1

Business Rules

A business rule defines or constrains conditions that must hold true in specified situations. Business rules provide custom validation logic for objects in a product or library. The rules determine if an object is eligible to go through defined checkpoints in the change management or promotion process and provide results of the evaluation as feedback to the user.

The goal of business rules is to provide the ability to automate the validation and feedback of conditions that must be met throughout a process. This can be checking to see that data meets requirements of the business during a change process; for example, the attributes of a part are acceptable for release, or the children of a CAD document are at an appropriate state to release the parent. The business rules framework provides the ability to define and execute the rules through configuration and provide process feedback. The end result is reduction of manual work to validate business data and errors in the enterprise due to incorrect information.

Business Rule Objects

There are three business rule objects:

- BusinessRuleSet—the persistence configuration used to navigate the persistent set of validation business rules for a context. This can be overridden at the context level.
- BusinessRule—holds the information about the rule to be executed. Each rule may have unique configuration requirements associated with it. The selector is mapped to the actual implemented rule validation class.
- BusinessRuleLink—the object-to-object binary link holding the references to BusinessRuleSet and BusinessRule. Each BusinessRuleSet is associated with a BusinessRule using the BusinessRuleLink.

The following table shows the attributes of the business rule objects:

BusinessRuleSet	uniqueKey, name, description, enabled, overridable
BusinessRule	uniqueKey, name, description, configs, enabled
BusinessRuleLink	blockNumber

Attribute	Required?	Description	Type
uniqueKey	Yes	Unique key for a given context.	String—up to 256 characters
name	Yes	Localized resource key to describe the name of the object.	String—up to 256 characters
description	No	Localized brief description of the business rule object. The description field is localized.	String—up to 4000 character
configs	No	Contains the information that can be used to configure variations for a give business rule. You can optionally define one or more config elements in the configs attribute. Each configs attribute can hold multiple configurations of name and value pairs.	Name value pair
enabled	No	Indicates the ability to configure the business rule object to be executed. When the value is false, the business rule object is ignored and the next available enabled object in the context hierarchical lookup is used.	boolean
overridable	No	Indicates if a lower-level context business rule instance can override	boolean

Attribute	Required?	Description	Type
		one defined at a higher level. If false, the business rule instance of the top-level context is used.	
blockNumber	No	Indicates the execution order of the rules. You can configure multiple rules to the same blockNumber, but there is no guarantee of order. A value of 1 indicates the first rule to be executed within the business rule set.	Numeric integer value

Available Business Rules

The available business rules validate the following:

- Checkout Rule—Objects are not checked out or checked out to a project.
- Attribute Rule—Specified attributes meet certain conditions
- Release Target Rule—All resulting objects are at an appropriate state for release and have an appropriate change management transition specified that is consistent with the assigned life cycle transition rules
- BOM Release Rule—All resulting objects are at an appropriate state and all their first-level dependants are at an appropriate state

These rules can be extended or overridden by registering a new rule validation class for the rule selector in service.properties. For more information about registering a new rule validation class, see the Windchill Customizer's Guide.

If the business rule fails, you can view the cause by clicking **View Conflicts** on the **Audit Change Notice** and the **Resolve Release Conflicts** task pages.

Checkout Rule

The **Checkout** rule validates that objects are not checked out or checked out to a project.

Attribute Rule

The attribute rule validates that specified attributes meet certain conditions. The object type is the only required input for an attribute rule configuration. The object type should be a wt.fc.Persistable type object. For example, com.ptc.Waiver is a subtype of wt.change2.WTVariance.

```
<configs>
```

```
     <config name="objectType" value="com.ptc.Waiver"></config>
</configs>
```

You can specify the type name or the type string value by using only the type external name (com.ptc.Waiver or WCTYPE|wt.change2.WTVariance|com.ptc.Waiver)

You can exclude subtypes of the specified object type from the attribute rule. For example, the following attribute rule configuration for object type WTVariance would exclude com.ptc.Waiver and com.ptc.Deviation type variances from being evaluated.

```
<configs>
     <config name="objectType" value="wt.change2.WTVariance "></config>
     <config name="excludedType" value="com.ptc.Waiver"></config>
     <config name="excludedType" value=" com.ptc.Deviation "></config>
</configs>
```

To configure a condition to validate on an attribute of the specified object type, the configuration name is the attribute name and the value is the condition. The following attribute configuration checks if a com.ptc.Waiver has an approved quantity between 10 and 100.

```
<configs>
     <config name="objectType" value="com.ptc.Waiver"></config>
     <config name="approvedQuantity" value="[10..100]"></config>
     </configs>
```

The following is an example where weight is the internal name of a global attribute on some subtype of part.

```
<configs>
     <config name="objectType" value="com.ptc.SubTypePart"></config>
     <config name="weight" value="<=100"></config>
</configs>
```

The following table shows the supported operations and object types supported in attribute rule configuration.

Operator	Value Types	Description	Example Load Conditions
>, >=, <, <=	Long, String, Date, Time-stamp, Floa-tingPoint, FloatingPoint-WithUnits	Validates that the object attribute value is greater than, greater than or equal to, less than, or less than or equal to the value specified. The operator characters are reserved characters in the import load; the less than operator (<) needs to	>2012-01-12 00:00:00 =<=20.002 >=10

Operator	Value Types	Description	Example Load Conditions
		be specified as ‘<’ and greater than operator (>) as ‘>’.	
[0..10]	Long, String, Date, Timestamp, FloatingPoint, FloatingPointWithUnits	Validates that the object attribute value is within the specified range. If no minimum value in the range is specified, the condition is equivalent to less than or equal to. If the maximum value in the range is not specified, the condition is treated a greater than or equal to.	[-10.09..10.09] [2012-01-12 00:00:00..2012-02-12 00:00:00] [..100] [0..]
=, !=	Boolean, Long, String, Date, Timestamp, FloatingPoint, FloatingPointWithUnits	Validates that the object attribute value is or is not the value specified. The operator character ! is a reserved character in the import load; it must be specified as ‘#33;’	=true #33;=2012-02-12 00:00:00 =10.09
[A,B]	Boolean, Long, String, Date, Timestamp, FloatingPoint, FloatingPointWithUnits	Validates that the object attribute value is within the discrete set.	[true] [2012-02-01 00:00:00,2012-02-15 00:00:00,2012-02-30 00:00:00] [MAJOR, CRITICAL] [10.01,10.02] [10,100,1000]
SET	Boolean, Long, String, Date, Timestamp, FloatingPoint,	Validates that the object attribute value is not null. Only the keyword SET is needed to specify the operation.	SET

Operator	Value Types	Description	Example Load Conditions
	FloatingPoint-WithUnits		
NOT_SET	Boolean, Long, String, Date, Time-stamp, Floa-tingPoint, FloatingPoint-WithUnits	Validates that the object attribute value is null. Only the keyword NOT_-SET is needed to specify the operation.	NOT_SET
EQUALS	String	Validates that the object attribute value is the specified string.	EQUALS (MAJOR)
BEGINSWITH	String	Validates that the object attribute value starts with a specified string.	BEGINSWITH (PN)
ENDSWITH	String	Validates that the object attribute value ends with a specified string.	ENDSWITH(0)
CONTAINS	String	Validates that the object attribute value contains the specified string.	CONTAINS(PN0)
LENGTH	String	Validates that the object attribute string length is within the specified range.	LENGTH(0..40) LENGTH(0..) LENGTH(..40)

Note

The system locale is used for the format of the dates and timestamps in the attribute condition.

Release Target Rule

This business rule 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 life cycle transition rules. If the user-selected transition is not defined for the current resulting object transition, the rule fails.

Any undefined change process transition value in the resulting objects associated with the plan for release is assumed to be the **Change** transition and its defined state. If the change process transition is not defined for the resulting object, the rule fails. If the resulting object does not have a defined change process transition, but

is already at the release target state that is specified by the change process transition, the rule passes. For more information about transitions, see Defining State Transitions.

BOM Release Rule

This business rule ensures that all resulting objects are at an appropriate state and all their first-level dependants are at the appropriate state. If a child object is also a resulting object, that child goes to the target state when released through the change notice, instead of staying at the current state.

Required configuration options include:

- targetState—a valid state for the resulting objects. You can define more than one targetState.
- validDependentState—a valid state for dependent objects. You can define more than one validDependentState.
- collectionComponentID—the collection component identifier used to collect the dependent objects. The default collection component identifier COLLECT_ITEMS_FOR_BOM_RELEASE_RULE is available with your Windchill system. Each collection component identifier has preferences that determine which objects are collected. The **BOM Maturity Release Collector** preference in the **Business Rules** section of the **Preference Management** utility controls the collection rules.

The following illustrate using the BOM Release Rule:

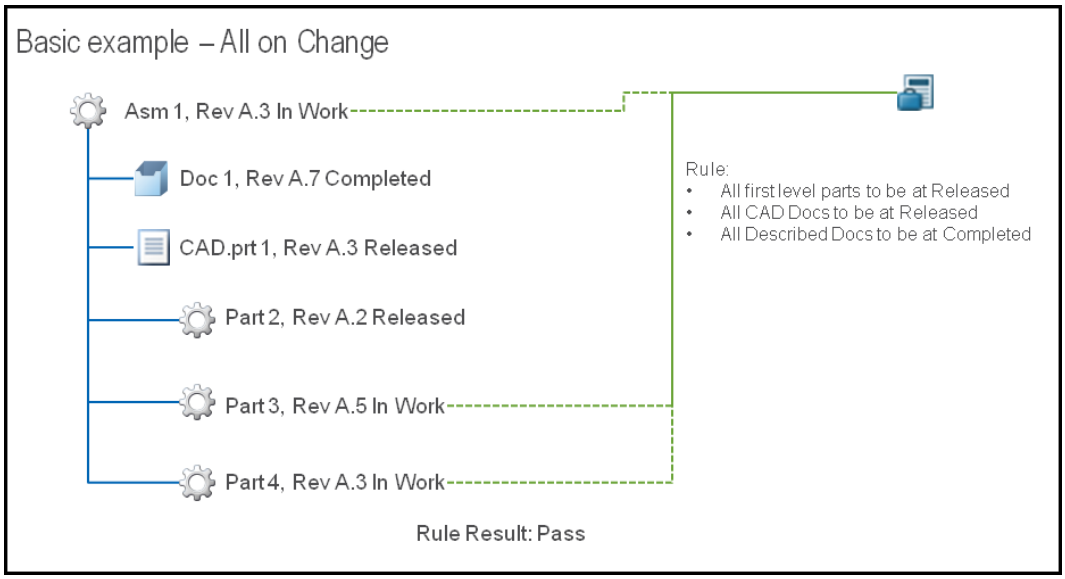
- Any assembly being released to the **Released** state must have all dependent component parts at the **Released** state.
- Any assembly being released to the **Prototype** state must have all dependent component parts at either the **Prototype** or **Released** state.

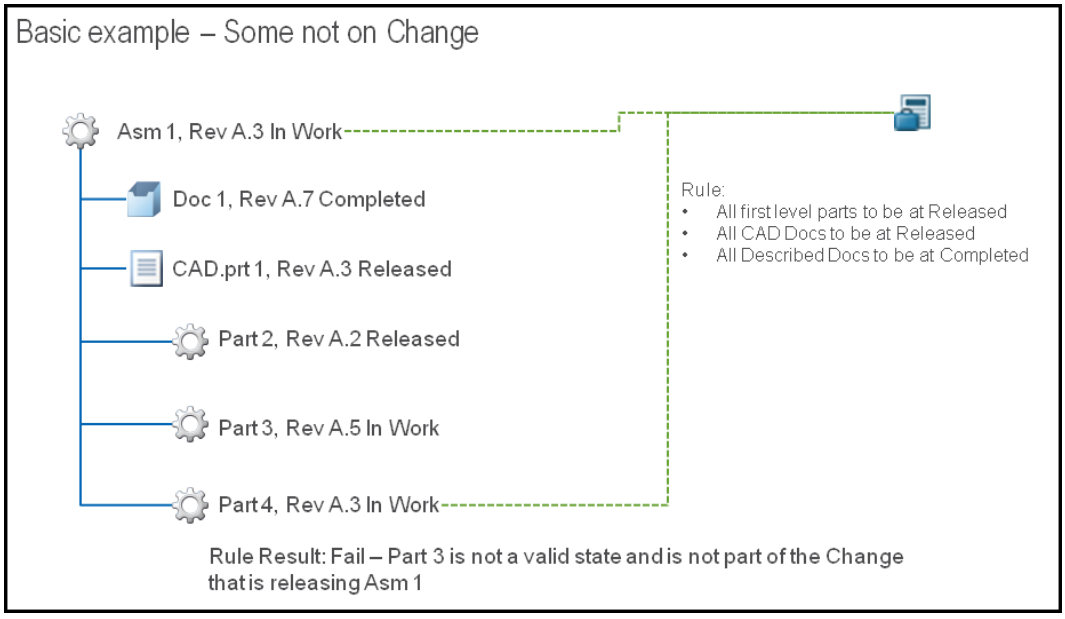
If there are any dependent component parts that are in an invalid state, the rule fails and a list of invalid assemblies are listed in the **View Conflicts** report.

Example configuration attributes:

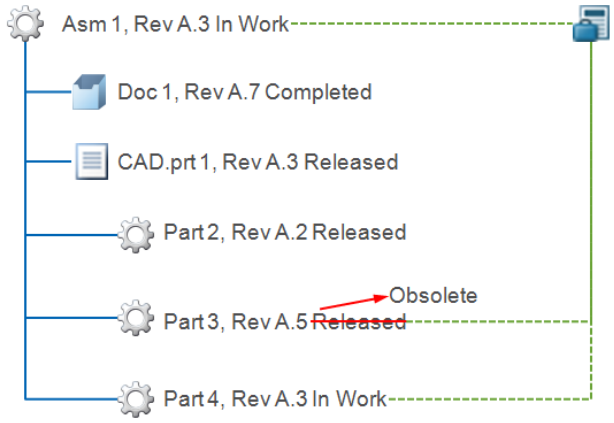
```
<configs>
   <config name="targetState" value="RELEASED"/>
   <config name="validDependentState" value="RELEASED"/>
   <config name="validDependentState" value="PROTOTYPE"/>
   <config name="collectionComponentID"
value="COLLECT_ITEMS_FOR_BOM_RELEASE_RULE"/>
</configs>
```

The following examples illustrate the BOM Release rule.


Basic example – All on Change
Asm 1, Rev A.3 In Work
Doc 1, Rev A.7 Completed
CAD.prt 1, Rev A.3 Released
Part 2, Rev A.2 Released
Part 3, Rev A.5 In Work
Part 4, Rev A.3 In Work
Rule:
• All first level parts to be at Released
• All CAD Docs to be at Released
• All Described Docs to be at Completed
Rule Result: Pass


Basic example – Some not on Change
Asm 1, Rev A.3 In Work
Doc 1, Rev A.7 Completed
CAD.prt 1, Rev A.3 Released
Part 2, Rev A.2 Released
Part 3, Rev A.5 In Work
Part 4, Rev A.3 In Work
Rule:
• All first level parts to be at Released
• All CAD Docs to be at Released
• All Described Docs to be at Completed
Rule Result: Fail – Part 3 is not a valid state and is not part of the Change that is releasing Asm 1

Invalid State – One part being Obsoleted



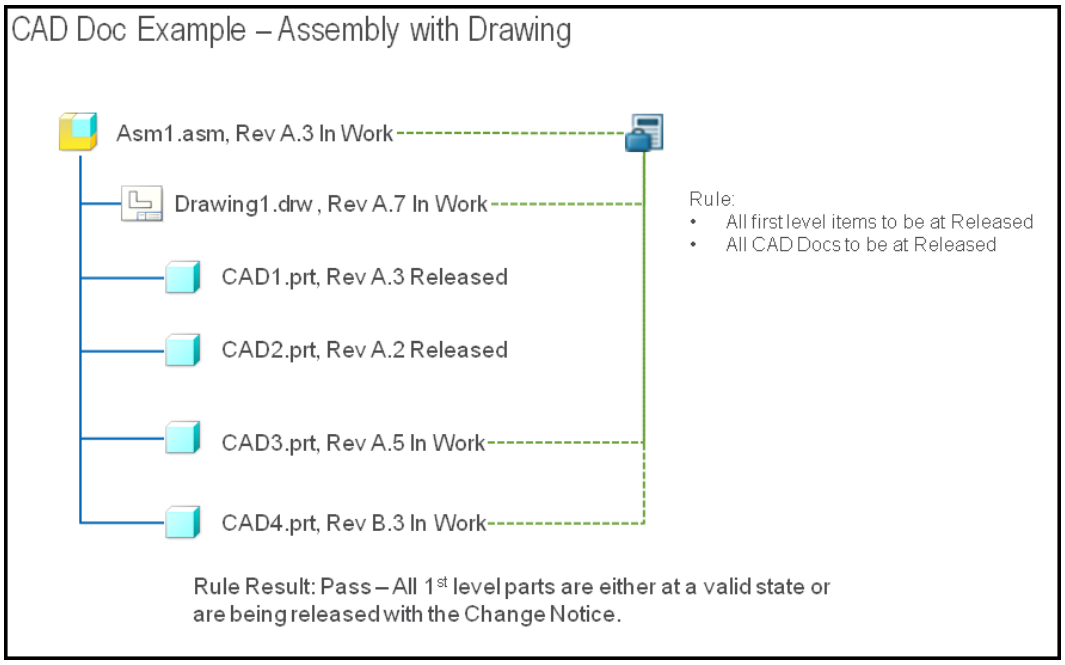
Rule:
- All first level parts to be at Released
- Invalid state for parts = Obsolete
- All CAD Docs to be at Released
- All Described Docs to be at Completed

Rule Result: Fail – As Part of the Change Part 3 is being set to Obsolete. Asm 1 will then fail the validation as it will contain an Obsolete part as a result.

CAD Doc Example - Assembly

Rule:
- All first level items to be at Released
- All CAD Docs to be at Released

Rule Result: Pass – All 1st level parts are either at a valid state or are being released with the Change Notice.



Available Business Rule Set

The CHANGEABLE_PRE_RELEASE business rule set is delivered with your system. It includes:

- Checkout Rule
- Release Target Rule

The change notice workflow executes the CHANGEABLE_PRE_RELEASE business rule set in the **Audit Change Notice** and **Resolve Release Conflicts** conditionals.

Note

If you have customized workflows or have updated your system from a previous version of Windchill, you will have to manually configure the workflows to include the business rule set.

Administration of Business Rules

The system or business administrator has the following duties:

- Configure the appropriate business rules for use
- Maintain the business rules
- Diagnose and manage issues with rules

Defining New Business Rules

You can define your own business rules by creating XML files and loading them. See the following examples:

1. Define the business rule.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE BusinessRule SYSTEM "standardX20.dtd">
<BusinessRule>
 <ObjectID><localId>wt.businessRules.BusinessRule:45346</localId>
</ObjectID>
 <objectContainerPath>/</objectContainerPath>

<key>IXBBusinessRule1_key</key>

 <name>
com.ptc.windchill.enterprise.change2.change2ClientResource:
CHECK_OUT_VALIDATOR_RULE_NAME</name>

 <description>
com.ptc.windchill.enterprise.change2.change2ClientResource:
CHECK_OUT_VALIDATOR_RULE_DESC</description>

 <enabled>true</enabled>

 <configs>
  <config name="key1" value="value1"></config>
  <config name="key2" value="value1"></config>
  <config name="key1" value="value2"></config>
 </configs>
</BusinessRule>
```

2. Define the business rule set.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE BusinessRuleSet SYSTEM "standardX20.dtd">
<BusinessRuleSet>
 <ObjectID><localId>wt.businessRules.BusinessRuleSet:45346</localId>
</ObjectID>
 <objectContainerPath>/wt.inf.container.OrgContainer=PTC/wt.pdmlink.
PDMLinkProduct=GOLF_CART>
</objectContainerPath>

<key>IXBBusinessRuleSet1_key</key>

 <name>
com.ptc.windchill.enterprise.change2.change2ClientResource:
CHANGE_PRE_RELEASE_RULESET_NAME</name>

 <description>
com.ptc.windchill.enterprise.change2.change2ClientResource:
CHANGE_PRE_RELEASE_RULESET_DESC</description>

 <enabled>true</enabled>

 <overridable>true</overridable>
</BusinessRuleSet>
```

3. Define the business rule link.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE BusinessRuleSet SYSTEM "standardX20.dtd">
<BusinessRuleLink>
 <ObjectID><localId>wt.businessRules.BusinessRuleLink:356747</localId>
</ObjectID>

<ruleset><ObjectReference><localId>wt.businessRules.BusinessRuleSet:
45345</localId></ObjectReference></ruleset>

<rule><ObjectReference><localId>wt.businessRules.BusinessRule:45346
</localId></ObjectReference></rule>
<blockNumber>10<blockNumber>
</BusinessRuleLink>
```

4. Load the XML files. For more information, see Loading Business Rules Objects on page 12.

Updating Existing Business Rules

You can update existing rules by adding the updateIfExists attribute to the business rules rule sets load files and setting the value of the attribute to true. If the attribute is not in the load file or the attribute is set to false, you cannot update existing rules and you get an error when you attempt to reload the load files.

Note

Because the key and context creates the BusinessRule object uniqueness, you cannot update both even if the updateIfExists attribute is set to true.

Here is an example of how to enable updating:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE BusinessRule SYSTEM "standardX20.dtd">
<BusinessRule>
   ...
   <updateIfExists>true</updateIfExists>
   ...
</BusinessRule>
```

Loading Business Rules Objects

Use this procedure:

1. Add the load files to a zip file. For example, add the BusinessRulesObjects in a zip file such as:

   `<windchill>/loadFiles/ixbImport/BusinessRuleObjects.zip.`

   ### *Note*

   The zip file must be in a folder path within the <windchill> directory.

2. Create the XML load file to load the generated zip file created in step 1.

Here is an example:

```
<?xml version="1.0"?>
<!DOCTYPE NmLoader SYSTEM "standardX20.dtd">

<NmLoader>
  <csvExecuteImport handler="wt.load.LoadImport.executeImport">

  <csvimportFilename>loadFiles/ixbImport/businessRulesObjects.zip
</csvimportFilename>
  </csvExecuteImport>
</NmLoader>
```

3. Using a windchill shell, execute the following load command to load the business rules objects.

   `windchill wt.load.LoadFromFile –d <PATH>\loadFileName`

   where loadFileName refers to the file created in step 2. Executing the command stores the business rule objects in the database, ready to be configured.

Deleting Business Rule Objects

You can delete existing business rule objects by using the DeleteBusinessRuleObjects command line utility.

To delete a business rule or business rule set, run the following command:

```
windchill wt.businessRules.DeleteBusinessRuleObjects “typeOfObjectToDelete:
key:containerPath”
```

To delete a business rule link, both the business rule set and the business rule need to be specified:

```
windchill wt.businessRules.DeleteBusinessRuleObjects "typeOfObjectToDelete:
Key:containerPath:Key:containerPath"
```

Argument Name	Description	Examples
typeOfObjectToDelete	The type of object to delete	BusinessRule BusinessRuleSet BusinessRuleLink
Key	The unique key of the object to delete	CHECKOUT_RULE CHANGEABLE_PRE_RELEASE
containerPath	The context path of the object	/wt.inf.container. OrgContainer=PTC /wt.pdmlink. PDMLinkProduct=GOLF_CART

Note

The argument for the object to delete must be in quotes since context paths may contain spaces.

If a business rule or business rule set is deleted, any link to which the object was associated. You can specify one or more objects, separated by a space.

The following table shows examples of deleting business rule objects:

Deleting a business rule	`windchill wt.businessRules.DeleteBusinessRuleObjects "BusinessRule|keyToDelete|/wt.inf.container.OrgContainer=Org Name/wt.pdmlink.PDMLinkProduct=Product Name"`
Deleting two business rule sets	`windchill wt.businessRules.DeleteBusinessRuleObjects "BusinessRuleSet|keyToDelete|/wt.inf.container.OrgContainer=Org Name/wt.pdmlink.PDMLinkProduct=Product Name" "BusinessRuleSet|anotherKey|/wt.inf.container.OrgContainer=Org Name/wt.pdmlink.PDMLinkProduct=Product Name"`
Deleting a business rule link	`windchill wt.businessRules.DeleteBusinessRuleObjects "BusinessRuleLink|businessRuleSetKey|/wt.inf.container.OrgContainer=Org Name/wt.pdmlink.PDMLinkProduct=Product Name|businessRuleKey|/wt.inf.container.OrgContainer=Org Name/wt.pdmlink.PDMLinkProduct=Product Name"`

Business Rules Execution

The business rules engine executes registered rules in a business rule set and returns validation results.

The phases of the execution of a business rule set include:

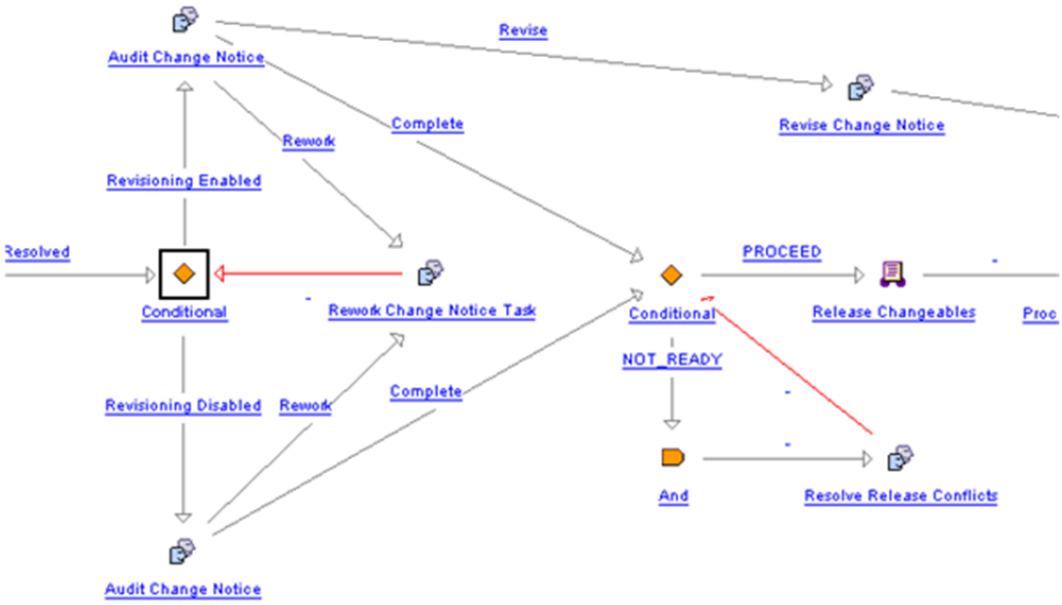
1. Retrieve the specified objects to be validated
2. Retrieve the business rule set
3. Generate and execute a plan for the business rule sets
4. Report the results

See the Windchill Customizer's Guide for more information about these steps.

Calling the Business Rules Engine

The default change notice workflow demonstrates how to call the business rules engine from a workflow. The default change notice workflow comes with two conditionals that call the business rules engine: the **Audit Change Notice** and the **Resolve Release Conflicts** tasks shows the results of the conditionals that invoke the business rules engine. The **Audit Change Notice** task shows the rule conflicts, allowing you to proceed. The **Resolve Release Conflicts** task shows the rule conflicts that must be corrected before you proceed.

If you want to be able to bypass business rule conflicts, see Example for Bypassing Business Rule Conflicts on page 18



The business rules engine retrieves the set of business rules given the business rule set unique key and the context reference of the objects. The business rules are executed against the objects. See the Javadoc for supported APIs.

Business Rule Sets Results

If the validation results are not successful, a **View Conflicts** link appears on the **Audit Change Notice** or **Resolve Release Conflicts** task pages. To show the results of the business rule sets on a change notice, open the task. Click the **View Conflicts** link to determine the rules that failed and why.

If there is no **View Conflicts** link, all business rules were validated.

Note

If you have customized workflows or have updated your system from a previous version of Windchill, you will have to manually configure the workflows to include the business rule set.

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, create a new workflow variable on that task named businessRulesResultSet. On the **Initialize From** field, set the value to be businessRulesResultSetGlobal, and the**Copy Into** field 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** link displays as part of the subject in the workflow task. To see the report of evaluated business rule sets, click the link.

The **Audit Change Notice** task presents the errors, but allows you to continue.

Rule Set Hierarchy

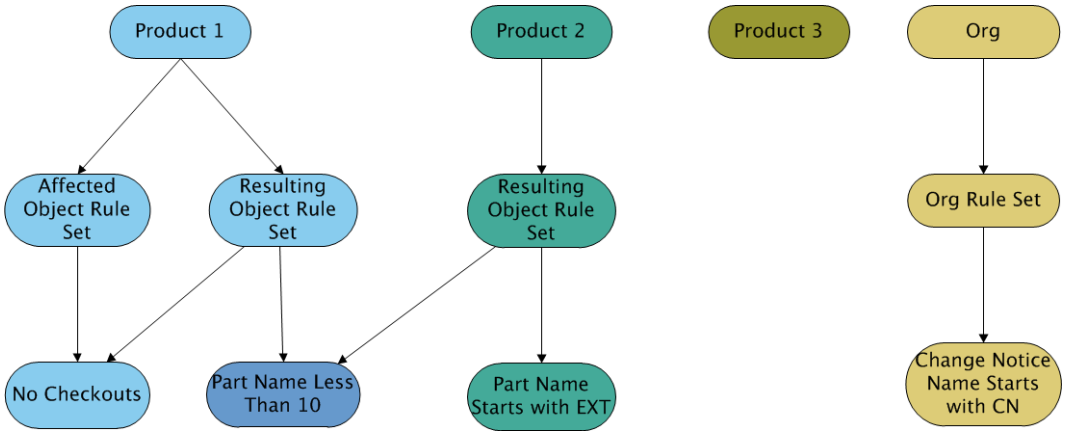
The following table shows that the context of the object determines the adherence of the rule set:

Context	Site Enabled	Site Overridable	Organization Enabled	Organization Overridable	Product Enabled	Result
Site	Y	Y	Y	Y	Y	Site
Site	N	Y	Y	Y	Y	
Organization	Y	Y	Y	Y	Y	Organization
Organization	Y	N	Y	Y	Y	Site
Organization	Y	Y	N	Y	Y	Site
Organization	Y	Y	Y	N	Y	Organization
Organization	N	Y	N	N	Y	
Product	Y	Y	Y	Y	Y	Product
Product	Y	N	Y	Y	Y	Site
Product	Y	Y	Y	N	Y	Organization
Product	Y	Y	Y	Y	N	Organization
Product	Y	Y	N	Y	N	Site
Product	N	Y	N	Y	N	

Example of Multiple Rules, Rule Sets, and Relationships

Suppose you have a need to configure rules against multiple products and multiple relationships. Consider the following example:

Product 1	Product 2	Product 3	Organization
Two rule sets: • Affected objects (checkout rule) • Resulting objects (checkout rule and part attribute rule—name must be less than 10)	One rule set: • Resulting objects (part attribute rule—name must be start with EXT)	No rules or rule sets	One rule set: • Change notice rule—name must start with CN



After all the rules are created, you could configure the workflow to use these rule sets by using code such as the following:

```
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 you 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 you had a document in Product 2 that was a resulting object, then there are no rules for it because there are only rules for parts.

Example for Bypassing Business Rule Conflicts

You might want to show some or all business rule failures, but not prevent the workflow from continuing or completing. Create two unique business rule sets, one in which the rules are required for continuing or completing the process and another in which failures of rules are reported but are not required to complete the process. Having two separate business rule sets allow you to configure a workflow process to call either rule set at different points in the workflow process.

The following example shows how to configure a set of rules that are required to proceed and a second set of rules that are not required. Assume that you want to check that certain attributes are filled in (in this case, Material). If the attributes are blank, the process will continue; however, the release target rule must pass.

1. Create a nonrequired business rule set for attribute checks.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE BusinessRuleSet SYSTEM "standardX20.dtd">
<BusinessRuleSet>
 <ObjectID><localId>wt.businessRules.BusinessRuleSet:10001</localId>
 </ObjectID>

<objectContainerPath>/wt.inf.container.OrgContainer=
 Demo Organization/wt.pdmlink.PDMLinkProduct=Example1TestProduct
 </objectContainerPath>

<key>EXAMPLE1_NON_REQUIRED_ATTRIBUTE_RULE_SET</key>
 <name>EXAMPLE1_NON_REQUIRED_ATTRIBUTE_RULE_SET</name>
 <description>EXAMPLE1_NON_REQUIRED_ATTRIBUTE_RULE_SET</description>
 <enabled>true</enabled>
 <overridable>true</overridable>
 <updateIfExists>true</updateIfExists>
</BusinessRuleSet>
```

2. Create an attribute rule for Material.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE BusinessRule SYSTEM "standardX20.dtd">
<BusinessRule>
 <ObjectID><localId>wt.businessRules.BusinessRule:10002</localId>
 </ObjectID>

 <objectContainerPath>/wt.inf.container.OrgContainer=
```

```
 Demo Organization/wt.pdmlink.PDMLinkProduct=Example1TestProduct
 </objectContainerPath>

  <key>EXAMPLE1_ATTRIBUTE_RULE</key>
  <selector>ATTRIBUTE_RULE</selector>
  <name>Example 1 Attribute Rule</name>
  <description>Example 1 Attribute Rule</description>
  <enabled>true</enabled>
  <updateIfExists>true</updateIfExists>
  <configs>
    <config name="objectType" value="com.ptc.Part"></config>
    <config name="Material" value="SET"></config>
  </configs>
</BusinessRule>
```

3. Create a link for the attribute rule and the nonrequired rule set.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE BusinessRule SYSTEM "standardX20.dtd">
<BusinessRule>
  <ObjectID><localId>wt.businessRules.BusinessRule:10002</localId>
  </ObjectID>

  <objectContainerPath>/wt.inf.container.OrgContainer=Demo Organization
  /wt.pdmlink.PDMLinkProduct=Example1TestProduct</objectContainerPath>

  <key>EXAMPLE1_ATTRIBUTE_RULE</key>
  <selector>ATTRIBUTE_RULE</selector>
  <name>Example 1 Attribute Rule</name>
  <description>Example 1 Attribute Rule</description>
  <enabled>true</enabled>
  <updateIfExists>true</updateIfExists>
  <configs>
    <config name="objectType" value="com.ptc.Part"></config>
    <config name="Material" value="SET"></config>
   </configs>
</BusinessRule>
```

4. Create a required business rule set for release target rule.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE BusinessRuleSet SYSTEM "standardX20.dtd">
<BusinessRuleSet>
  <ObjectID><localId>wt.businessRules.BusinessRuleSet:20001</localId>
  </ObjectID>

  <objectContainerPath>/wt.inf.container.OrgContainer=Demo Organization
  /wt.pdmlink.PDMLinkProduct=Example1TestProduct</objectContainerPath>

  <key>EXAMPLE1_RELEASE_RULE_SET</key>
  <name>Example1_ReleaseRuleSet</name>
  <enabled>true</enabled>
  <description>Example1_ReleaseRuleSet</description>
  <overridable>true</overridable>
  <updateIfExists>true</updateIfExists>
</BusinessRuleSet>
```

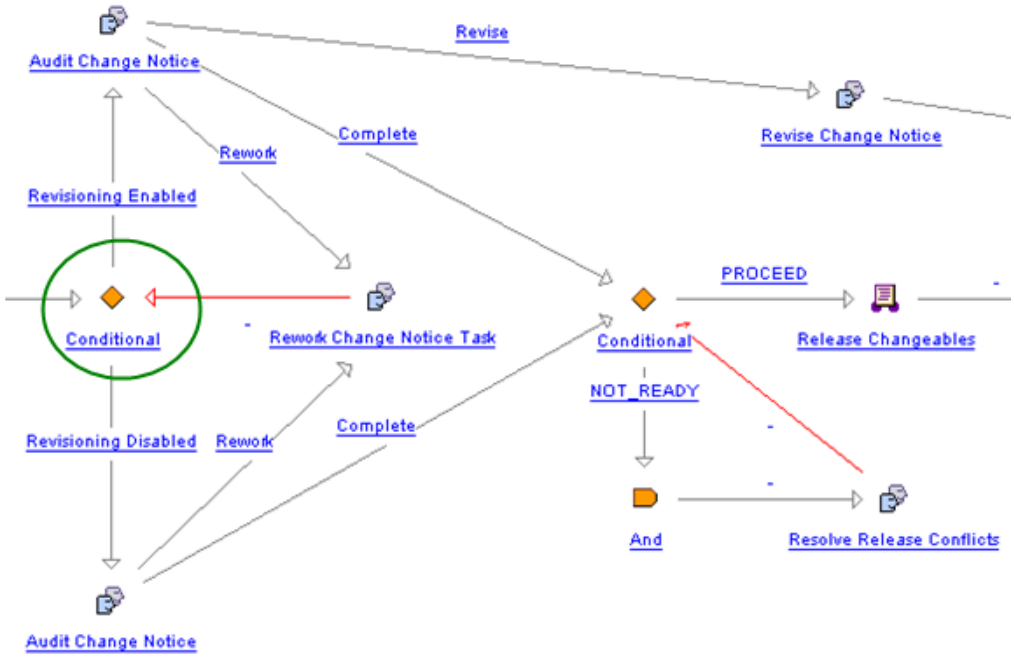
5. Reload the site release target rule.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE BusinessRule SYSTEM "standardX20.dtd">
<BusinessRule>
  <ObjectID><localId>wt.businessRules.BusinessRule:20002</localId>
  </ObjectID>

  <key>RELEASE_TARGET</key>
  <selector>RELEASE_TARGET</selector>
  <name>com.ptc.windchill.enterprise.change2.change2ClientResource:
 RELEASE_TARGET_RULE_NAME</name>

 <description>com.ptc.windchill.enterprise.change2.
 change2ClientResource:RELEASE_TARGET_RULE_DESC</description>

 <enabled>true</enabled>
 <updateIfExists>true</updateIfExists>
</BusinessRule>
```

6. Create the link for the target rule and required rule set.

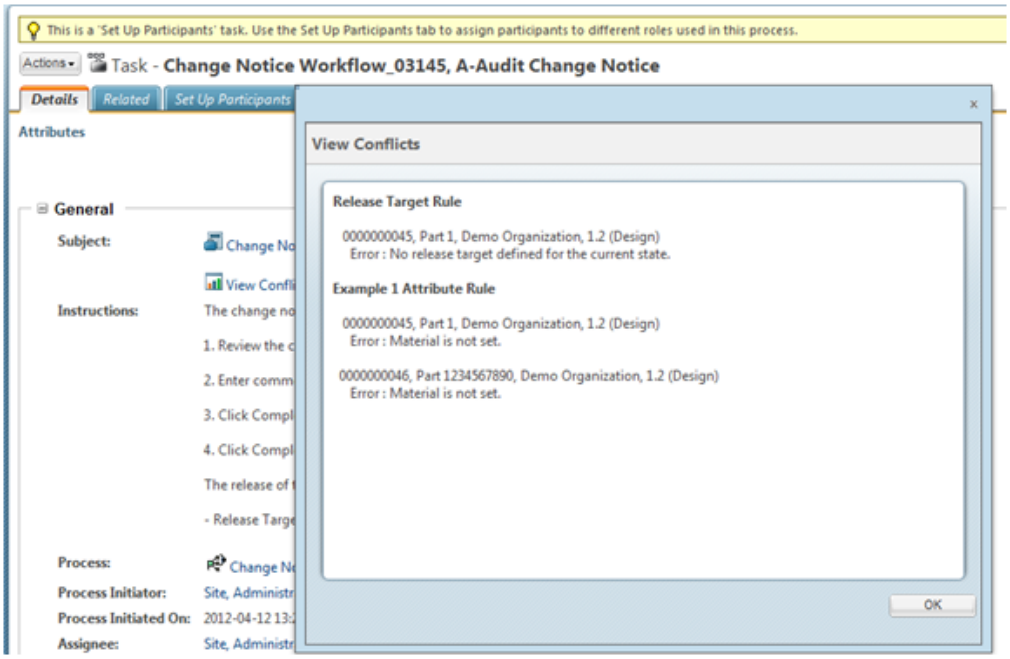
```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE BusinessRuleLink SYSTEM "standardX20.dtd">
 <BusinessRuleLink>
  <ObjectID><localId>wt.businessRules.BusinessRuleLink:20003</localId>
  </ObjectID>

  <ruleSet><ObjectReference><localId>wt.businessRules.BusinessRuleSet:
  20001</localId></ObjectReference></ruleSet>

  <rule><ObjectReference><localId>wt.businessRules.BusinessRule:20002
  </localId></ObjectReference></rule>

  <blockNumber>1</blockNumber>
  <updateIfExists>true</updateIfExists>
</BusinessRuleLink>
```

7. Execute all required and nonrequired business rule sets. To display all required and nonrequired rule conflicts, update the **Audit Change Notice** workflow task conditional to use both rule sets EXAMPLE1_RELEASE_RULE_SET and EXAMPLE1_NON_REQUIRED_ATTRIBUTE_RULE_SET.




```
if (wt.change2.ChangeHelper2.isTrackingChange((wt.inf.container.
  WTContained)primaryBusinessObject)) {

  result = "Revisioning Enabled";
}else {
  result = "Revisioning Disabled";
  com.ptc.core.businessRules.engine.BusinessRuleSetBean[] beans = new
  com.ptc.core.businessRules.engine.BusinessRuleSetBean[] {

   // Configure to call non-required rule set
   com.ptc.core.businessRules.engine.BusinessRuleSetBean.newBusinessRuleSetBean
   ("EXAMPLE1_NON_REQUIRED_ATTRIBUTE_RULE_SET", "wt.change2.ChangeRecord2"),

   // Configure to call required rule set
   com.ptc.core.businessRules.engine.BusinessRuleSetBean.newBusinessRuleSetBean
   ("EXAMPLE1_RELEASE_RULE_SET", "wt.change2.ChangeRecord2")
  };
}
com.ptc.core.businessRules.validation.RuleValidationResultSet resultSet =
wt.businessRules.BusinessRulesHelper.engine.execute(primaryBusinessObject, beans);

if ( resultSet.hasResultsByStatus(com.ptc.core.businessRules.validation.
  RuleValidationStatus.FAILURE)) {

  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,
```

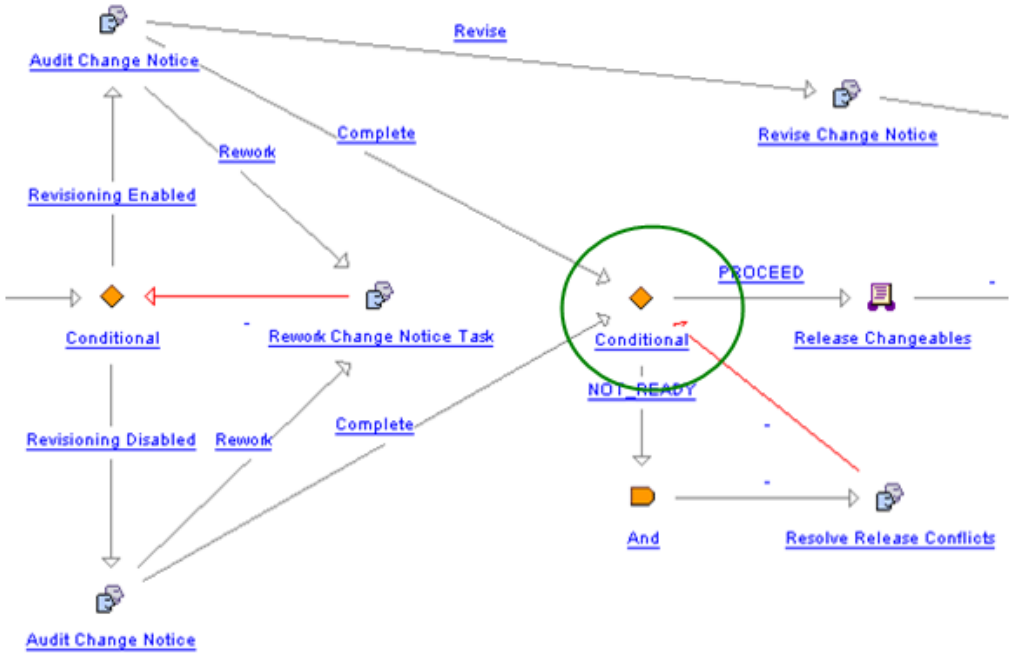
```
 null).getLocalizedMessage();

 preReleaseConflictsMsg = preReleaseConflictsMsg + "\n" + resultSet.
 getFailedRulesMessage(java.util.Locale.getDefault());
}
```

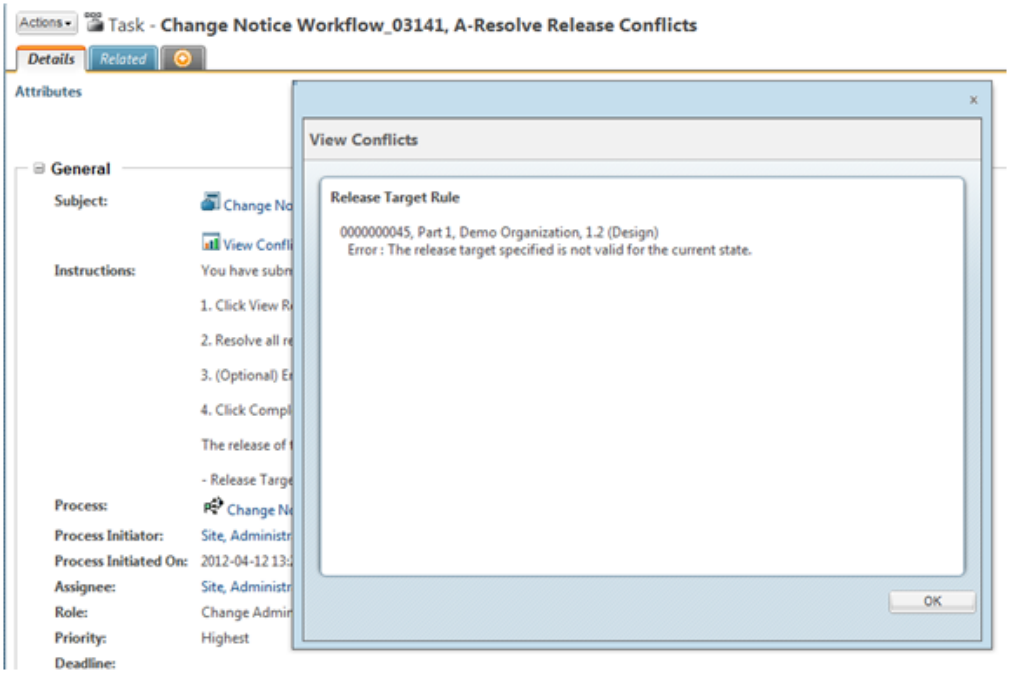


8. Execute the required business rule sets. In order to prevent the change notice workflow process for proceeding for the required business rules, update the conditional to use only the required rule set EXAMPLE1_RELEASE_RULE_SET for the **Resolve Release Conflicts** workflow task.

```
result = "NOT_READY";

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

  // Configure to call required rule set only
  com.ptc.core.businessRules.engine.BusinessRuleSetBean.
  newBusinessRuleSetBean("EXAMPLE1_RELEASE_RULE_SET",
  "wt.change2.ChangeRecord2")
};

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

if ( !resultSet.hasResultsByStatus(com.ptc.core.businessRules.
  validation.RuleValidationStatus.FAILURE)) {

  result = "PROCEED";
} else {
```

```
  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());
}
```





The **Resolve Release Conflicts** workflow task continues to regenerate the task until all conflicts have been resolved.

Troubleshooting Configuration Issues

To help you diagnose unexpected results, enable the following loggers on the method server. To enable the logging, follow these steps:

1. Navigate to `\<windchill>\codebase\WEB-INF`
2. Open log4jMethodServer.properties with a text editor
3. Add the following lines:
   - To view potential configuration issues, such as rules or rule sets not being found in a context:

     ```
     log4j.logger.com.ptc.core.businessRules.engine=WARN
     ```
   - To view potential performance issues, such as execution time of a business rule validation:

     ```
     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, remove what was previously added or set the following:

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

4. Reboot the method server.