Name: Business Rules Best Practice

<INTERNAL>

By: ChangeManagement-dev

Change History:

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Description** |
| 03/23/2012 | Nathan Kral | Initial draft |
| 04/12/2012 | Michael McLachlan | Incorporating team review comments |

</INTERNAL>

<INTERNAL>

# Best Practice Name and Classification

## Name

*Business Rules* Best Practice

## Classifications

Core Design Concept

## Also Known As

None

</INTERNAL>

# Objective

## Problem Statement

Provide the ability to create business rules to provide custom validation logic for objects in PDMLink.  These rules should be used to determine eligibility of an object to go through defined checkpoints in the change management or promotion processes. The user will require feedback on the result of the evaluation. The business rules validation should be scalable and configurable.

## Background

Business Rules offer a validation framework which can be utilized within workflow execution. For example, before objects are released through the Change Notice workflow, we must ensure that none of the Resulting Objects are checked out. We can make a Business Rule to perform this validation.

## Scope/Applicability/Assumptions

This document is heavily focused on the Business Rules Engine itself. It also provides a few working examples.

## Intended Outcome

After going through this document, you should be able to start using the rules that are provided with Windchill 10.1 M010, or you should be able to create custom business rules for your change process.

# Solution

## Solution Statement

Solution: Use business rule sets and rules to validate that the objects are valid to be processed.

## Prerequisite knowledge

To apply this best practice, you need to have an understanding of the following:

* Basic development involving Java and XML
* The management of Windchill properties

[*Load Business Rule Objects* Best Practice](http://ah-grok/xref/x-22-M010/wcmod/modules/BusinessRules/src/doc-files/wt/businessRules/LoadBusinessObjectsBestPractice.doc)

## Solution Elements

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Type** | **Package** | **Description** |
| BusinessRuleSet | Java class | wt.businessRules | A BusinessRuleSet instance is the configuration used to navigate a set of business rules for a container. |
| BusinessRule | Java class | wt.businessRules | BusinessRule will hold the information about the rule to be executed. Each rule may have unique configuration requirements associated to it. |
| BusinessRuleLink | Java class | wt.businessRules | An ObjectToObject binary link holding the references to BusinessRuleSet and BusinessRule. Each BusinessRuleSet is associated to BusinessRules using BusinessRuleLinks. |
| BusinessRulesHelper | Java Class | wt.businessRules | Helper for the business rules service and engine. |
| BusinessRulesEngine | Java Class | com.ptc.core.businessRules.engine | The business rules engine will execute the validation of each business rule in order as defined by the execution plan set in the business rule set. |
| BusinessRuleSetBean | Java Class | com.ptc.core.businessRules.engine | Used to specify more than one business rule set and target collection relationship key at a time in order to generate an accumulation of results with one call to the business rule engine. |
| RuleValidationCriteria | Java Class | com.ptc.core.businessRules.validation | Contains information that is used by all business rules within a business rule set. |
| RuleValidation | Java Class | com.ptc.core.businessRules.validation | Interface for the business rule validators. Custom validators should implement this interface. |
| RuleValidationResult | Java Class | com.ptc.core.businessRules.validation | Contains one validation output per business rule. It contains a rule validation key to indicate the business rule that was executed. It will hold a validation status attribute to indicate if the business rule successfully passed the validation or not. The RuleValidationResult will also contain the target object, which links the result to a specific WTReference, and a list of feedback messages which contains text that could optionally be displayed in a report. |
| RuleValidationResultSet | Java Class | com.ptc.core.businessRules.validation | A set of RuleValidationResult objects. |
| BusinessRuleSetRelationshipDelegate | Java Class | com.ptc.core.businessRules.relationship | Used to get the seed objects for a primary business object that is to be evaluated for a particular business rule set. Relationship delegates are registered in service.properties. See section 3.4.5 of this document for more information on these delegates. |

## Create Business Rules

See [*Load Business Rule Objects* Best Practice](http://ah-grok/xref/x-22-M010/wcmod/modules/BusinessRules/src/doc-files/wt/businessRules/LoadBusinessObjectsBestPractice.doc) for details on how create, modify and remove business rule sets, business rules, and business rule links.

## Business Rules Configuration and Execution

The business rules engine will execute the registered rules in a business rule set and return validation results. The business rules engine will retrieve the set of business rules given the specified business rule set unique keys and the container references from the seed objects. With the set of business rules, the engine will execute them against the seed objects.

The following are the phases that describe the execution of business rule sets by the business rules engine:

1. Retrieval of the specified seed objects to be validated
2. Retrieval of the business rule sets
3. Generating and executing an execution plan for the business rule sets
4. Reporting of the results of the plan executed.

### Business Rules Execution Flow

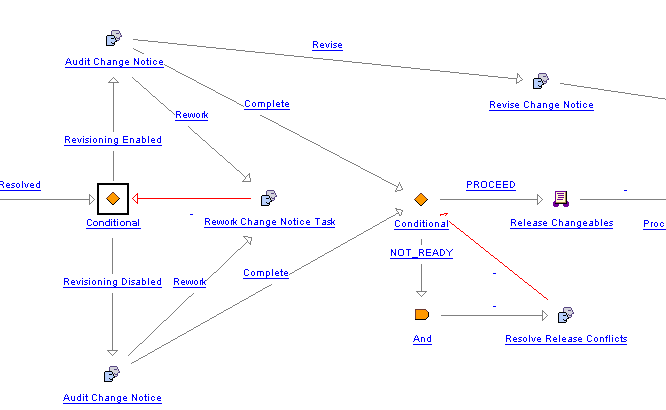


### Calling the Business Rules Engine

There are examples in the default Change Notice workflow and Promotion Request workflows that show how to call the business rules engine from a workflow. The default Change Notice workflow comes with two different conditionals which call the business rules engine. If there are any failures, the Audit Change Notice Task will show a link called “View Conflicts” on the task. The Audit Change Notice Task allows you to proceed without fixing the failures. Similarly, the Resolve Release Conflicts Task shows the conflicts if there are any, but you must fix the problems before proceeding on this task.

Conditional invokes Business Rules

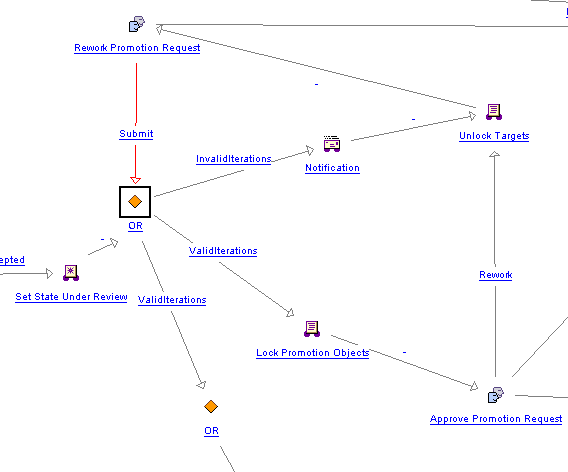
Shows rule conflicts and allows you to proceed

****

Shows rule conflicts that must be corrected before you can proceed

Conditional invokes Business Rules

Here is another example of how you might want to set up a rework loop in the Promotion Request Approval Process. The Business rule is executed in the OR conditional. If it fails, then an email is sent out and the Rework Promotion Request task appears:



There are two supported APIs to execute business rule sets.

BusinessRulesHelper.engine.execute(primaryBusinessObject, BusinessRuleSetBean[]) : RuleValidationResultSet

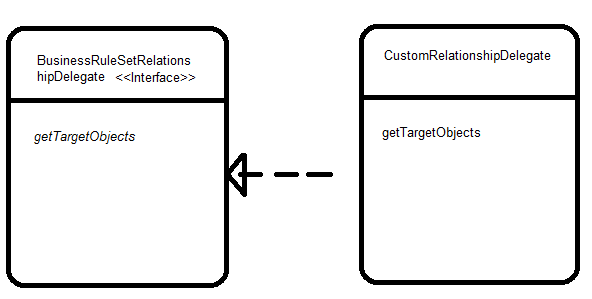
The second option allows for overriding the container for all objects regardless of the container that the seed objects are in. This API requires that the seed objects be specified up front and only supports executing one business rule set at a time.

BusinessRulesHelper.engine.execute(“Business rule set”, RuleValidationCriteria) : RuleValidationResultSet

See Javadoc for more details.

### Retrieval of Seed Objects

The relationship delegate is used to get the seed objects for a primary business object that is to be evaluated for a particular business rule set. All relationship delegates implement the RuleValidation interface.

  
  
Relationship delegates are registered in *service.properties*. The following is an example of a delegate for getting the resulting objects for a primary object:

<Service context="default" name="com.ptc.core.businessRules.relationship.BusinessRuleSetRelationshipDelegate">

<Option serviceClass="com.ptc.core.businessRules.relationship.ResultingObjectsRelationshipDelegate"

selector="wt.change2.ChangeRecord2"

cardinality="singleton"

requestor="null"/>

The selector is the relationship key which is used to look up the relationship delegate instance. The relationship key naming convention is to use the link class name but is not a requirement. Note that if the relationship delegate returns the links for the relationship, then the role B objects of the links will be used as the seed objects. The target link role can be optionally specified in the *BusinessRuleSetBean* or *RuleValidationCriteria*. See *BusinessRuleSetBean* for details on how to specify the relationship for a business rule set and current supported relationship delegates.

The following table shows the default delegates that are provided in a standard Windchill installation:

|  |  |  |
| --- | --- | --- |
| Relationship Key | Relationship Delegate | Description |
| wt.change2.AffectedActivityData | com.ptc.core.businessRules.relationship.AffectedObjectsRelationshipDelegate | Gets the affected object links of the primary business object if it is a change notice or change task. |
| wt.change2.ReportedAgainst | com.ptc.core.businessRules.relationship.AffectedObjectsRelationshipDelegate | Gets the affected object links of the primary business object if it is a problem report or variance. |
| wt.change2.RelevantRequestData | com.ptc.core.businessRules.relationship.AffectedObjectsRelationshipDelegate | Gets the affected object links of the primary business object if it is a change request |
| wt.change2.ChangeRecord2 | com.ptc.core.businessRules.relationship.ResultingObjectsRelationshipDelegate | Gets the resulting object links of the primary business object if it is a change notice or change task. |
| wt.change2.IncludedIn2 | com.ptc.core.businessRules.relationship.IncludedInRelationshipDelegate | Gets the change tasks of the primary business object if it is a change notice or the change notice if the primary business object is a change task. |
| wt.maturity.MaturityBaseline | com.ptc.core.businessRules.relationship.MaturityBaselineObjectsRelationshipDelegate | Gets the baseline items of the primary business object if it is a promotion notice. |
| wt.maturity.PromotionSeed | com.ptc.core.businessRules.relationship.PromotionSeedObjectsRelationshipDelegate | Gets the promotion seeds of the primary business object if it is a promotion notice. |
| wt.maturity.PromotionTarget | com.ptc.core.businessRules.relationship.PromotionTargetObjectsRelationshipDelegate | Gets the promotion targets of the primary business object if it is a promotion notice. |

### Retrieval of the Business Rule Set Based on Container

The following information is required to obtain a business rule set from the system:

1. The unique key for a business rule set
2. The container reference of which the rules should be evaluated

The business rules engine will be responsible for acquiring the appropriate business rule set from the Windchill services. The business rule sets will be discovered using a hierarchical container lookup. Business rule sets that are not enabled will be ignored.

The following flow chart explains flow of the container look up for a rule set key and a container.



### Execute the Business Rule Set Execution Plan

The generation of the execution plan of a business rule set will involve:

1. Creation of a specific *RuleValidation* instance based on the selector defined in the business rule.
2. The *RuleValidation* instance will be used to determine if the configuration of the business rule is valid using the *isConfigurationValid* method. If the configuration is not valid, the *RuleValidation* will be skipped for the business rule set execution plan and error will be logged.
3. For each business rule the *RuleValidation* instance is called with the following items:
   1. Collection of objects that rules are to evaluate
   2. Context specific criteria (such as the container reference) that may be used by a *RuleValidation* to configure its execution.
4. The *RuleValidation* instance is added to the execution plan.

All the business rule validators will implement the *RuleValidation* interface. A typical business rule will fall under two primary categories:

* **Simple validation:** The rules check is done on the business object itself or on the seed objects as configured by the relationship. An example could be checking for attribute validation or checking a life cycle state of the object prior to release.
* **Structured validation:** The rules may or may not be performed on the business objects but will include some validation checks on its associated objects like the children of the part or document. In this type of validation, typically more complex set of algorithm might be required to calculate the business validation rules and populate the validation result with appropriate flag. An example of structured validation rule is a validation check that makes sure that all resulting objects and all first level dependents are in the correct state.

The following are supported RuleValidation implementations that are provided with the default Windchill installation:

|  |  |  |
| --- | --- | --- |
| Business Rule Selector | RuleValidation | Description |
| CHECKOUT\_RULE | com.ptc.core.businessRules.validation.CheckoutRuleValidator | Validates that objects are not checked out or checked out to a project. |
| RELEASE\_TARGET | com.ptc.core.businessRules.validation.ReleaseTargetValidator | Ensures that all “Resulting Objects” are at an appropriate state for release and have an appropriate change management transition specified that is consistent with the assigned lifecycle transition rules. |
| ATTRIBUTE\_RULE | com.ptc.core.businessRules.validation.AttributeRuleValidator | Validates that the specified attribute rule configuration for an object type is valid. |
| BOM\_RELEASE\_RULE | com.ptc.core.businessRules.validation.BOMReleaseRuleValidator | Ensures that all resulting objects are at an appropriate state and all their first level children are at an appropriate state. |

Note: as of the 11.0 release, the Promotion Request supports the Attribute Rule and the Check Out rule. The Change Notice supports any rule.

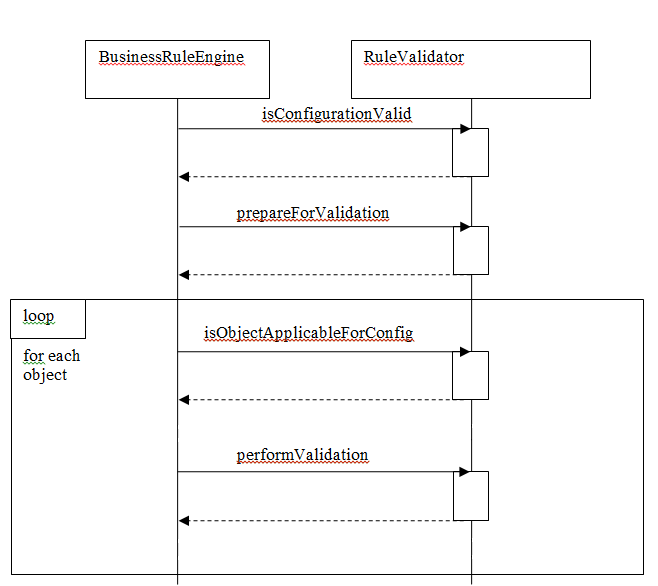
The instantiation of the [*RuleValidation*](http://wc-javadoc/javadoc/Windchill_x-22-M010/docs/api_14/wnc/BusinessRules/index.html) will be based on the selector defined in the business rule. Application services will be used to lookup the implementing class for the defined selector. If no implementing class is defined, or the class fails to be instantiated an error will be logged.

Prior to executing any validation the engine will call the *RuleValidation isConfigurationValid* which will determine if the business rule configuration is valid. This validation of the business rule configuration is also called when loading the business rule (see section 3.4), but since it is possible to change the registered *RuleValidation,* validation of configuration is also called prior to executing the business rule validation.

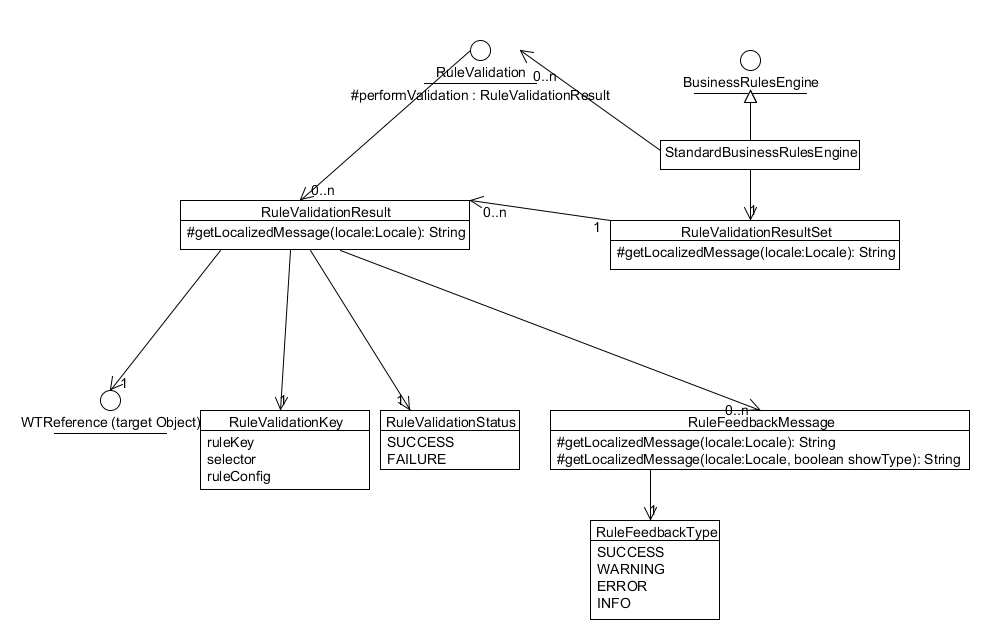
The *RuleValidation* initialization method called *prepare* is used to perform some initial preprocessing of the entire set of seed objects that will be validated against the set of business rules. This is to allow any pre-validation to occur within the Windchill server (example might be retrieving the lifecycle template which would be common to many of the items). It is best to call multi-object APIs from the *RuleValidation.prepare* method. It is advised not to use the *RuleValidation.peformValidation* method to perform multi-object calls.

If the method *RuleValidation.isConfigurationValid* returns true,the *RuleValidation* *performValidation* is called for each seed object. The seed object is passed inside the *RuleValidationObject*. The *RuleValidationObject* holds the target object (the seed object if it is not a link or the role A or role B if seed object is a link) and the link (if the seed object is a link). When the business rule is a structured type *RuleValidation*.the *RuleValidationObject* also contains the collected children and a map of the collected children to the links that they were collected with. The performValidation should always return a new *RuleValidationResult* that has a reference to the target object and not the link.

##### Business Rules Engine and Sequence Diagram



##### Business Rules Engine and Validation Interaction Diagram



*RuleValidation* implementations are registered in service.properties. To create or override a *RuleValidation* implementation, add the following to a custom service.properties xconf file:

<Service context="default" name="com.ptc.core.businessRules.validation.RuleValidation">

<Option serviceClass="com.ptc.core.businessRules.validation.CheckoutRuleValidator"

selector="CHECKOUT\_RULE"

cardinality="singleton"

requestor="null"/>

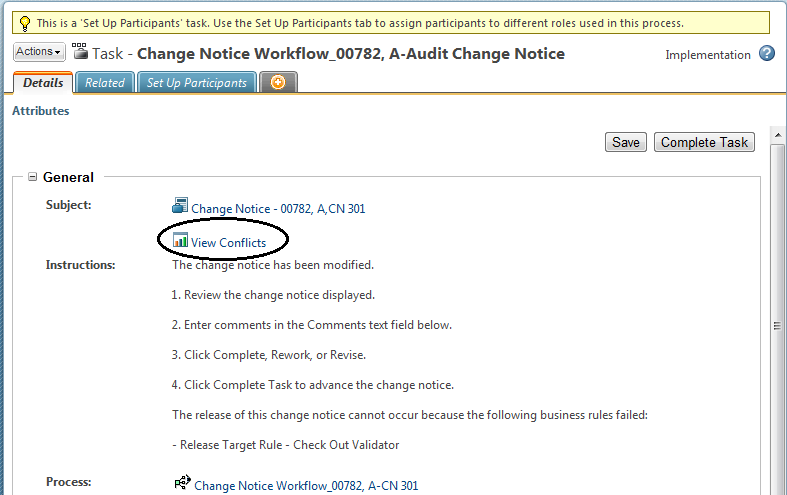
</Service>

When possible the cardinality of the *RuleValidation* should be set to “singleton” in order to reduce the number of instances of a particular RuleValidation implementation. The [*RuleValidationKey*](http://wc-javadoc/javadoc/Windchill_x-22-M010/docs/api_14/wnc/BusinessRules/index.html)*.processingMap* can be used for storing pre-validation data in order to support thread safe processing of *RuleValidation* implementations when the cardinality is set to “singleton”.

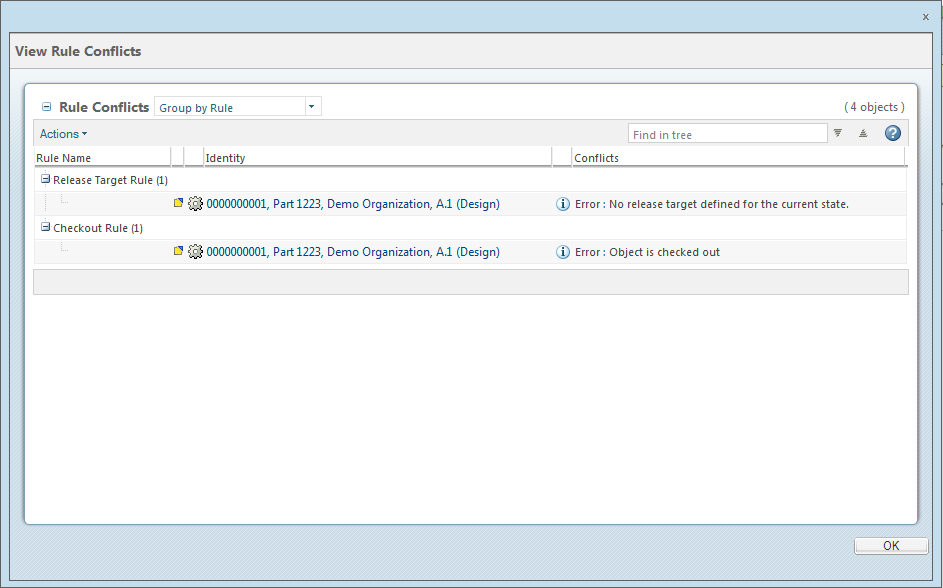
### Returning Results

The single validation result is generated for each target object that is evaluated is combined into an aggregated set of validation results (*RuleValidationResultSet)*. The *RuleValidationResultSet* may contain only partial results if there are failures during the execution of the business rule set. The business rules engine may be invoked with an optional parameter indicating that the execution plan should abort further rule validation upon hitting a failure in a rule. Since the failure could occur in different rules for different objects, the execution plan will continue to the next seed object if the failure on error flag is set. The caller of the business rules engine must determine which mode the execution plan so execute in.

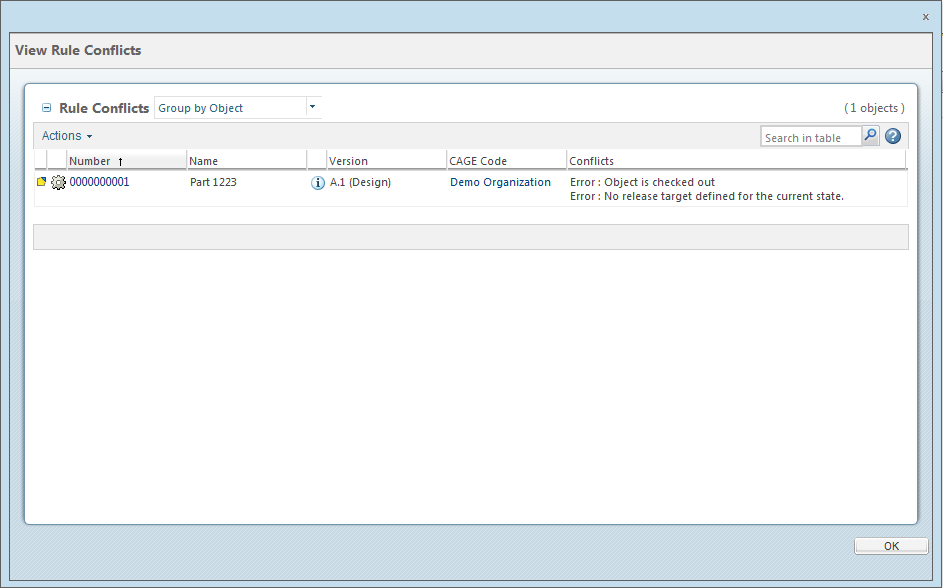
To show the results of the business rule sets on a workflow task, you need to create a workflow variable named *businessRulesResultSetGlobal* and set the type to serialized string of the validation result set. Then, for any task that you want to show the results in, you should create a new workflow variable on that task named *businessRulesResultSet*. On the InitializeFrom field, you should set the value to be *businessRulesResultSetGlobal*, and the copy into should be *businessRulesResultSet*. This copies the global value into the current task value so that history can be maintained on the workflow task results. Use *BusinessRulesHelper.serialize* to create a serialized string of the validation result set. When the *businessRulesResultSetGlobal* is set a “View Conflicts” report link icon will display as part of the subject in the workflow task. To see the report of the evaluated business rule sets click on the “View Conflicts”.



Audit Change Notice with View Conflicts



Business Rule Set Validation Report (Group by Rule view)



Business Rule Set Validation Report (Group by Object view)

# Limitations

The execution of business rules in 10.1 M10 is a single threaded operation.

# Sample Code

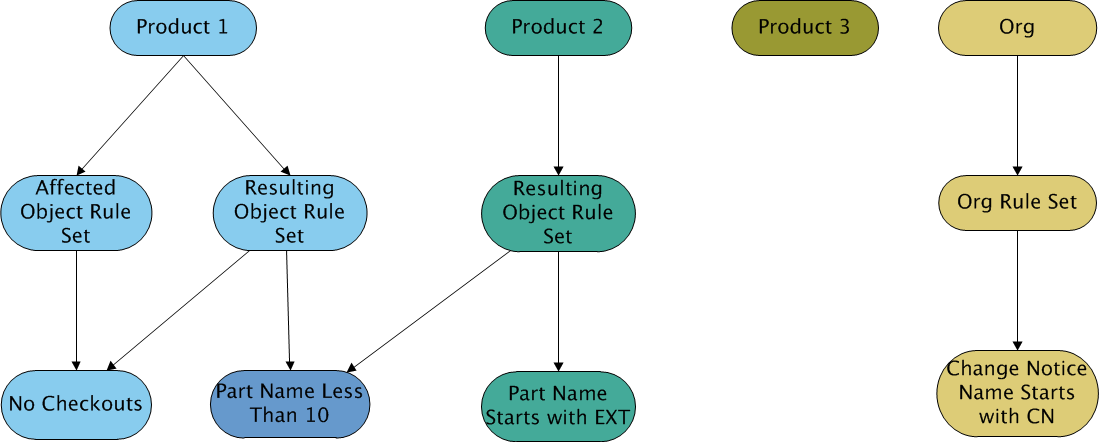
The following is an example of the workflow conditional syntax in the delivered Change Notice workflow template. The workflow conditional makes a call to the business rules engine for the CHANGEABLE\_PRE\_RELEASE business rule set. The CHANGEABLE\_PRE\_RELEASE business rule contains the CHECKOUT\_RULE and RELEASE\_TARGET business rules.

|  |
| --- |
| Routing Expression: |
| result = "NOT\_READY";  com.ptc.core.businessRules.engine.BusinessRuleSetBean[] ruleSetBeans = new com.ptc.core.businessRules.engine.BusinessRuleSetBean[] {  com.ptc.core.businessRules.engine.BusinessRuleSetBean.newBusinessRuleSetBean("CHANGEABLE\_PRE\_RELEASE", "wt.change2.ChangeRecord2")  };  //Invoke the business rule engine  com.ptc.core.businessRules.validation.RuleValidationResultSet resultSet = wt.businessRules.BusinessRulesHelper.engine.execute(primaryBusinessObject, ruleSetBeans);  if ( !resultSet.hasResultsByStatus(com.ptc.core.businessRules.validation.RuleValidationStatus.FAILURE)) {  result = "PROCEED";  }  else {  //This line of code will start exception handling for the business rules that failed  businessRulesResultSetGlobal = wt.businessRules.BusinessRulesHelper.serialize(resultSet);  preReleaseConflictsMsg = new wt.util.WTMessage("com.ptc.windchill.enterprise.change2.change2ClientResource", com.ptc.windchill.enterprise.change2.change2ClientResource.BUSINESS\_RULES\_PRERELEASE\_VALIDATION\_MSG, null).getLocalizedMessage();  preReleaseConflictsMsg = preReleaseConflictsMsg + "\n" + resultSet.getFailedRulesMessage(java.util.Locale.getDefault());  } |

Note that if the validation results are not successful, the attribute *businessRulesResultSetGlobal* is set to the serialized string of the validation result set. The “View Conflicts” report link will show up on the “Audit Change Notice” and “Resolve Release Conflicts” workflow task pages.

# Customization Example of Multiple Rules, Rule Sets, and Relationships

Here is another example of how you might configure rules against multiple products and multiple relationships. In this case, there are 4 rules, 4 rule sets and 4 containers.



After all the rules are created, you could configure the workflow to use these rule sets by using some code that looks like this:

com.ptc.core.businessRules.engine.BusinessRuleSetBean[] ruleSetBeans = new com.ptc.core.businessRules.engine.BusinessRuleSetBean[] {

com.ptc.core.businessRules.engine.BusinessRuleSetBean.newBusinessRuleSetBean("CNOrgRuleSet", com.ptc.core.businessRules.engine.BusinessRuleSetBean.PRIMARY\_BUSINESS\_OBJECT),

com.ptc.core.businessRules.engine.BusinessRuleSetBean.newBusinessRuleSetBean("AffectedPartProdRuleSet", wt.change2.AffectedActivityData.class.getName()),

com.ptc.core.businessRules.engine.BusinessRuleSetBean.newBusinessRuleSetBean("ResultingPartProdRuleSet", wt.change2.ChangeRecord2.class.getName())

};

com.ptc.core.businessRules.validation.RuleValidationResultSet resultSet = wt.businessRules.BusinessRulesHelper.engine.execute(primaryBusinessObject, ruleSetBeans );

Examples:

If we had a part in Product 1 that was a Resulting Object, the part would have to have a part name less than 10 characters, and it could not be checked out.

If we had a document in Product 2 that was a resulting object, then there are no rules for it. This is because there are only rules for Parts.

# Troubleshooting Configuration Issues

In order to troubleshoot configuration issues, we recommend to enable the following loggers on the Method Server. To enable the logging, navigate to \Windchill\codebase\WEB-INF, open log4jMethodServer.properties with a text editor, add the following lines, and reboot your method server:

To view potential configuration issues, such as rules or rule sets not being found in a container, enable this logger:

log4j.logger.com.ptc.core.businessRules.engine=WARN

To view potential performance issues, such as execution time of a business rule validation, enable this logger:

log4j.logger.com.ptc.core.businessRules.engine=TRACE

To view what rules are invoked by the business rules framework, set the level to:

log4j.logger.com.ptc.core.businessRules.engine=INFO

To set the debug mode back to default, set the following logger, or just remove what was previously added:

log4j.logger.com.ptc.core.businessRules.engine=ERROR

# See Also

<INTERNAL>

## Related Best Practices

* [*Load Business Rule Objects* Best Practice](http://ah-grok/xref/x-22-M010/wcmod/modules/BusinessRules/src/doc-files/wt/businessRules/LoadBusinessObjectsBestPractice.doc)

</INTERNAL>