRecord newRecord = opList
            .newOperationSelectionRecord(new Date(),
            new Date(
                        futureTime), rule);
// Add the new scheduled business rule to the list of
// scheduled rule
opList.addOperationSelectionRecord(newRecord);
```

Validate operation to insure that an overlap does not exist.

```
// Validate the list to insure there isn't an overlap
List<Problem> problems = op.validate();
```

```
                  if (problems.size() == 0)
                  {

                        // Use the original list or create a new list
                        // of business rule groups
                        List<BusinessRuleGroup> publishList = new
                        ArrayList<BusinessRuleGroup>();
                        // Add the changed business rule group to the
                        list
                        publishList.add(brg);
                        // Publish the list with the updated business
                        rule group
                        BusinessRuleManager.publish(publishList, true);
                        out.println("");

                        // Retrieve the business rule groups again to
                        verify the
                        // changes were published
                        out.printlnBold("Business Rule Group after
                        publish:");

                        brgList =
                        BusinessRuleManager.getBRGsByTNSAndName(
                                    "http://BRSamples/com/ibm/websphere
                                    /sample/brules",
                                    QueryOperator.EQUAL,
                                    "DiscountRules",
                                    QueryOperator.EQUAL, 0, 0);
                        brg = brgList.get(0);

                        op =
                        brg.getOperation("calculateShippingDiscount");

                        printOperationSelectionRecord(op);
                  }
                  // else handle the validation error
            }

      } catch (BusinessRuleManagementException e)
      {
            e.printStackTrace();
            out.println(e.getMessage());
      }
      return out.toString();
}

/*
Method to print the operation selection record for an operation. The
start date and end date are printed as well as the name of the rule
artifact for the scheduled time.
*/
private static void printOperationSelectionRecord(Operation op)
{
      OperationSelectionRecordList opSelectionRecordList = op
                  .getOperationSelectionRecordList();
      Iterator<OperationSelectionRecord> opSelRecordIterator =
      opSelectionRecordList
                  .iterator();

      OperationSelectionRecord record = null;

      while (opSelRecordIterator.hasNext())
      {
            out.printlnBold("Scheduled Destination:");
            record = opSelRecordIterator.next();

            out.println("Start Date: " + record.getStartDate()
                        + " - End Date: " + record.getEndDate());
            BusinessRule ruleArtifact = record.getBusinessRuleTarget();
            out.println("Rule: " + ruleArtifact.getName());
      }
   }
}
```

Web browser output for example 9.

```
Executing example9


Business Rule Group before publish:
Scheduled Destination:
Start Date: Thu Dec 01 00:00:00 CST 2005 - End Date: Sun Dec 25 00:00:00 CST 2005
Rule: calculateShippingDiscountHoliday

Business Rule Group after publish:
Scheduled Destination:
Start Date: Thu Dec 01 00:00:00 CST 2005 - End Date: Sun Dec 25 00:00:00 CST 2005
Rule: calculateShippingDiscountHoliday
Scheduled Destination:
Start Date: Mon Jan 07 21:08:31 CST 2008 - End Date: Mon Jan 07 22:08:31 CST 2008
Rule: calculateShippingDiscount
```

3.10 Example 10: Modify a parameter value in a template in a rule set

In this example a rule instance defined with a template is modified by changing a parameter value and then publish.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.ParameterValue;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSet;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetRule;
import
com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetTemplateInstanceRule;
import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleBlock;

public class Example10
{
static Formatter out = new Formatter();

static public String executeExample10()
{
      try
      {
            out.clear();

            // Retrieve a business rule group by target namespace and
            name
            List<BusinessRuleGroup> brgList = BusinessRuleManager
                        .getBRGsByTNSAndName(
                                    "http://BRSamples/com/ibm/websphere
                                    /sample/brules",
                                    QueryOperator.EQUAL,
                                    "ApprovalValues",
                                    QueryOperator.EQUAL, 0, 0);

            if (brgList.size() > 0)
            {
                  // Get the first business rule group from the list
                  // This should be the only business rule group in the
                  list as
                  // the combination of target namespace and name are
                  unique
                  BusinessRuleGroup brg = brgList.get(0);
```

```
// Get the operation of the business rule group that
// has the business rule that will be modified as
// the business rules are associated with a specific
// operation
Operation op = brg.getOperation("getApprover");

// Get the business rule on the operation that will
be modified
List<BusinessRule> ruleList =
op.getBusinessRulesByName(
            "getApprover", QueryOperator.EQUAL, 0,
            0);

if (ruleList.size() > 0)
{
      out.println("");
      out.printlnBold("Rule set before publish:");
      // Get the rule to be modified. Rules are
      unique by
      // target namespace and name, but for this
      example
      // there is only one business rule named
      "getApprover"
      RuleSet ruleSet = (RuleSet) ruleList.get(0);
      out.print(RuleArtifactUtility.printRuleSet(rule
      Set));
```

All of the rules in a rule set are in a rule block. Only one rule block is supported and the `getFirstRuleBlock` method should be used to retrieve the rule block.

```
      // A rule set has all of the rules defined in a
      rule block
      RuleBlock ruleBlock =
      ruleSet.getFirstRuleBlock();

      Iterator<RuleSetRule> ruleIterator =
      ruleBlock.iterator();

      // Iterate through the rules in the rule block
      to find the
      // rule instance called "LargeOrderApprover"
      while (ruleIterator.hasNext())
      {
            RuleSetRule rule = ruleIterator.next();
```

If a rule is not defined with a rule template, it only has a Web presentation that can be retrieved. No updates can be made to a rule that is not defined with a template. It is best to check if a rule has been defined with a template if the name of the rule is unknown.

```
            // The rule must have been defined with a
            template
            // in order for it to be changed. Check
            if the current
            // rule is even based on a template.
```

```
                        if (rule instanceof
                        RuleSetTemplateInstanceRule)
                        {
```

Use the TemplateInstance object to create the rule.

```
                             // Get the rule template instance
                             RuleSetTemplateInstanceRule
                             templateInstance =
                             (RuleSetTemplateInstanceRule) rule;

                             // Check for the rule instance
                             which matches
                             // the rule to modify
                             if
                             (templateInstance.getName().equals(
                                         "LargeOrderApprover"))
                             {
```

For the template instance, only parameter values can be modified. The parameters are modified by retrieving the `ParameterValue` and setting it to the appropriate value. Because the `ParameterValue` is passed by reference, the update is made directly on the rule, rule set, and business rule group.

```
                                   // Get the parameter from the
                                   rule instance
                                   ParameterValue parameter =
                                   templateInstance
                                         .getParameterValue("par
                                         am2");

                                   // Modify the value of the
                                   parameter
                                   parameter.setValue("superviso
                                   r");
                                   break;
                             }
                        }
                  }

                  // Use the original list or create a new list
                  // of business rule groups
                  List<BusinessRuleGroup> publishList = new
                  ArrayList<BusinessRuleGroup>();

                  // Add the changed business rule group to the list
                  publishList.add(brg);

                  // Publish the list with the updated business rule
                  group
                  BusinessRuleManager.publish(publishList, true);

                  out.println("");
                  // Retrieve the business rule groups again to verify
                  the
                  // changes were published
                  out.printlnBold("Rule set after publish:");

                  brgList = BusinessRuleManager
                  .getBRGsByTNSAndName(
                              "http://BRSamples/com/ibm/websphere/sampl
                              e/brules",
                              QueryOperator.EQUAL, "ApprovalValues",
                              QueryOperator.EQUAL, 0, 0);

                  brg = brgList.get(0);
                  op = brg.getOperation("getApprover");
                  ruleList = op.getBusinessRulesByName(
                              "getApprover", QueryOperator.EQUAL, 0,
                              0);

                  ruleSet = (RuleSet) ruleList.get(0);
                  out.print(RuleArtifactUtility.printRuleSet(ruleSet));
                  }
            }
      } catch (BusinessRuleManagementException e)
      {
            e.printStackTrace();
            out.println(e.getMessage());
      }
      return out.toString();
   }
}
```

Web browser output for example 10.

Executing example10

Rule set before publish:
Rule Set
Name: getApprover
Namespace: http://BRSamples/com/ibm/websphere/sample/brules
Rule: LargeOrderApprover
Display Name: LargeOrderApprover
Description: null
Expanded User Presentation: If the number of items order is above 10 and the order is above $5000, then it requires the approval of manager
User Presentation: If the number of items order is above {0} and the order is above ${1}, then it requires the approval of {2}
Parameter Name: param0
Parameter Value: 10
Parameter Name: param1
Parameter Value: 5000
Parameter Name: param2

Parameter Value: manager
Rule: DefaultApprover
Display Name: DefaultApprover
Description: null
Expanded User Presentation: approver = peer
User Presentation: approver = {0}
Parameter Name: param0
Parameter Value: peer

Rule set after publish:
Rule Set
Name: getApprover
Namespace: http://BRSamples/com/ibm/websphere/sample/brules
Rule: LargeOrderApprover
Display Name: LargeOrderApprover
Description: null
Expanded User Presentation: If the number of items order is above 10 and the order is above $5000, then it requires the approval of supervisor
User Presentation: If the number of items order is above {0} and the order is above ${1}, then it requires the approval of {2}
Parameter Name: param0
Parameter Value: 10
Parameter Name: param1
Parameter Value: 5000
Parameter Name: param2
Parameter Value: supervisor
Rule: DefaultApprover
Display Name: DefaultApprover
Description: null
Expanded User Presentation: approver = peer
User Presentation: approver = {0}
Parameter Name: param0
Parameter Value: peer

3.11 Example 11: Add a new rule from a template to a rule set

In this example, a new rule is added from a template to a rule set. Before the new rule instance is created, parameters for the new rule instance are created.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.Parameter;
import com.ibm.wbiserver.brules.mgmt.ParameterValue;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleBlock;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSet;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetRuleTemplate;
import
com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetTemplateInstanceRule;

public class Example11
{
static Formatter out = new Formatter();

static public String executeExample11()
{
      try
      {
            out.clear();

            // Retrieve a business rule group by target namespace and
            name
            List<BusinessRuleGroup> brgList = BusinessRuleManager
                        .getBRGsByTNSAndName(
                                    "http://BRSamples/com/ibm/websphere
                                    /sample/brules",
                                    QueryOperator.EQUAL,
                                    "ApprovalValues",
                                    QueryOperator.EQUAL, 0, 0);

            if (brgList.size() > 0)
            {
                  // Get the first business rule group from the list
                  // This should be the only business rule group in the
                  list as
                  // the combination of target namespace and name are
                  unique
                  BusinessRuleGroup brg = brgList.get(0);
```

```
// Get the operation of the business rule group that
// has the business rule that will be modified as
// the business rules are associated with a specific
// operation
Operation op = brg.getOperation("getApprover");

// Get the business rule on the operation that will
be modified
List<BusinessRule> ruleList =
op.getBusinessRulesByName(
            "getApprover", QueryOperator.EQUAL, 0,
            0);

if (ruleList.size() > 0)
{
      out.println("");
      out.printlnBold("Rule set before publish:");
      // Get the rule to be modified. Rules are
      unique by
      // target namespace and name, but for this
      example
      // there is only one business rule named
      "getApprover"
      RuleSet ruleSet = (RuleSet) ruleList.get(0);
      out.print(RuleArtifactUtility.printRuleSet(rule
      Set));
```

In order to add a new rule to the rule set, the appropriate template must be located in the rule set and an instance created from the template. The template can be located by name.

```
      // Get the list of rule templates
      List<RuleSetRuleTemplate> ruleTemplates =
      ruleSet
                  .getRuleTemplates();

      Iterator<RuleSetRuleTemplate> templateIterator
      = ruleTemplates
                  .iterator();

      while (templateIterator.hasNext())
      {
            RuleSetRuleTemplate template =
            templateIterator.next();

            // Locate the template to use to create a
            new rule
            if
            (template.getName().equals("Template_Larg
            eOrder"))
            {
```

For a template instance, a list of parameters must be created.

```
// Create a list for the parameters
for this template
// rule instance
List<ParameterValue> paramList =
new ArrayList<ParameterValue>();

// From the template definition,
get a specific parameter
// and set the value
Parameter param =
template.getParameter("param0");
ParameterValue paramValue = param
            .createParameterValue("
            20");

// Add parameter to the list
paramList.add(paramValue);

// Get the next parameter and set
the value
param =
template.getParameter("param1");
paramValue =
param.createParameterValue("7500");

// Add parameter to the list
paramList.add(paramValue);


// Get the next parameter and set
the value
param =
template.getParameter("param2");
paramValue = param
            .createParameterValue("
            2nd-line manager");

// Add parameter to the list
paramList.add(paramValue);
```

With the parameters created, the template instance can be created.

```
// Create the template rule
instance with the parameter
// list
RuleSetTemplateInstanceRule
templateInstance = template
            .createRuleFromTemplate
            ("ExtraLargeOrder",
            paramList);

// Get the ruleblock for the rule
set
RuleBlock ruleBlock =
ruleSet.getFirstRuleBlock();
```

Once the template instance is created, it can be added to the ruleblock. Once it is added to the rule block it can be ordered among other template rule instances

```
                        // Add the template rule to the
                        ruleblock
                        ruleBlock.addRule(templateInstance)
                        ;

                        break;
                  }
            }

            // Use the original list or create a new list
            // of business rule groups
            List<BusinessRuleGroup> publishList = new
            ArrayList<BusinessRuleGroup>();

            // Add the changed business rule group to the
            list
            publishList.add(brg);

            // Publish the list with the updated business
            rule group
            BusinessRuleManager.publish(publishList, true);

            out.println("");

            // Retrieve the business rule groups again to
            verify the
            // changes were published
            out.printlnBold("Rule set after publish:");

            brgList = BusinessRuleManager
            .getBRGsByTNSAndName(
                        "http://BRSamples/com/ibm/websphere
                        /sample/brules",
                        QueryOperator.EQUAL,
                        "ApprovalValues",
                        QueryOperator.EQUAL, 0, 0);

            brg = brgList.get(0);
            op = brg.getOperation("getApprover");
            ruleList = op.getBusinessRulesByName(
                        "getApprover", QueryOperator.EQUAL,
                        0, 0);

            ruleSet = (RuleSet) ruleList.get(0);
            out.print(RuleArtifactUtility.printRuleSet(rule
            Set));
      }
   }
} catch (BusinessRuleManagementException e)
{
      e.printStackTrace();
      out.println(e.getMessage());
}
```

```
      return out.toString();
}
}
```

Web browser output for example 11.

Executing example11

Rule set before publish:
Rule Set
Name: getApprover
Namespace: http://BRSamples/com/ibm/websphere/sample/brules
Rule: LargeOrderApprover
Display Name: LargeOrderApprover
Description: null
Expanded User Presentation: If the number of items order is above 10 and the order is above $5000, then it requires the approval of supervisor
User Presentation: If the number of items order is above {0} and the order is above ${1}, then it requires the approval of {2}
Parameter Name: param0
Parameter Value: 10
Parameter Name: param1
Parameter Value: 5000
Parameter Name: param2
Parameter Value: supervisor
Rule: DefaultApprover
Display Name: DefaultApprover
Description: null
Expanded User Presentation: approver = peer
User Presentation: approver = {0}
Parameter Name: param0
Parameter Value: peer

Rule set after publish:
Rule Set
Name: getApprover
Namespace: http://BRSamples/com/ibm/websphere/sample/brules
Rule: LargeOrderApprover
Display Name: LargeOrderApprover
Description: null
Expanded User Presentation: If the number of items order is above 10 and the order is above $5000, then it requires the approval of supervisor
User Presentation: If the number of items order is above {0} and the order is above ${1}, then it requires the approval of {2}
Parameter Name: param0

Parameter Value: 10
Parameter Name: param1
Parameter Value: 5000
Parameter Name: param2
Parameter Value: supervisor
Rule: DefaultApprover
Display Name: DefaultApprover
Description: null
Expanded User Presentation: approver = peer
User Presentation: approver = {0}
Parameter Name: param0
Parameter Value: peer
Rule: ExtraLargeOrder
Display Name:
Description: null
Expanded User Presentation: If the number of items order is above 20 and the order is above $7500, then it requires the approval of 2nd-line manager
User Presentation: If the number of items order is above {0} and the order is above ${1}, then it requires the approval of {2}
Parameter Name: param0
Parameter Value: 20
Parameter Name: param1
Parameter Value: 7500
Parameter Name: param2
Parameter Value: 2nd-line manager

3.12 Example 12: Modify a template in a decision table by changing a parameter value and then publish

In this example, a condition and action, both defined with templates, are modified in a decision table by changing the parameter values before it is published.

The easiest way to modify conditions and actions in a decision table is to use unique names for the templates at each condition level and for each action. The unique names can be searched for and then changes can be made to template instances defined with the template. When changes are made to a template instance of a particular template, all of the condition values defined with that template at that level will be updated. For action expressions, each instance is unique and a change to one does not change others.

For this example, there are a number of additional methods that were created to simplify the locating of a specific case edge for update, finding the specific parameter value, and finding the action expression defined with a specific template.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Vector;

import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.ParameterValue;
import com.ibm.wbiserver.brules.mgmt.Template;
import com.ibm.wbiserver.brules.mgmt.dtable.ActionNode;
import com.ibm.wbiserver.brules.mgmt.dtable.CaseEdge;
import com.ibm.wbiserver.brules.mgmt.dtable.ConditionNode;
import com.ibm.wbiserver.brules.mgmt.dtable.DecisionTable;
import com.ibm.wbiserver.brules.mgmt.dtable.TemplateInstanceExpression;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeAction;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeBlock;
import
com.ibm.wbiserver.brules.mgmt.dtable.TreeConditionValueDefinition;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeNode;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;

public class Example12 {
static Formatter out = new Formatter();

static public String executeExample12()
{
      try
      {
            out.clear();
            // Retrieve a business rule group by target namespace and
            name
            List<BusinessRuleGroup> brgList = BusinessRuleManager
                        .getBRGsByTNSAndName(
                                    "http://BRSamples/com/ibm/websphere
                                    /sample/brules",
                                    QueryOperator.EQUAL,
                                    "ConfigurationValues",
                                    QueryOperator.EQUAL, 0, 0);

            if (brgList.size() > 0)
            {
                  // Get the first business rule group from the list
                  // This should be the only business rule group in the
                  list as
                  // the combination of target namespace and name are
                  unique
                  BusinessRuleGroup brg = brgList.get(0);

                  // Get the operation of the business rule group that
                  // has the business rule that will be modified as
                  // the business rules are associated with a specific
                  // operation
                  Operation op = brg.getOperation("getMessages");

                  // Get all business rules available for this
                  operation
                  List<BusinessRule> ruleList =
                  op.getAvailableTargets();

                  // For this operation there is only 1 business rule
                  and
                  // it is the business that we want to update
                  DecisionTable decisionTable = (DecisionTable)
                  ruleList.get(0);
                  out.println("");
                  out.printlnBold("Decesion table before publish:");
                  out
                              .print(RuleArtifactUtility
                                          .printDecisionTable(decisionT
                                          able));
```

The init rule and condition and actions are contained in a tree block. With the tree block, the root node can be retrieved.

```
                  // Get the tree block that contains all of the
                  conditions
                  // and actions for the decision table
                  TreeBlock treeBlock = decisionTable.getTreeBlock();

                  // From the tree block, get the tree node which is
                  the
                  // starting point for navigating through the decision
                  table
                  TreeNode treeNode = treeBlock.getRootNode();
```

The condition to be updated was defined with a template by the name of “Condition Value Template 2.1”. The method `getCaseEdge` will recursively search from the TreeNode down to the appropriate case edge to find the case edge where the template is defined. The method expects that the level at which the template is defined is known as well as the current level. This method can be used to find the case edge with a template by a specific name in case the same name is used at multiple case edges.

```
                  // Find the case edge at level 1 below the root with
                  // specific template with a parameter value that has
                  // a specific name. Since we are starting at the top,
                  // the current depth is 0
                  CaseEdge caseEdge = getCaseEdge(treeNode, "param0",
                              "Condition Value Template 2.1", 1, 0);
```

With the case edge found, the `ConditionValueTemplateInstance` for the condition can be retrieved.

```
                  if (caseEdge != null)
                  {
                        // Case edge was found. Get the value
                        definition of the
                        // case edge
                        TreeConditionValueDefinition condition =
                        caseEdge
                                    .getValueDefinition();

                        // Get the condition expression defined with a
                        template
                        TemplateInstanceExpression conditionExpression
                        = condition
                                    .getConditionValueTemplateInstance(
                                    );
```

With the `ConditionValueTemplateInstance` , the appropriate parameter value can be retrieve and then updated with the `getParameterValue` method.

```
                        // Get the template for the expression
                        Template conditionTemplate =
                        conditionExpression
                                    .getTemplate();

                        // Check that template is correct as it is
                        possible to have
                        // multiple templates for a condition value,
                        but only one
                        // applied
                        if (conditionTemplate.getName().equals(
                                    "Condition Value Template 2.1"))
                        {
                              // Get the parameter value
                              ParameterValue parameterValue =
                              getParameterValue(
                                          "param0",
                                          conditionExpression);

                              // Set the new parameter value
                              parameterValue.setValue("info");
                        }
```

The different action expressions defined with templates that need to be updated can then be retrieved. The `getActionExpressions` method will return all actions that are defined with the template by name Action Value Template 1.

```
      ConditionNode conditionNode = (ConditionNode)
      treeNode;

      // Get the case edges tree node
      List<CaseEdge> caseEdges =
      conditionNode.getCaseEdges();

      // Create a list to hold all of the action
      expressions that
      // also need to be updated. Because every
      action is
      // independent of other action even though the
      template is
      // shared, all must be updated.
      List<TemplateInstanceExpression> expressions =
      new Vector<TemplateInstanceExpression>();

      // Retrieve all of the expressions
      for (CaseEdge edge : caseEdges)
      {
            getActionExpressions("Action Value
            Template 1", edge,
                        expressions);
      }
```

With the list of action expressions, each item can be updated. For action expressions defined with templates the correct parameter value can be updated.

```
      // Update the correct parameter in each
      expression
      for (TemplateInstanceExpression expression :
      expressions)
      {

            for (ParameterValue parameterValue :
            expression
                        .getParameterValues())
            {
                  // Check for correct parameter
                  although there is
                  // only one paramater in our
                  template
                  if
                  (parameterValue.getParameter().getN
                  ame().equals(
                              "param0")) {
                        String value =
                        parameterValue.getValue();
                        parameterValue.setValue("Info
                        "
                              +
                              value.substring(value.i
                              ndexOf(":"),
                              value.length()));
                  }
            }
      }

      // With the condition value and actions
      updated, the
      // business rule group can be published.
      // Use the original list or create a new list
      // of business rule groups
      List<BusinessRuleGroup> publishList = new
      ArrayList<BusinessRuleGroup>();

      // Add the changed business rule group to the
      list
      publishList.add(brg);

      // Publish the list with the updated business
      rule group
      BusinessRuleManager.publish(publishList, true);

      out.println("");

      // Retrieve the business rule groups again to
      verify the
      // changes were published
      out.printlnBold("Decision table after
      publish:");

      brgList =
      BusinessRuleManager.getBRGsByTNSAndName(
                  "http://BRSamples/com/ibm/websphere
                  /sample/brules",
                  QueryOperator.EQUAL,
                  "ConfigurationValues",
                  QueryOperator.EQUAL, 0, 0);

      brg = brgList.get(0);
      op = brg.getOperation("getMessages");
      ruleList = op.getAvailableTargets();

      decisionTable = (DecisionTable)
      ruleList.get(0);
      out.print(RuleArtifactUtility
                  .printDecisionTable(decisionTable))
                  ;

      }
}
```

```
      } catch (BusinessRuleManagementException e)
       {
            e.printStackTrace();
            out.println(e.getMessage());
      }
      return out.toString();
}

/*
Method to recursively navigate through a decision table and locate a
case
edge that has a template with a specific name and contains a specific
parameter to change. This method assumes that the level(depth) in the
decesion table of the value that is to be changed is known and the
current level(currentDepth) is tracked *
*/
static private CaseEdge getCaseEdge(TreeNode node, String pName,
            String templateName, int depth, int currentDepth)
{

      // Check if the current node is an action. This is an indication
      // that this branch of the decision table has been exhausted
      // looking for the case edge
      if (node instanceof ActionNode)
      {
            return null;
      }

      // Get the case edges for this node
      List<CaseEdge> caseEdges = ((ConditionNode) node).getCaseEdges();
      for (CaseEdge caseEdge : caseEdges)
      {

            // Check if the correct level has been reached
            if (currentDepth < depth)
            {
                  // Move down one level and then call getCaseEdge
                  again
                  // to process that level
                  currentDepth++;
                  return getCaseEdge(caseEdge.getChildNode(), pName,
                              templateName, depth, currentDepth);
            } else
            {
                  // The correct level has been reached. Get the
                  condition in
                  // order to check the templates on that condition on
                  whether
                  // they match the template sought
                  TreeConditionValueDefinition condition = caseEdge
                              .getValueDefinition();

                  // Get the expression for the condition which has
                  been defined
                  // with a template
                  TemplateInstanceExpression expression = condition
                              .getConditionValueTemplateInstance();
```

```
                // Get the template from the expression
                Template template = expression.getTemplate();

                // Check if this is the template sought
                if (template.getName().equals(templateName))
                {
                      // The template is found to match
                      return caseEdge;
                } else
                      caseEdge = null;
          }
    }
    return null;
}

/*
This method will check the different parameter values for an expression
and if the correct one is found, return that parameter value.
*/
private static ParameterValue getParameterValue(String pName,
            TemplateInstanceExpression expression)
{

      // Check that the expression is not null as null would indicate
      // that the expression that was passed in was probably not
      defined
      // with a template and does not have any parameters to check.
      if (expression != null) {
            // Get the parameter values for the expression
            List<ParameterValue> parameterValues = expression
                        .getParameterValues();

            for (ParameterValue parameterValue : parameterValues)
            {
                  // For the different parameters, check that it
                  matches the
                  // parameter value sought

                  if
                  (parameterValue.getParameter().getName().equals(pName
                  ))
                  {
                        // Return the parameter value that matched
                        return parameterValue;
                  }
            }
      }
      return null;
}

/*
This method finds all of the action expressions that are
defined with a specific template. It recursively works through
a case edge and adds action expressions that match to the
expressions parameter.
*/
```

```
private static void getActionExpressions(String templateName,
            CaseEdge next, List<TemplateInstanceExpression>
            expressions)
{

      ActionNode actionNode = null;
      TreeNode treeNode = next.getChildNode();

      // Check if the current node is at the action node level
      if (treeNode instanceof ConditionNode)
      {
            List<CaseEdge> caseEdges = ((ConditionNode) treeNode)
                        .getCaseEdges();

            Iterator<CaseEdge> caseEdgesIterator =
            caseEdges.iterator();

            // Work through all case edges to find the action
            // expressions
            while (caseEdgesIterator.hasNext())
            {
                  getActionExpressions(templateName,
                  caseEdgesIterator.next(),
                              expressions);
            }
      } else {
            // ActionNode found
            actionNode = (ActionNode) treeNode;

            List<TreeAction> treeActions = actionNode.getTreeActions();

            // Check that there is at least one treeAction specified
            for
            // the expression and work through the expressions checking
            // if the expressions have been created with the specific
            // template.
            if (!treeActions.isEmpty())
            {

                  Iterator<TreeAction> iterator =
                  treeActions.iterator();

                  while (iterator.hasNext())
                  {
                        TreeAction treeAction = iterator.next();
                        TemplateInstanceExpression expression =
                        treeAction
                                    .getValueTemplateInstance();

                        Template template = expression.getTemplate();

                        if (template.getName().equals(templateName))
                        {
                              // Expression found with matching
                              template
                              expressions.add(expression);
                        }
                        }
                    }
                }
            }
        }
```

Web browser output for example 12.

Executing example12

Rule set before publish:
Decision Table
Name: getMessages
Namespace: http://BRSamples/com/ibm/websphere/sample/brules

Error Code Value	10 {0} Parameter Name: param0 Parameter Value: 10	Severity	warning {0} Parameter Name: param0 Parameter Value: warning	Message for Error Code	Warning: Address not found. Parameter Name: param0 Parameter Value: Warning: Address not found.
			error {0} Parameter Name: param0 Parameter Value: error	Message for Error Code	Error: Address not found. Parameter Name: param0 Parameter Value: Error: Address not found.
	20 {0} Parameter Name: param0 Parameter Value: 20	Severity	warning {0} Parameter Name: param0 Parameter Value: warning	Message for Error Code	Warning: Insurance Policy is not available in this region. Parameter Name: param0 Parameter Value: Warning: Insurance Policy is not available in this region.
			error {0} Parameter Name: param0	Message for Error Code	Error: Insurance Policy is not available in this region. Parameter Name: param0

			Parameter Value: error		Parameter Value: Error: Insurance Policy is not available in this region.
	30 {0} Parameter Name: param0 Parameter Value: 30	Severity	warning {0} Parameter Name: param0 Parameter Value: warning	Message for Error Code	Warning: The insurance policy was not approved. Parameter Name: param0 Parameter Value: Warning: The insurance policy was not approved.
			error {0} Parameter Name: param0 Parameter Value: error	Message for Error Code	Error: The insurance policy was not approved. Parameter Name: param0 Parameter Value: Error: The insurance policy was not approved.
	Otherwise	Severity	warning {0} Parameter Name: param0 Parameter Value: warning	Message for Error Code	null
			error {0} Parameter Name: param0 Parameter Value: error	Message for Error Code	null

Decision table after publish:
Decision Table
Name: getMessages
Namespace: http://BRSamples/com/ibm/websphere/sample/brules

Error Code Value	10 {0} Parameter Name:	Severity	info {0} Parameter	Message for Error Code	Info: Address not found. Parameter Name:

	param0 Parameter Value: 10		Name: param0 Parameter Value: info		param0 Parameter Value: Info: Address not found.
			error {0} Parameter Name: param0 Parameter Value: error	Message for Error Code	Error: Address not found. Parameter Name: param0 Parameter Value: Error: Address not found.
	20 {0} Parameter Name: param0 Parameter Value: 20	Severity	info {0} Parameter Name: param0 Parameter Value: info	Message for Error Code	Info: Insurance Policy is not available in this region. Parameter Name: param0 Parameter Value: Info: Insurance Policy is not available in this region.
			error {0} Parameter Name: param0 Parameter Value: error	Message for Error Code	Error: Insurance Policy is not available in this region. Parameter Name: param0 Parameter Value: Error: Insurance Policy is not available in this region.
	30 {0} Parameter Name: param0 Parameter Value: 30	Severity	info {0} Parameter Name: param0 Parameter Value: info	Message for Error Code	Info: The insurance policy was not approved. Parameter Name: param0 Parameter Value: Info: The insurance policy was not approved.
			error {0} Parameter Name: param0	Message for Error Code	Error: The insurance policy was not approved. Parameter Name:

			Parameter Value: error		param0 Parameter Value: Error: The insurance policy was not approved.
	Otherwise	Severity	info {0} Parameter Name: param0 Parameter Value: info	Message for Error Code	null
			error {0} Parameter Name: param0 Parameter Value: error	Message for Error Code	null

3.13 Example 13: Add a condition value and actions to a decision table

In this example, a condition value and action are added to a decision table. Condition values can be added to a decision table through the use of a template. When adding a condition value to a condition node, you are actually adding a case edge. The new case edge is added at the end of the list of case edges. For the condition value, you must specify a template instance expression that has the appropriate parameter values set. In order to specify the template instance expression you will have to use a specific template. As a best practice, it is best to give the template names at each condition node level unique names in order to retrieve the correct templates for that type of condition. If a single template definition is used, it may make it difficult to determine at which level the condition is being added.

When setting the condition value in a condition node, this will actually add the condition value with the same template instance to all condition nodes at the same level. This is done as the decision table is balanced. Also as part of the adding a new condition value, new action nodes will be added. These action nodes have tree actions that have null values specified for the user presentation and template instance expression. Because the condition value can be added to a condition node that does not have an action node as a child node, the addition of a condition node may result in a large number of action nodes. The number of action nodes is based upon the level the condition node is added and the number of condition nodes at that level and the number of condition nodes at each child level.

In order to find the action nodes that have been created, a search of action nodes with tree actions that have null user presentation and template instance expression may be performed. A `TreeActionValueTemplate` can be used to create an expression that can be set into the `TreeAction`. This pattern would need to be repeated for all new action nodes.

For this example two methods were provided to assist in setting up the new tree actions. `getEmptyActionNode` recursively looks for an empty action node from the current condition node and getParameterValue returns the value of a parameter that was specified by name.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.Parameter;
import com.ibm.wbiserver.brules.mgmt.ParameterValue;
import com.ibm.wbiserver.brules.mgmt.Template;
import com.ibm.wbiserver.brules.mgmt.ValidationException;
import com.ibm.wbiserver.brules.mgmt.dtable.ActionNode;
import com.ibm.wbiserver.brules.mgmt.dtable.CaseEdge;
import com.ibm.wbiserver.brules.mgmt.dtable.ConditionNode;
import com.ibm.wbiserver.brules.mgmt.dtable.DecisionTable;
import com.ibm.wbiserver.brules.mgmt.dtable.TemplateInstanceExpression;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeAction;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeActionTermDefinition;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeActionValueTemplate;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeBlock;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeConditionValueTemplate;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeNode;
import com.ibm.wbiserver.brules.mgmt.problem.Problem;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;

public class Example13
{
      static Formatter out = new Formatter();

      static public String executeExample13()
      {
            try
            {
                  out.clear();

                  // Retrieve a business rule group by target namespace
                  // and name
                  List<BusinessRuleGroup> brgList = BusinessRuleManager
                              .getBRGsByTNSAndName(
                  "http://BRSamples/com/ibm/websphere/sample/brules",
                        QueryOperator.EQUAL,"ConfigurationValues",
                                          QueryOperator.EQUAL, 0, 0);

                  if (brgList.size() > 0)
                  {
                        // Get the first business rule group from the
                        // list.  This should be the only business
                        // rule group in the list as the combination
                        // of target namespace and name are unique
                        BusinessRuleGroup brg = brgList.get(0);

                        // Get the operation of the business rule
                        // group that has the business rule that will
                        // be modified as the business rules are
                        // associated with a specific operation
                        Operation op = brg.getOperation("getMessages");

                        // Get all business rules available for
                        // this operation
                        List<BusinessRule> ruleList =
                                          op.getAvailableTargets();

                        // For this operation there is only 1 business
                        // rule and it is the business that we want
                        // to update
```

```
DecisionTable decisionTable = (DecisionTable)
                  ruleList.get(0);

out.printlnBold("Decision table before
                  publish:");
out.print(RuleArtifactUtility
      .printDecisionTable(decisionTable));
```

The level at which the condition value is going to be added needs to be located. This is typically passed as parameter as the user interface or application using the classes knows where to add the condition.

```
// Get the tree block that contains all of
// the conditions and actions for the decision
// table
TreeBlock treeBlock =
                  decisionTable.getTreeBlock();

// From the tree block, get the tree node which
// is the starting point for navigating through
// the decision table
ConditionNode conditionNode = (ConditionNode)
                  treeBlock.getRootNode();

// Get the case edges for this node which is
// the first level of conditions
List<CaseEdge> caseEdges =
                  conditionNode.getCaseEdges();

// Get the case edge which will have the new
// condition added
CaseEdge caseEdge = caseEdges.get(0);

// For the case edge get the condition node in
// order to retrieve the templates for the
// condition
conditionNode = (ConditionNode)
                  caseEdge.getChildNode();

// Get the templates for the condition
List<TreeConditionValueTemplate>
treeValueConditionTemplates = conditionNode
            .getAvailableValueTemplates();

Iterator<TreeConditionValueTemplate>
      treeValueConditionTemplateIterator =
      treeValueConditionTemplates.iterator();

TreeConditionValueTemplate conditionTemplate =
                  null;
```

By using unique template names at each condition node level in the decision table, you can more easily insure the condition value is being added at the correct condition node value.

```
// Find the template that should be used
while
(treeValueConditionTemplateIterator.hasNext())
{
      conditionTemplate =
            treeValueConditionTemplateIterator
                  .next();
      if (conditionTemplate.getName().equals(
                  "Condition Value Template
                  2.1"))
      {
            // Template found
            break;
      }
      conditionTemplate = null;
}

if (conditionTemplate != null)
{
```

With the correct template found, an instance can be created and the appropriate parameter value set before it is added to the condition node.

```
      // Get the parameter definition from the
      // template
      Parameter conditionParameter =
      conditionTemplate.getParameter("param0");

      // Create a parameter value instance to
      // be used in a new condition template
      // instance
      ParameterValue conditionParameterValue =
            conditionParameter
            .createParameterValue("fatal");

      List<ParameterValue>
            conditionParameterValues = new
            ArrayList<ParameterValue>();

      // Add the parameter value to a list

      conditionParameterValues
            .add(conditionParameterValue);

      // Create a new condition template
      // instance with the parameter value
      TemplateInstanceExpression
            newConditionValue =
            conditionTemplate
            .createTemplateInstanceExpression(c
      onditionParameterValues);

      // Add the condition template instance to
      // this condition node
      conditionNode
```

```
                .addConditionValueToThisLevel(newConditionValue);

                            // When a condition node is added there
                            // are new action nodes that are created
                            // and empty. These must be filled with
                            // action template instances. By
                            // searching for each empty action
                            // node from the parent level, all of the
                            // new empty action nodes can be found.
                            conditionNode = (ConditionNode)
                                  conditionNode.getParentNode();
```

With the condition value added to the condition node, the tree actions in the new action nodes must be set with a `TreeActionValueTemplate`. First locate the empty action node for the case edges. Use the parent condition node to insure that as you iterate through the condition nodes, you will pick up all action nodes.

```
                            // Get the case edges for the parent node
                            caseEdges = conditionNode.getCaseEdges();

                            Iterator<CaseEdge> caseEdgesIterator =
                                        caseEdges.iterator();

                            while (caseEdgesIterator.hasNext())
                            {
                                  // For each case edge, retrieve an
                                  // empty action node if it exists
                                  ActionNode actionNode =
                            getEmptyActionNode(caseEdgesIterator
                                                  .next());

                                  // Check if all actions are filled
                                  if (actionNode != null)
                                  {
```

When an action node is found with empty tree actions, the tree action must be set with a `TreeActionValueTemplate`. First locate the template and then specify the parameters before creating a template instance. With the template instance created, the tree action can be updated. For this example the parameter was set with a value from another tree action in another action node under the same condition node. For other decision tables where another tree action might not have a value that may be used to create the new parameter values, the value will have to be passed as a parameter from the application.

```
                                        // Get the list of tree
                                        // actions. These
                                        // are not the actual
                                        // actions, but the
                                        // placeholders for the
                                        // actions
                                        List<TreeAction>
                                        treeActionList = actionNode
                                        .getTreeActions();

                                  List<TreeActionTermDefinition>
```

```
treeActionTermDefinitions =
      treeBlock
.getTreeActionTermDefinitions();

List<TreeActionValueTemplate>
      treeActionValueTemplates =
treeActionTermDefinitions
      .get(0).getValueTemplates();

TreeActionValueTemplate
            actionTemplate = null;

      for (TreeActionValueTemplate
            tempActionTemplate :
      treeActionValueTemplates)
      {

            if
            (tempActionTemplate.get
            Name().equals(
                  "Action Value
                  Template 1"))
            {
                  actionTemplate =
            tempActionTemplate;
                  break;
            }
      }

      if (actionTemplate != null)
      {
            // Get another action
            // that is under
            // the parent condition
            // node in order
            // to use the value as
            // the basis for
            // the error message in
            // the new
            // action node. Move up
            // to the
            // parent condition
            // node first
            ConditionNode
            parentNode =
            (ConditionNode)
            actionNode
                  .getParentNode();

            // Get the first case
            // edge of the
            // parent node as this
            // action will
            // always be filled in
            // as new actions
            // are added to the end
            // of the case
            // edge list.
            CaseEdge caseE =
                  parentNode.getCas
            eEdges().get(
                        0);

            // The child node is an
            // action node
            // and at the same
            // level as the new
            // action node.
            ActionNode aNode =
            (ActionNode) caseE
                  .getChildNode();

            // Get the list of tree
            // actions
            TreeAction
            existingTreeAction =
            aNode
                  .getTreeActions()
            .get(0);

            // Get the template
            // instance
            // expression for the
            // tree action
            // from which you can
            // retrieve the
            // parameter

      TemplateInstanceExpression
      existingExpression =
            existingTreeAction
      .getValueTemplateInstance();

            ParameterValue
      existingParameterValue =
      getParameterValue(
            "param0",
            existingExpression);

            String actionValue =
            existingParameterValue
                  .getValue();

            // Create the new
            // message from the
            // message of the
            // existing
            // tree action
            actionValue = "Fatal"
                        +
actionValue.substring(actionValue
      .indexOf(":"), actionValue
                        .length());
```

```
                                          Parameter
                                          actionParameter =
                                          actionTemplate
                                    .getParameter("param0");

                                          // Get the parameter
                                          // from the template
                                          ParameterValue
                                          actionParameterValue =
                                                actionParameter
                              .createParameterValue(actionValue);

                                          // Add the parameter to
                                          // a list of templates
                                          List<ParameterValue>
                                    actionParameterValues = new
                                    ArrayList<ParameterValue>();

        actionParameterValues.add(actionParameterValue);

                                          // Create a new tree
                                          // action instance

                                    TemplateInstanceExpression
                                    treeAction = actionTemplate
  .createTemplateInstanceExpression(actionParameterValues);

                                          // Set the tree action
                                          // in the action node
                                          // by setting it in the
                                          // tree action list
```

Here the tree action in the action node is updated.

```
                                    treeActionList.get(0)
                                    .setValueTemplateInstance(
                                    treeAction);
                                    }
                              }
                        }
                  }

                  // With the condition value and actions
                  // updated, the business rule group can be
                  // published.
                  // Use the original list or create a new list
                  // of business rule groups
                  List<BusinessRuleGroup> publishList = new
                              ArrayList<BusinessRuleGroup>();

                  // Add the changed business rule group to the
                  // list
                  publishList.add(brg);

                  // Publish the list with the updated business
                  // rule group
                  BusinessRuleManager.publish(publishList, true);

                  brgList =
                        BusinessRuleManager.getBRGsByTNSAndName(
            "http://BRSamples/com/ibm/websphere/sample/brules",
                  QueryOperator.EQUAL, “ConfigurationValues",
                              QueryOperator.EQUAL, 0, 0);
                  brg = brgList.get(0);
                  op = brg.getOperation("getMessages");
                  ruleList = op.getAvailableTargets();
                  decisionTable = (DecisionTable)
                              ruleList.get(0);
                  out.printlnBold("Decision table after
                              publish:");
                  out
                              .print(RuleArtifactUtility
                        .printDecisionTable(decisionTable));
            }
      } catch (ValidationException e)
      {
            List<Problem> problems = e.getProblems();

            out.println("Problem = " +
                  problems.get(0).getErrorType().name());

            e.printStackTrace();
            out.println(e.getMessage());
      } catch (BusinessRuleManagementException e)
      {
            e.printStackTrace();
            out.println(e.getMessage());
      }
      return out.toString();
}

/*
 * This method searches from the current case edge for any
 * action nodes that have empty tree actions. An empty
 * action node is found by looking at the end of the list
 * of case edges and checking if the action node has tree
 * actions that have both a null user presentation and
 * TemplateInstanceExpression.
 */
private static ActionNode getEmptyActionNode(CaseEdge next)
{

      ActionNode actionNode = null;
      TreeNode treeNode = next.getChildNode();

      if (treeNode instanceof ConditionNode)
      {
            List<CaseEdge> caseEdges = ((ConditionNode) treeNode)
                        .getCaseEdges();

            if (caseEdges.size() > 1)
            {
                  // Get right-most case-edge as the new
```

```
                        // condition and thus empty actions are at the
                        // right-end of the case edges
                        actionNode = getEmptyActionNode(caseEdges
                                    .get(caseEdges.size() - 1));

                        if (actionNode != null)
                        {
                              return actionNode;
                        }
                  }
            } else
            {
                  actionNode = (ActionNode) treeNode;

                  List<TreeAction> treeActions =
                  actionNode.getTreeActions();

                  if (!treeActions.isEmpty())
                  {
                        if
            ((treeActions.get(0).getValueUserPresentation() == null)
                                    &&
(treeActions.get(0).getValueTemplateInstance() == null))
                        {
                              return actionNode;
                        }
                  }
                  actionNode = null;
            }
            return actionNode;
      }

      /*
       * This method will check the different parameter values for an
       * expression and if the correct one is found, return that
       * parameter value.
       */
      private static ParameterValue getParameterValue(String pName,
                  TemplateInstanceExpression expression)
      {
            ParameterValue parameterValue = null;

            // Check that the expression is not null as null would
            // indicate that the expression that was passed in was
            // probably not defined with a template and does not have
            // any parameters to check.
            if (expression != null)
            {
                  // Get the parameter vlues for the expression
                  List<ParameterValue> parameterValues = expression
                              .getParameterValues();

                  Iterator<ParameterValue> parameterIterator =
parameterValues
                              .iterator();

                  // For the different parameters, check that it
                  // matches the parameter value sought
                  while (parameterIterator.hasNext())
                  {
                        parameterValue = parameterIterator.next();

                        if
(parameterValue.getParameter().getName().equals(pName))
                        {
                              // Return the parameter value that
                              // matched
                              return parameterValue;
                        }
                  }
            }
            return parameterValue;
      }
}
```

Web browser output for example 13.

Executing example13

Decision table before publish:
Decision Table
Name: getMessages
Namespace: http://BRSamples/com/ibm/websphere/sample/brules

Error Code Value	10 {0} Parameter Name: param0 Parameter Value: 10	Severity	warning {0} Parameter Name: param0 Parameter Value: warning	Message for Error Code	Warning: Address not found. Parameter Name: param0 Parameter Value: Warning: Address not found.
			error {0} Parameter Name: param0 Parameter Value: error	Message for Error Code	Error: Address not found. Parameter Name: param0 Parameter Value: Error: Address not found.
	20 {0} Parameter Name: param0 Parameter Value: 20	Severity	warning {0} Parameter Name: param0 Parameter	Message for Error Code	Warning: Insurance Policy is not available in this region. Parameter Name: param0 Parameter Value:

			Value: warning		Warning: Insurance Policy is not available in this region.
			error {0} Parameter Name: param0 Parameter Value: error	Message for Error Code	Error: Insurance Policy is not available in this region. Parameter Name: param0 Parameter Value: Error: Insurance Policy is not available in this region.
	30 {0} Parameter Name: param0 Parameter Value: 30	Severity	warning {0} Parameter Name: param0 Parameter Value: warning	Message for Error Code	Warning: The insurance policy was not approved. Parameter Name: param0 Parameter Value: Warning: The insurance policy was not approved.
			error {0} Parameter Name: param0 Parameter Value: error	Message for Error Code	Error: The insurance policy was not approved. Parameter Name: param0 Parameter Value: Error: The insurance policy was not approved.
	Otherwise	Severity	warning {0} Parameter Name: param0 Parameter Value: warning	Message for Error Code	null
			error {0} Parameter Name: param0 Parameter Value: error	Message for Error Code	null

Decision table after publish:

Decision Table
Name: getMessages
Namespace: http://BRSamples/com/ibm/websphere/sample/brules

Error Code Value	10 {0} Parameter Name: param0 Parameter Value: 10	Severity	warning {0} Parameter Name: param0 Parameter Value: warning	Message for Error Code	Warning: Address not found. Parameter Name: param0 Parameter Value: Warning: Address not found.
			error {0} Parameter Name: param0 Parameter Value: error	Message for Error Code	Error: Address not found. Parameter Name: param0 Parameter Value: Error: Address not found.
			fatal {0} Parameter Name: param0 Parameter Value: fatal	Message for Error Code	Fatal: Address not found. Parameter Name: param0 Parameter Value: Fatal: Address not found.
	20 {0} Parameter Name: param0 Parameter Value: 20	Severity	warning {0} Parameter Name: param0 Parameter Value: warning	Message for Error Code	Warning: Insurance Policy is not available in this region. Parameter Name: param0 Parameter Value: Warning: Insurance Policy is not available in this region.
			error {0} Parameter Name: param0 Parameter Value: error	Message for Error Code	Error: Insurance Policy is not available in this region. Parameter Name: param0 Parameter Value: Error: Insurance Policy is not available in this region.

			fatal {0} Parameter Name: param0 Parameter Value: fatal	Message for Error Code	Fatal: Insurance Policy is not available in this region. Parameter Name: param0 Parameter Value: Fatal: Insurance Policy is not available in this region.
	30 {0} Parameter Name: param0 Parameter Value: 30	Severity	warning {0} Parameter Name: param0 Parameter Value: warning	Message for Error Code	Warning: The insurance policy was not approved. Parameter Name: param0 Parameter Value: Warning: The insurance policy was not approved.
			error {0} Parameter Name: param0 Parameter Value: error	Message for Error Code	Error: The insurance policy was not approved. Parameter Name: param0 Parameter Value: Error: The insurance policy was not approved.
			fatal {0} Parameter Name: param0 Parameter Value: fatal	Message for Error Code	Fatal: The insurance policy was not approved. Parameter Name: param0 Parameter Value: Fatal: The insurance policy was not approved.
	Otherwise	Severity	warning {0} Parameter Name: param0 Parameter Value: warning	Message for Error Code	null

			error {0} Parameter Name: param0 Parameter Value: error	Message for Error Code	null
			fatal {0} Parameter Name: param0 Parameter Value: fatal	Message for Error Code	null

3.14 Example 14: Handle errors in a rule set

This example focuses on how to catch problems in a rule set and find out what problem has occurred such that the appropriate message can be displayed or action can be taken to correct the situation.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.ParameterValue;
import com.ibm.wbiserver.brules.mgmt.ValidationException;
import com.ibm.wbiserver.brules.mgmt.problem.Problem;
import
com.ibm.wbiserver.brules.mgmt.problem.ProblemStartDateAfterEndDate;
import com.ibm.wbiserver.brules.mgmt.problem.ValidationError;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleBlock;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSet;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetRule;
import
com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetTemplateInstanceRule;

public class Example14 {
static Formatter out = new Formatter();

static public String executeExample14() {
      try {
            out.clear();

            // Retrieve a business rule group by target namespace and
            name
            List<BusinessRuleGroup> brgList = BusinessRuleManager
                        .getBRGsByTNSAndName(
                                    "http://BRSamples/com/ibm/websphere
                                    /sample/brules",
                                    QueryOperator.EQUAL,
                                    "ApprovalValues",
                                    QueryOperator.EQUAL, 0, 0);

            if (brgList.size() > 0) {
                  // Get the first business rule group from the list
                  // This should be the only business rule group in the
                  list as
                  // the combination of target namespace and name are
                  unique
                  BusinessRuleGroup brg = brgList.get(0);
                  out.println("Business Rule Group retrieved");
```

```
                    // Get the operation of the business rule group that
                    // has the business rule that will be modified as
                    // the business rules are associated with a specific
                    // operation
                    Operation op = brg.getOperation("getApprover");

                    // Retrieve specific rule by name
                    List<BusinessRule> ruleList =
                    op.getBusinessRulesByName(
                                "getApprover", QueryOperator.EQUAL, 0,
                                0);

                    // Get the specific rule
                    RuleSet ruleSet = (RuleSet) ruleList.get(0);
                    out.println("Rule Set retrieved");

                    RuleBlock ruleBlock = ruleSet.getFirstRuleBlock();

                    Iterator<RuleSetRule> ruleIterator =
                    ruleBlock.iterator();

                    // Search through the rules to find the rule to
                    change
                    while (ruleIterator.hasNext()) {
                          RuleSetRule rule = ruleIterator.next();

                          // Check that the rule was defined with a
                          template
                          // as it can be changed.
                          if (rule instanceof
                          RuleSetTemplateInstanceRule) {
                                // Get the template rule instance
                                RuleSetTemplateInstanceRule
                                templateInstance =
                                (RuleSetTemplateInstanceRule) rule;

                                // Check for the correct template rule
                                instance
                                if (templateInstance.getName().equals(
                                            "LargeOrderApprover")) {
```

To cause a problem, this example sets a parameter to a value that is not compatible for the expression. The parameter is expecting an integer, but a string is passed in.

```
                                      // Get the parameter from the
                                      template instance
                                      ParameterValue parameter =
                                      templateInstance
                                                  .getParameterValue("par
                                                  am1");

                                      // Set an incorrect value for this
                                      parameter
                                      // This will cause a validation
                                      error
```

```
                                    parameter.setValue("$3500");
                                    out.println("Incorrect parameter
                                    value set");
                                    break;
                              }
                        }
                  }

                  // This code should never be reached because of the
                  error
                  // introduced
                  // above

                  // With the condition value and actions updated, the
                  business
                  // rule
                  // group can be published.
                  // Use the original list or create a new list
                  // of business rule groups
                  List<BusinessRuleGroup> publishList = new
                  ArrayList<BusinessRuleGroup>();

                  // Add the changed business rule group to the list
                  publishList.add(brg);

                  // Publish the list with the updated business rule
                  group
                  BusinessRuleManager.publish(publishList, true);
            }
```

A `ValidationException` can be caught and from the exception, the problems can be retrieved. For each problem, the error can be checked to determine which error has occurred. A message can be printed out or the appropriate action can be taken.

```
      } catch (ValidationException e) {
            out.println("Validation Error");

            List<Problem> problems = e.getProblems();

            Iterator<Problem> problemIterator = problems.iterator();

            // Check the list of problems for the appropriate error and
            // perform the appropriate action (report error, correct
            // error, etc.)
            while (problemIterator.hasNext()) {
                  Problem problem = problemIterator.next();
                  ValidationError error = problem.getErrorType();

                  // Check for specific error value
                  if (error == ValidationError.TYPE_CONVERSION_ERROR) {
                        // Handle this error by reporting the problem
                        out
                                    .println("Problem: Incorrect value
                                    entered for a parameter");
                        return out.toString();
                  }
                  // else if....
                  // Checks can be done for other errors and the
                  // appropriate error message or action can be
                  performed
                  // correct the problem
            }
      } catch (BusinessRuleManagementException e) {
            out.println("Error occurred.");
            e.printStackTrace();
      }
      return out.toString();
}
}
```

Web browser output for example 14.

Executing example14

Business Rule Group retrieved
Rule Set retrieved
Validation Error
Problem: Incorrect value entered for a parameter

3.15 Example 15: Handle errors in a business rule group

This example is similar to example 14 as it shows how to handle problems that occur when a business rule group is published. It shows how the problem can be determined and the correct message can be printed or action performed.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.ArrayList;
import java.util.Date;
import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.OperationSelectionRecord;
import com.ibm.wbiserver.brules.mgmt.OperationSelectionRecordList;
import com.ibm.wbiserver.brules.mgmt.ParameterValue;
import com.ibm.wbiserver.brules.mgmt.ValidationException;
import com.ibm.wbiserver.brules.mgmt.problem.Problem;
import
com.ibm.wbiserver.brules.mgmt.problem.ProblemStartDateAfterEndDate;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleBlock;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSet;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetRule;
import
com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetTemplateInstanceRule;

public class Example15
{
static Formatter out = new Formatter();

static public String executeExample15()
{
      try
      {
            out.clear();

            // Retrieve a business rule group by target namespace and
            name
            List<BusinessRuleGroup> brgList = BusinessRuleManager
                        .getBRGsByTNSAndName(
                                    "http://BRSamples/com/ibm/websphere
                                    /sample/brules",
                                    QueryOperator.EQUAL,
                                    "ApprovalValues",
                                    QueryOperator.EQUAL, 0, 0);
            if (brgList.size() > 0)
            {
                  // Get the first business rule group from the list
                  // This should be the only business rule group in the
                  list as
                  // the combination of target namespace and name are
                  unique
                  BusinessRuleGroup brg = brgList.get(0);
                  out.println("Business Rule Group retrieved");

                  // Get the operation of the business rule group that
                  // has the business rule that will be modified as
                  // the business rules are associated with a specific
                  // operation
                  Operation op = brg.getOperation("getApprover");

                  // Retrieve specific rule by name
                  List<BusinessRule> ruleList =
                  op.getBusinessRulesByName(
                              "getApprover", QueryOperator.EQUAL, 0,
                              0);

                  // Get the specific rule
                  RuleSet ruleSet = (RuleSet) ruleList.get(0);
                  out.println("Rule Set retrieved");

                  RuleBlock ruleBlock = ruleSet.getFirstRuleBlock();

                  Iterator<RuleSetRule> ruleIterator =
                  ruleBlock.iterator();

                  // Search through the rules to find the rule to
                  change
                  while (ruleIterator.hasNext())
                  {
                        RuleSetRule rule = ruleIterator.next();

                        // Check that the rule was defined with a
                        template
                        // as it can be changed.
                        if (rule instanceof
                        RuleSetTemplateInstanceRule)
                        {
                              // Get the template rule instance
                              RuleSetTemplateInstanceRule
                              templateInstance =
                              (RuleSetTemplateInstanceRule) rule;

                              // Check for the correct template rule
                              instance
                              if (templateInstance.getName().equals(
                                          "LargeOrderApprover"))
                              {
                                    // Get the parameter from the
                                    template instance
                                    ParameterValue parameter =
                                    templateInstance
                                                .getParameterValue("par
                                                am1");
                                    // Set the value for this parameter
                                    // This value is in the correct
                                    format and will
                                    // not cause a validation error
                                    parameter.setValue("4000");
                                    out.println("Rule set parameter
                                    value on set correctly");
                                    break;
                              }
                        }
                  }
```

To insure a rule set is correct, the validate method can be called. The validate method is available on all objects and will return a list of problems that can be checked to determine the problem.   When calling validate on an object, the validate method is called on all contained objects as well.

```
      // Validate the changes made the rule set
      List<Problem> problems = ruleSet.validate();
      out.println("Rule set validated");

      // No errors should occur for this test case,
      however, as a
      // best practice, check if there are problems and
      then
      // perform the correct action to recover or report
      the error
      if (problems != null)
      {
            Iterator<Problem> problemIterator =
            problems.iterator();

            while (problemIterator.hasNext())
            {
                  Problem problem = problemIterator.next();

                  if (problem instanceof
                  ProblemStartDateAfterEndDate)
                  {
                        out
                                    .println("Incorrect
                                    value entered for a
                                    parameter");
                        return out.toString();
                  }
            }
      } else
      {
            out.println("No problems found for the rule
            set");
      }
      // Get the list of available rule targets
      List<BusinessRule> ruleList2 =
      op.getAvailableTargets();
```

```
// Get the first rule that will be scheduled
incorrectly
BusinessRule rule = ruleList2.get(0);


// The error condition will be to set the end time
for a
// scheduled rule to be 1 hour before the start time
// This will cause a validation error
Date future = new Date();
long futureTime = future.getTime() - 360000;


// Get the operation selection list to add the
incorrectly
// scheduled item
OperationSelectionRecordList opList = op
            .getOperationSelectionRecordList();



// Create a new scheduled rule instance
// No error is thrown until validated or a publish
// occurs as more changes might be made
OperationSelectionRecord newRecord = opList
            .newOperationSelectionRecord(new Date(),
            new Date(
                        futureTime), rule);
```

When the record is added with an incorrect set of dates, this does not cause an error. It is possible overlaps might occur or no selection records are set for the operation as things are in the process of being changed. The error will be found when the business rule group with the operation selection record is published. The validate method is called before the objects are published and exceptions will be thrown if any errors exists.

```
// Add the scheduled rule instance to the operation
// No error here either
opList.addOperationSelectionRecord(newRecord);
out.println("New selection record added with
incorrect schedule");

// With the condition value and actions updated, the
business
// rule
// group can be published.
// Use the original list or create a new list
// of business rule groups
List<BusinessRuleGroup> publishList = new
ArrayList<BusinessRuleGroup>();

// Add the changed business rule group to the list
publishList.add(brg);

// Publish the list with the updated business rule
group
BusinessRuleManager.publish(publishList, true);
```

```
            }
      } catch (ValidationException e) {
            out.println("Validation Error");

            List<Problem> problems = e.getProblems();

            Iterator<Problem> problemIterator = problems.iterator();
            // There might be multiple problems
            // Go through the problems and handle each one or
            // report the problem
            while (problemIterator.hasNext())
            {
                  Problem problem = problemIterator.next();

                  // Each problem is a different type that can be
                  compared
                  if (problem instanceof ProblemStartDateAfterEndDate)
                  {
                        out
                                    .println("Rule schedule is
                                    incorrect.  Start date is after end
                                    date.");
                        return out.toString();
                  }
                  // else if....
                  // Checks can be done for other errors and the
                  // appropriate error message or action can be
                  performed
                  // correct the problem
            }
}catch (BusinessRuleManagementException e)
{
            out.println("Error occurred.");
            e.printStackTrace();
      }
      return out.toString();
   }
}
```

Web browser output for example 15.

Executing example15

```
Business Rule Group retrieved
Rule Set retrieved
Rule set parameter value on set correctly
Rule set validated
Validation Error
Rule schedule is incorrect. Start date is after end date.
```

4.0 Appendix

4.1 Formatter class

This class provides different methods to help with displaying the different examples. It adds different HTML tags to format the output.

```
package com.ibm.websphere.sample.brules.mgmt;

public class Formatter {

private StringBuffer buffer;

public Formatter()
{
      buffer = new StringBuffer();
}

public void println(Object o)
{
      buffer.append(o);
      buffer.append("<br>\n");
            }

            public void print(Object o)
            {
                        buffer.append(o);
            }

            public void printlnBold(Object o)
            {
                        buffer.append("<b>");
                        buffer.append(o);
                        buffer.append("</b><br>\n");
            }

            public void printBold(Object o)
            {
                        buffer.append("<b>");
                        buffer.append(o);
                        buffer.append("</b>");
            }

            public String toString()
            {
                        return buffer.toString();
            }

            public void clear()
            {
                        buffer = new StringBuffer();
            }
}
```

4.2 RuleArtifactUtility class

This utility class has two public methods. The first public method is for printing out a decision table. This method makes use of a private method that uses recursion to print out the conditions and actions for the decision table. The second public method is for printing out a rule set.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.Parameter;
import com.ibm.wbiserver.brules.mgmt.ParameterValue;
import com.ibm.wbiserver.brules.mgmt.RuleTemplate;
import com.ibm.wbiserver.brules.mgmt.Template;
import com.ibm.wbiserver.brules.mgmt.dtable.ActionNode;
import com.ibm.wbiserver.brules.mgmt.dtable.CaseEdge;
import com.ibm.wbiserver.brules.mgmt.dtable.ConditionNode;
import com.ibm.wbiserver.brules.mgmt.dtable.DecisionTable;
import com.ibm.wbiserver.brules.mgmt.dtable.DecisionTableRule;
import
com.ibm.wbiserver.brules.mgmt.dtable.DecisionTableTemplateInstanceRule;
import com.ibm.wbiserver.brules.mgmt.dtable.TemplateInstanceExpression;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeAction;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeActionTermDefinition;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeBlock;
import
com.ibm.wbiserver.brules.mgmt.dtable.TreeConditionTermDefinition;
import
com.ibm.wbiserver.brules.mgmt.dtable.TreeConditionValueDefinition;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeNode;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleBlock;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSet;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetRule;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetRuleTemplate;
import
com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetTemplateInstanceRule;

public class RuleArtifactUtility
{

            static Formatter out = new Formatter();

            /*
      Method to print out a decision table with the conditions and
      actions printed out in a HTML tabular format.  The conditions
      and actions are printed out with a separate method that
      recursively works through the case edges of the decision
```

```
tables.
            */
      public static String printDecisionTable(BusinessRule
      ruleArtifact)
      {
                  out.clear();
                  out.printlnBold("Decision Table");
                  DecisionTable decisionTable = (DecisionTable)
                  ruleArtifact;
                  out.println("Name: " +
                  decisionTable.getName());
                  out.println("Namespace: " +
                  decisionTable.getTargetNameSpace());

                  // Output the init rule for the decision table
                  before
                  // working through the table of conditions and
                  actions
                  DecisionTableRule initRule =
                  decisionTable.getInitRule();
                  if (initRule != null)
                  {
                              out.printBold("Init Rule: ");
                              out.println(initRule.getName());
                              out.println("Display Name: " +
                              initRule.getDisplayName());
                              out.println("Description: " +
                              initRule.getDescription());
                              // The expanded user presentation
                              will automatically populate the
                              // presentation with the parameter
                              values and can be used for
                              // display if the init rule was
                              defined with a template. If no
                              // template was defined the
                              expanded user presentation
                              // is the same as the regular
                              presentation.
                              out.println("Extended User
                              Presentation: "
                                          +
                                          initRule.getExpandedUse
                                          rPresentation());
                              // The regular user presentation
                              will have placeholders in the
                              // string where the
                              // parameter can be substituted if
                              the init rule was defined with a
                              // template
                              // If the rule was not defined with
                              a template, the user
                              // presentation will only
                              // be a string without
                              placeholders. The placeholders are
                              of a
                              // format of {n} where
// n is the index (zero-based) of
the parameter in the template. This
// value
// can be used to create an
interface for editing where there
are
// fields with
// the parameter values available
for editing

out.println("User Presentation: " +
initRule.getUserPresentation());

// Init rules might be defined with
or without a template
// Check to make sure a template
was used before trying
// to access the parameters
if (initRule instanceof
DecisionTableTemplateInstanceRule)
{
            DecisionTableTemplateIn
            stanceRule
            templateInstance =
            (DecisionTableTemplateI
            nstanceRule) initRule;

            RuleTemplate template =
            templateInstance.getRul
            eTemplate();

            List<Parameter>
            parameters =
            template.getParameters(
            );
            Iterator<Parameter>
            paramIterator =
            parameters.iterator();

            Parameter parameter =
            null;

            while
            (paramIterator.hasNext(
            )) {
            parameter =
            paramIterator.next();

            out.println("Parameter
            Name: " +
            parameter.getName());
            out.println("Parameter
            Value: "
            +
            templateInstance.getPar
            ameterValue(parameter
            .getName()));
```

```
                                    }
                        }
            }
            // For the rest of the decision table, start at
            the root and
            // recursively work through the different case
            edges and
            // actions
            TreeBlock treeBlock =
            decisionTable.getTreeBlock();
            TreeNode treeNode = treeBlock.getRootNode();

            printDecisionTableConditionsAndActions(treeNode
            , 0);
            out.println("");
            return out.toString();
      }

      /*
Method to recursively work through the case edges and print
out the conditions and actions
            */
      static private void printDecisionTableConditionsAndActions(
                        TreeNode treeNode, int indent)
      {
            out.print("<table border=\"1\">");
            if (treeNode instanceof ConditionNode)
            {
                        // Get the case edges for the
                        current TreeNode
                        // and for each case edge print out
                        the conditions
                        ConditionNode conditionNode =
                        (ConditionNode) treeNode;

                        List<CaseEdge> caseEdges =
                        conditionNode.getCaseEdges();
                        Iterator<CaseEdge> caseEdgeIterator
                        = caseEdges.iterator();

                        CaseEdge caseEdge = null;

                        while (caseEdgeIterator.hasNext())
                        {
                                    out.print("<tr>");
                                    // If this is the start
                                    of the conditions for
                                    the
                                    // condition node,
                                    print out the condition
                                    term
                                    if (indent == 0)
                                    {
                                    out.print("<td>");
```

```
TreeConditionTermDefini
tion termDefinition =
conditionNode
.getTermDefinition();

out.print(termDefinitio
n.getUserPresentation()
);
out.print("</td>");
indent++;
} else {
// After the condition
term has been printed
for a
// case edge skip for
the rest of the case
edges
out.print("<td></td>");
}

caseEdge =
caseEdgeIterator.next()
;

out.print("<td>");

// Check if the
caseEdge is defined by
a template
if
(caseEdge.getValueDefin
ition() != null)
{
TemplateInstanceExpress
ion templateInstance =
caseEdge
.getValueTemplateInstan
ce();

out.println(templateIns
tance.getExpandedUserPr
esentation());

TreeConditionValueDefin
ition valueDef =
caseEdge
.getValueDefinition();

out.println(valueDef.ge
tUserPresentation());

Template template =
templateInstance.getTem
plate();
```

```
// Get the parameters
for the template
definition and
// print out the
parameter names and
values
List<Parameter>
parameters =
template.getParameters(
);
Iterator<Parameter>
paramIterator =
parameters.iterator();

List<ParameterValue>
parameterValues =
templateInstance
.getParameterValues();
Iterator<ParameterValue
> paramValues =
parameterValues
.iterator();

Parameter parameter =
null;
ParameterValue
parameterValue = null;

while
(paramIterator.hasNext(
) &&
paramValues.hasNext())
{
parameter =
paramIterator.next();
parameterValue =
paramValues.next();

out.println("Parameter
Name: " +
parameter.getName());
out.println("Parameter
Value: "
+
parameterValue.getValue
());
}
}

out.print("</td><td>");
// Print the child node
for the caseEdge
printDecisionTableCondi
tionsAndActions(caseEdg
e.getChildNode(),
0);
```

```
                        out.print("</td></tr>")
                        ;
            }

            // Add Otherwise condition if it
            exists
            TreeNode otherwise =
            conditionNode.getOtherwiseCase();

            if (otherwise != null)
            {
                        out.print("<tr><td></td
                        ><td>Otherwise</td><td>
                        ");
                        // Print the Otherwise
                        ConditionNode
                        printDecisionTableCondi
                        tionsAndActions(otherwi
                        se, 0);
                        out.print("</td></tr>")
                        ;
            }
            out.print("</table>");
} else {
            // ActionNode has been found and
            different logic is needed
            // to print out the TreeActions
            ActionNode actionNode =
            (ActionNode) treeNode;
            List<TreeAction> treeActions =
            actionNode.getTreeActions();

            Iterator<TreeAction>
            treeActionIterator =
            treeActions.iterator();

            TreeAction treeAction = null;

            // The ActionNode can contain
            multiple TreeActions to
            // print out
            while
            (treeActionIterator.hasNext())
            {
                        out.print("<tr>");
                        treeAction =
                        treeActionIterator.next
                        ();

                        TreeActionTermDefinitio
                        n treeActionTerm =
                        treeAction
                        .getTermDefinition();

                        if (indent == 0) {
                        out.print("<td>");
                        out.print(treeActionTer
                        m.getUserPresentation()
                        );
                        out.print("</td>");
                        }
                        out.print("<td>");
                        TemplateInstanceExpress
                        ion templateInstance =
                        treeAction
                        .getValueTemplateInstan
                        ce();

                        // Check that a
                        template was specified
                        for
                        // the TreeAction
                        before working with the
                        // parameter name and
                        values
                        if (templateInstance !=
                        null) {
                        out.println(templateIns
                        tance.getExpandedUserPr
                        esentation());

                        Template template =
                        templateInstance.getTem
                        plate();

                        List<Parameter>
                        parameters =
                        template.getParameters(
                        );
                        Iterator<Parameter>
                        paramIterator =
                        parameters.iterator();

                        List<ParameterValue>
                        parameterValues =
                        templateInstance
                        .getParameterValues();
                        Iterator<ParameterValue
                        > paramValues =
                        parameterValues
                        .iterator();

                        Parameter parameter =
                        null;
                        ParameterValue
                        parameterValue = null;

                        while
                        (paramIterator.hasNext(
                        ) &&
                        paramValues.hasNext())
                        {
```

```
                                                    parameter =
                                                    paramIterator.next();
                                                    parameterValue =
                                                    paramValues.next();

                                                    out.println(" Parameter
                                                    Name: " +
                                                    parameter.getName());
                                                    out.println(" Parameter
                                                    Value: "
                                                    +
                                                    parameterValue.getValue
                                                    ());

                                                    }
                                                    } else
                                                    {
                                                    // If a template was
                                                    not used, the only item
                                                    that is
                                                    // available is the
                                                    UserPresentation if it
                                                    was
                                                    // specified when the
                                                    rule was created
                                                    out.print(treeAction.ge
                                                    tValueUserPresentation(
                                                    ));
                                                    }

                                                    out.print("</td></tr>")
                                                    ;
                                    }
                                    out.print("</table>");
                        }
            }

            /*
      Method to print out a rule set
                  */
            public static String printRuleSet(BusinessRule
            ruleArtifact)
            {
                        out.clear();
                        out.printlnBold("Rule Set");
                        RuleSet ruleSet = (RuleSet) ruleArtifact;
                        out.println("Name: " + ruleSet.getName());
                        out.println("Namespace: " +
                        ruleSet.getTargetNameSpace());

                        // The rules in a rule set are contained in a
                        rule block
                        RuleBlock ruleBlock =
                        ruleSet.getFirstRuleBlock();

                        Iterator<RuleSetRule> ruleIterator =
                        ruleBlock.iterator();
```

```
RuleSetRule rule = null;

// Iterate through the rules in the rule block.
while (ruleIterator.hasNext())
{
            rule = ruleIterator.next();
            out.printBold("Rule: ");
            out.println(rule.getName());
            out.println("Display Name: " +
            rule.getDisplayName());
            out.println("Description: " +
            rule.getDescription());
            // The expanded user presentation
            will automatically populate the
            // presentation with the parameter
            values and can be used for
            // display if the rule was defined
            with a template. If no
            // template was defined the
            expanded user presentation
            // is the same as the regular
            presentation.
            out.println("Expanded User
            Presentation: "
                        +
                        rule.getExpandedUserPre
                        sentation());
            // The regular user presentation
            will have placeholders in the
            // string where the parameter can
            be substituted if the rule
            // was defined with a template. If
            the rule was not defined with
            // a template, the user
            presentation will only be a string
            // without placeholders. The
            placeholders are of a format of {n}
            // where n is the index (zero-
            based) of the parameter in the
            // template. This value can be used
            to create an interface for
            // editing where there are fields
            with the parameter values
            // available for editing
            out.println("User Presentation: " +
            rule.getUserPresentation());

            // Check if the rule was defined
            with a template
            if (rule instanceof
            RuleSetTemplateInstanceRule) {
                        RuleSetTemplateInstance
                        Rule templateInstance =
                        (RuleSetTemplateInstanc
                        eRule) rule;
```

```
                                                RuleSetRuleTemplate
                                                template =
                                                templateInstance
                                                .getRuleSetRuleTemplate
                                                ();

                                                List<Parameter>
                                                parameters =
                                                template.getParameters(
                                                );
                                                Iterator<Parameter>
                                                paramIterator =
                                                parameters.iterator();

                                                Parameter parameter =
                                                null;

                                                // Retrieve all of the
                                                parameters and output
                                                the name and value
                                                while
                                                (paramIterator.hasNext(
                                                ))
                                                {
                                                parameter =
                                                paramIterator.next();

                                                out.println("Parameter
                                                Name: " +
                                                parameter.getName());
                                                out.println("Parameter
                                                Value: "
                                                +
                                                templateInstance.getPar
                                                ameterValue(
                                                parameter.getName()).ge
                                                tValue());
                                                }
                                }
                        }
                        out.println("");
                        return out.toString();
                }
}
```

4.3 Additional Query Examples

The following examples are not included with the application containing examples 1-15, however they provide more examples on creating queries to retrieve business rule groups. In these examples different properties and wildcard values (‘_’, ‘%’) are used with different operators (AND, OR, LIKE, NOT_LIKE, EQUAL and NOT_EQUAL).

For the examples, queries will be performed that return between different combinations of 4 business rule groups. It is important to understand the different attributes and properties of the business rule groups as these are used in the queries.
Name: BRG1
Targetnamespace : http://BRG1/com/ibm/br/rulegroup
Properties:
organization, 8JAA
department, claims
ID, 00000567
region: SouthCentralRegion
manager: Joe Bean

Name: BRG2
Targetnamespace : http://BRG2/com/ibm/br/rulegroup
Properties:
organization, 7GAA
department, accounting
ID, 0000047
ID_cert45, ABC
region: NorthRegion

Name: BRG3
Targetnamespace : http://BRG3/com/ibm/br/rulegroup
Properties:
organization, 7FAB
department, finance
ID, 0000053
ID_app45, DEF
region: NorthCentralRegion

Name: BRG4
Targetnamespace : http://BRG4/com/ibm/br/rulegroup
Properties:
organization, 7HAA
department, shipping
ID, 0000023
ID_app45, GHI
region: SouthRegion

4.3.1 Query by a single property

```
List<BusinessRuleGroup> brgList = null;

brgList = BusinessRuleManager.getBRGsBySingleProperty(
"department", QueryOperator.EQUAL,
                                    "accounting", 0, 0);
// Returns BRG2
```

4.3.2 Query business rule groups by properties and wildcard (%) at the beginning and at the end of the value

```
// Query Prop AND Prop
QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode(
                        "region", QueryOperator.LIKE,
                        "%Region");

QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode(
                        "ID", QueryOperator.LIKE,
                        "000005%");

QueryNode queryNode =
QueryNodeFactory.createAndNode(leftNode,
                        rightNode);

brgList =
BusinessRuleManager.getBRGsByProperties(queryNode, 0, 0);

// Returns BRG1 and BRG3
```

4.3.3 Query business rule groups by properties and wildcard (‘_’)

```
brgList = BusinessRuleManager.getBRGsBySingleProperty("ID",
QueryOperator.LIKE, "00000_3", 0, 0);

// Returns BRG3 and BRG4
```

4.3.4 Query business rule group by properties with multiple wildcards (‘_’ and ‘%’)

```
brgList =
BusinessRuleManager.getBRGsBySingleProperty("region",
QueryOperator.LIKE, "__uth%Region",
                        0, 0);

// Returns BRG1 and BRG4
```

4.3.5 Query business rule groups by NOT_LIKE operator and wildcard (‘_’)

```
brgList =
BusinessRuleManager.getBRGsBySingleProperty("organization",
QueryOperator.NOT_LIKE,
                        "7__A", 0, 0);

// Returns BRG1 and BRG3
```

4.3.6 Query business rule groups by NOT_LIKE operator and wildcard (‘_’)

```
brgList =
BusinessRuleManager.getBRGsBySingleProperty("organization",
QueryOperator.NOT_LIKE,
                        "7%", 0, 0);

// Returns BRG1
```

4.3.7 Query business rule groups by NOT_EQUAL operator

```
brgList =
BusinessRuleManager.getBRGsBySingleProperty("department",
QueryOperator.NOT_EQUAL,
                        "claims", 0, 0);

// Returns BRG1
```

4.3.8 Query business rule groups by PropertyIsDefined

```
PropertyIsDefinedQueryNode node =
QueryNodeFactory.createPropertyIsDefinedQueryNode("manager"
);

brgList = BusinessRuleManager.getBRGsByProperties(node, 0,
0);

// Returns BRG1
```

4.3.9 Query business rule groups by NOT PropertyIsDefined

```
// NOT Prop
QueryNode node =
QueryNodeFactory.createPropertyIsDefinedQueryNode("manager"
);

NotNode notNode = QueryNodeFactory.createNotNode(node);

brgList = BusinessRuleManager.getBRGsByProperties(notNode,
0, 0);

// Returns BRG1
```

4.3.10 Query business rule groups by multiple properties with a single NOT node

```
// Prop AND NOT Prop
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("department",
                        QueryOperator.EQUAL, "accounting");

NotNode notNode =
QueryNodeFactory.createNotNode(rightNode);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("ID",
                        QueryOperator.LIKE, "00000%");

AndNode andNode = QueryNodeFactory.createAndNode(leftNode,
                        notNode);

brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);

// Returns BRG2
```

4.3.11 Query business rule groups by multiple properties with multiple NOT nodes combined with AND operator

```
// NOT Prop AND NOT Prop
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("department",
QueryOperator.EQUAL, "accounting");

NotNode notNode =
QueryNodeFactory.createNotNode(rightNode);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("department",
                        QueryOperator.LIKE, "cla%");

NotNode notNode2 =
QueryNodeFactory.createNotNode(leftNode);

AndNode andNode = QueryNodeFactory.createAndNode(notNode,
notNode2);

brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);

// Returns BRG1 and BRG2
```

4.3.12 Query business rule groups by multiple properties with multiple NOT nodes combined with OR operator

```
// NOT Prop OR NOT Prop
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("department",
                        QueryOperator.LIKE, "acc%");

NotNode notNode =
QueryNodeFactory.createNotNode(rightNode);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode(
                        "department", QueryOperator.EQUAL,
                        "claims");

NotNode notNode2 =
QueryNodeFactory.createNotNode(leftNode);

OrNode orNode = QueryNodeFactory.createOrNode(notNode,
notNode2);

brgList = BusinessRuleManager.getBRGsByProperties(orNode,
0, 0);

//Returns BRG1, BRG2, BRG3, and BRG4
```

4.3.13 Query business rule groups by multiple properties combined with multiple AND operators

```
// (Prop AND Prop) AND (Prop AND Prop)
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("department",
                        QueryOperator.LIKE, "acc%");

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("organization",
                        QueryOperator.EQUAL, "7GAA");

AndNode andNodeLeft =
QueryNodeFactory.createAndNode(leftNode,rightNode);

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("ID",
QueryOperator.LIKE,"000004_");

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("region",
QueryOperator.EQUAL,
                        "NorthRegion");

AndNode andNodeRight =
QueryNodeFactory.createAndNode(leftNode2, rightNode2);
```

```
AndNode andNode =
QueryNodeFactory.createAndNode(andNodeLeft,andNodeRight);

brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);

// Returns BRG2
```

4.3.14 Query business rule groups by multiple properties combinded with AND and OR operators

```
// (Prop AND Prop) OR (Prop AND NOT Prop)
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("department",
                        QueryOperator.LIKE, "acc%");

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("organization",
                        QueryOperator.EQUAL, "7GAA");

AndNode andNodeLeft =
QueryNodeFactory.createAndNode(leftNode, rightNode);

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
                        QueryOperator.EQUAL, "8JAA");

NotNode notNode =
QueryNodeFactory.createNotNode(rightNode2);

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("region",
                        QueryOperator.LIKE, "%lRegion");

AndNode andNodeRight =
QueryNodeFactory.createAndNode(leftNode2, notNode);

OrNode orNode = QueryNodeFactory.createOrNode(andNodeLeft,
andNodeRight);

brgList = BusinessRuleManager.getBRGsByProperties(orNode,
0, 0);

// Returns BRG2 and BRG3
```

4.3.15 Query business rule groups by multiple properties combined with AND and NOT operators

```
// Prop AND NOT (Prop AND Prop)
QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("ID",
QueryOperator.LIKE, "000005%");
```

```
QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
                                    QueryOperator.EQUAL,
                                    "8JAA");

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("region",QueryOper
ator.LIKE,
                                    "%lRegion");

AndNode andNodeRight =
QueryNodeFactory.createAndNode(leftNode2, rightNode2);

NotNode notNode =
QueryNodeFactory.createNotNode(andNodeRight);

AndNode andNode = QueryNodeFactory.createAndNode(leftNode,
notNode);

brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);

// Returns BRG3
```

4.3.16 Query business rule groups by multiple properties combined with NOT and OR operators

```
// NOT (Prop AND Prop) OR Prop
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("organization",
                                    QueryOperator.LIKE,
                                    "8_A_");

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
                                    QueryOperator.LIKE,
                                    "7%");

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("region",QueryOper
ator.LIKE,
                                    "%lRegion");

AndNode andNodeRight =
QueryNodeFactory.createAndNode(leftNode2,rightNode2);

NotNode notNode =
QueryNodeFactory.createNotNode(andNodeRight);

OrNode orNode = QueryNodeFactory.createOrNode(notNode,
rightNode);

brgList = BusinessRuleManager.getBRGsByProperties(orNode,
0, 0);
```

```
// Returns BRG3
```

4.3.17 Query business rule groups by multiple properties combined with nested AND operators

```
// Prop AND (Prop AND (Prop AND Prop))
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
                                    QueryOperator.LIKE,
                                    "___thRegion");

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
                                    QueryOperator.LIKE,
                                    "7%");

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("department",
                                    QueryOperator.LIKE,
                                    "%ing");

AndNode andNodeRight =
QueryNodeFactory.createAndNode(leftNode2,rightNode2);

AndNode andNodeLeft =
QueryNodeFactory.createAndNode(rightNode,andNodeRight);

PropertyIsDefinedQueryNode node2 =
QueryNodeFactory.createPropertyIsDefinedQueryNode("ID_cert4
5");

AndNode andNode = QueryNodeFactory.createAndNode(node2,
andNodeLeft);

brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);
// Returns BRG2
```

4.3.18 Query business rule groups by multiple properties combined with nested AND operators

```
// (Prop AND (Prop AND Prop)) AND Prop
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",QueryOper
ator.LIKE,
                                    "___thRegion");

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
                                    QueryOperator.LIKE,
                                    "7%");
```

```
QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("department",
                                    QueryOperator.LIKE,
                                    "%ing");

AndNode andNodeRight =
QueryNodeFactory.createAndNode(leftNode2,rightNode2);

AndNode andNodeLeft =
QueryNodeFactory.createAndNode(rightNode,andNodeRight);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("ID_app45",QueryOp
erator.LIKE, "GH_");

AndNode andNode =
QueryNodeFactory.createAndNode(andNodeLeft, leftNode);

brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);

// Returns BRG4
```

4.3.19 Query business rule groups by multiple properties combined with nested AND operators and a NOT node

```
// Prop AND (Prop AND (Prop AND NOT Prop))
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("organization",
                                    QueryOperator.LIKE,
                                    "7%");

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("region",
                                    QueryOperator.LIKE,
                                    "%lRegion");

NotNode notNode =
QueryNodeFactory.createNotNode(rightNode2);

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("department",
                                    QueryOperator.LIKE,
                                    "%ing");

AndNode andNodeRight =
QueryNodeFactory.createAndNode(leftNode2,notNode);

AndNode andNodeLeft =
QueryNodeFactory.createAndNode(rightNode,andNodeRight);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("ID_cert45",
                                    QueryOperator.LIKE,
                                    "AB_");
```

```
AndNode andNode = QueryNodeFactory.createAndNode(leftNode,
andNodeLeft);

brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);

// Returns BRG2
```

4.3.20 Query business rule groups by multiple properties combined with nested AND operators

```
// (Prop AND (Prop AND Prop)) AND Prop - Return empty
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
                                    QueryOperator.LIKE,
                                    "___thRegion");

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
                                    QueryOperator.LIKE,
                                    "7%");

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("department",
                                    QueryOperator.LIKE,
                                    "%ing");

AndNode andNodeRight =
QueryNodeFactory.createAndNode(leftNode2,rightNode2);

AndNode andNodeLeft =
QueryNodeFactory.createAndNode(rightNode,andNodeRight);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("ID_cert45",
                                    QueryOperator.LIKE,
                                    "GH_");

AndNode andNode =
QueryNodeFactory.createAndNode(andNodeLeft, leftNode);

brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);

//Returns no BRGs
```

4.3.21 Query business rule groups by multiple properties combined with nested OR operators

```
// (Prop OR (Prop OR Prop)) OR Prop
```

```
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
                                    QueryOperator.LIKE,
                                    "___thRegion");

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
                                    QueryOperator.LIKE,
                                    "7%");

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("department",
                                    QueryOperator.LIKE,
                                    "%ing");

OrNode orNodeRight =
QueryNodeFactory.createOrNode(leftNode2,rightNode2);

OrNode orNodeLeft =
QueryNodeFactory.createOrNode(rightNode,orNodeRight);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("ID_cert45",
                                    QueryOperator.LIKE,
                                    "GH_");

OrNode orNode = QueryNodeFactory.createOrNode(orNodeLeft,
leftNode);

brgList = BusinessRuleManager.getBRGsByProperties(orNode,
0, 0);

// Returns BRG1
```

4.3.22 Query business rule groups by multiple properties combined with nested OR operators

```
// (Prop OR (Prop OR NOT Prop)) OR Prop
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
                                    QueryOperator.LIKE,
                                    "___thRegion");

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
                                    QueryOperator.LIKE,
                                    "7%");

NotNode notNode =
QueryNodeFactory.createNotNode(rightNode2);

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("department",
                                    QueryOperator.LIKE,
                                    "%ing");
```

```
OrNode orNodeRight =
QueryNodeFactory.createOrNode(leftNode2,notNode);

OrNode orNodeLeft =
QueryNodeFactory.createOrNode(rightNode,orNodeRight);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("ID_cert45",
                                    QueryOperator.LIKE,
                                    "GH_");

OrNode orNode = QueryNodeFactory.createOrNode(orNodeLeft,
leftNode);

brgList = BusinessRuleManager.getBRGsByProperties(orNode,
0, 0);

// Returns BRG3
```

4.3.23 Query business rule groups by multiple properties combined with nested OR operators and a NOT node

```
// Prop OR NOT(Prop OR Prop)
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
                                    QueryOperator.LIKE,
                                    "___thRegion");

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode(
                                    "organization",
                                    QueryOperator.LIKE,
                                    "7%");

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode(
                                    "department",
                                    QueryOperator.LIKE,
                                    "%ing");

OrNode orNodeRight =
QueryNodeFactory.createOrNode(rightNode2,
                                    rightNode);

NotNode notNode =
QueryNodeFactory.createNotNode(orNodeRight);

OrNode orNodeLeft = QueryNodeFactory.createOrNode(leftNode,
notNode);

brgList =
BusinessRuleManager.getBRGsByProperties(orNodeLeft, 0, 0);

// Returns BRG3
```

4.3.24 Query business rule groups by multiple properties combined with nested OR operators and a NOT node

```
//  NOT(Prop OR Prop) OR Prop
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
                                    QueryOperator.LIKE,
                                    "%lRegion");

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode(
                                    "organization",
                                    QueryOperator.LIKE,
                                    "7%");

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode(
                                    "department",
                                    QueryOperator.LIKE,
                                    "%ing");

OrNode orNodeRight =
QueryNodeFactory.createOrNode(rightNode2,rightNode);

NotNode notNode =
QueryNodeFactory.createNotNode(orNodeRight);

OrNode orNodeLeft =
QueryNodeFactory.createOrNode(notNode,leftNode);

brgList =
BusinessRuleManager.getBRGsByProperties(orNodeLeft, 0, 0);


// Returns BRG2 and BRG4
```

4.3.25 Query business rule groups by a list of nodes that are combined with an AND operator

```
//  AND list
List<QueryNode> list = new ArrayList<QueryNode>();


QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
                                    QueryOperator.LIKE,
                                    "%thRegion");
list.add(rightNode);
```

```
QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
                                    QueryOperator.LIKE,
                                    "7%");
list.add(rightNode2);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("department",
                                    QueryOperator.LIKE,
                                    "%ing");
list.add(leftNode);

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
                                    QueryOperator.LIKE,
                                    "7H%");
list.add(leftNode2);

AndNode andNode = QueryNodeFactory.createAndNode(list);

brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);

// Returns BRG4
```

4.3.26 Query business rule groups by a list of nodes and NOT node combined with an AND operator

```
//  AND list with a notNode
List<QueryNode> list = new ArrayList<QueryNode>();

QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
                                    QueryOperator.LIKE,
                                    "%thRegion");
list.add(rightNode);

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
                                    QueryOperator.LIKE,
                                    "8%");
NotNode notNode =
QueryNodeFactory.createNotNode(rightNode2);

list.add(notNode);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("department",
                                    QueryOperator.LIKE,
                                    "%ing");
list.add(leftNode);

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
                                    QueryOperator.LIKE,
                                    "7H%");
list.add(leftNode2);

AndNode andNode = QueryNodeFactory.createAndNode(list);

brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);

// Return BRG4
```

4.3.27 Query business rule groups by a list of nodes that are combined with an OR operator

```
//  OR list
List<QueryNode> list = new ArrayList<QueryNode>();

QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
                                    QueryOperator.LIKE,
                                    "%thRegion");
list.add(rightNode);

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
                                    QueryOperator.LIKE,
                                    "8%");


list.add(rightNode2);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("department",
                                    QueryOperator.LIKE,
                                    "%ing");
list.add(leftNode);

OrNode orNode = QueryNodeFactory.createOrNode(list);

brgList = BusinessRuleManager.getBRGsByProperties(orNode,
0, 0);

//Returns BRG3
```

4.3.28 Query business rule groups by a list of nodes and Not node combined with an OR operator

```
// OR list with Not node
List<QueryNode> list = new ArrayList<QueryNode>();

QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
                                    QueryOperator.LIKE,
                                    "%thRegion");
list.add(rightNode);

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
                                    QueryOperator.LIKE,
                                    "8%");

NotNode notNode =
QueryNodeFactory.createNotNode(rightNode2);

list.add(notNode);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("department",
                                    QueryOperator.LIKE,
                                    "%ing");
list.add(leftNode);

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
                                    QueryOperator.LIKE,
                                    "8%");
list.add(leftNode2);

OrNode orNode = QueryNodeFactory.createOrNode(list);

brgList = BusinessRuleManager.getBRGsByProperties(orNode,
0, 0);

//Returns BRG1, BRG2, BRG3, and BRG4
```