Rule Resolution: Advanced Topics

PegaRULES Process Commander v 5.4

This document is the property of:
Pegasystems Inc.
101 Main Street
Cambridge, MA 02142-1590

Phone: (617) 374-9600
Fax: (617) 374-9620
www.pega.com

PegaRULES Process Commander
Document: Rule Resolution
Software Version 5.4
Updated: April 18, 2008

Contents

Introduction

Benefits

Pegasystems provides an architecture that separates the strategic business logic (the "rules") from both the application and the underlying operating system. Just as business data (i.e., customer information, transaction history, and accounting data) belongs in its own database, rules regarding business practices belong in a distinct business rules database (the **rulebase**). Rules are a corporate asset, and separating rules from the data and the code makes it possible to codify your business rules and manage them more effectively. Rule Resolution is the mechanism that makes this separation feasible in practice with Java-speed performance.

Pegasystems' patented Rules technology is the foundation of Process Commander BPM solutions, providing flexibility and agility. Through dynamic rule selection, Process Commander provides flexibility that is otherwise impossible (except with the alternative of applying rules specified in code). Rule resolution identifies and executes the right rule at the right time, without the need for additional context-specific code (such as nested "if-then" statements).

Process Commander rule resolution simplifies the management and extension of business and process rules in a production environment. Rule resolution is more effective than storing and managing rules with Enterprise JavaBeans (EJBs), Web services, or Java code – where the right rule must be determined at *design* time of the code (as opposed to run-time). In contrast to those solutions, rule resolution permits "late-bound" decisions to rules at run time. (Late-bound decision making is the binding or grouping of relevant rules assigned as "on-call" so that the application returns a correct answer for a particular decision.)

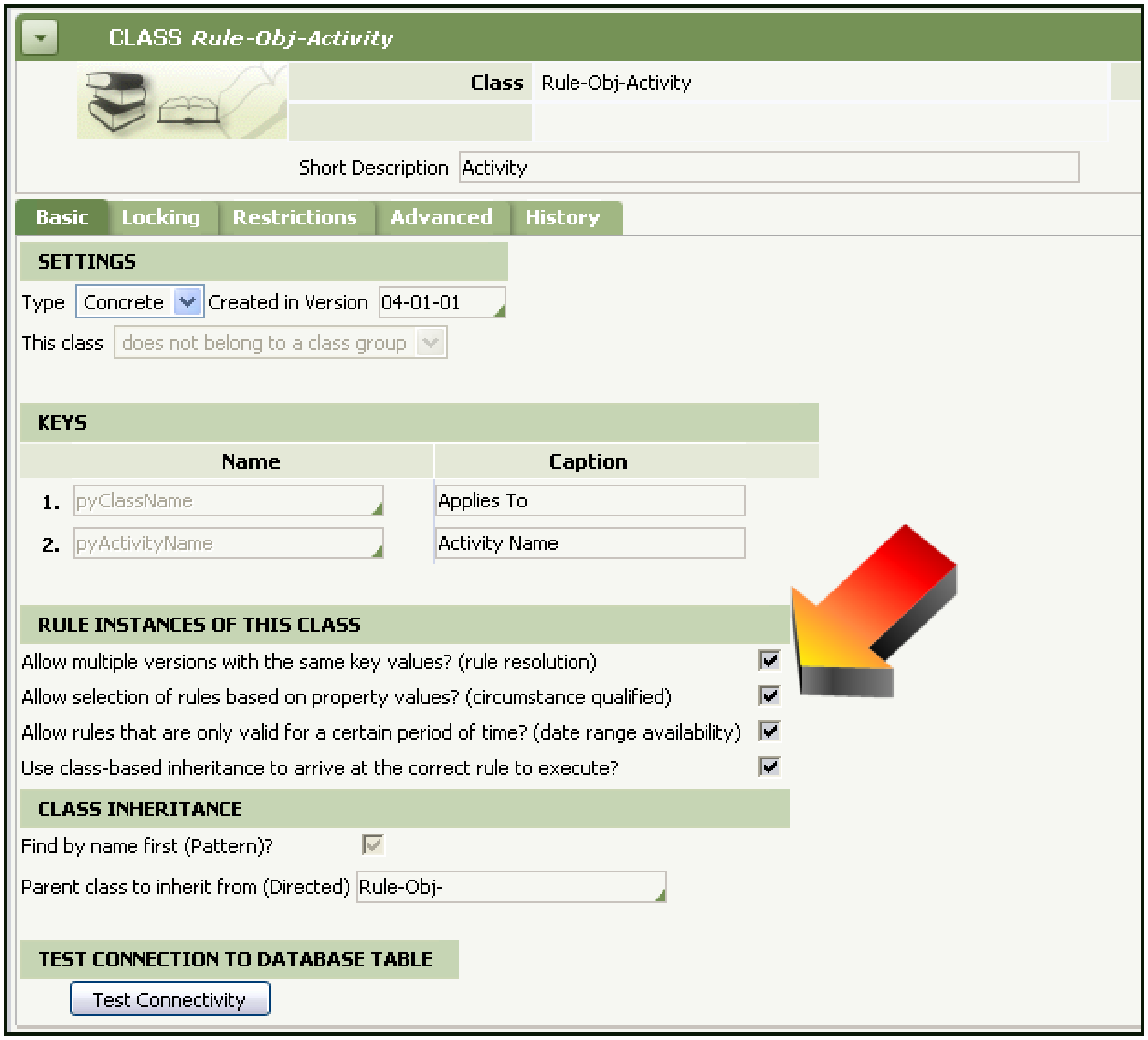
IMPORTANT: This tech note is intended to be a companion piece to the articles on the *Reuse and Version Management* page of the PDN. That page includes descriptions of the new Multivariate Circumstancing functionality, availability options, and how to create differently-qualified rules. As the rest of this document assumes that readers are familiar with that information, it is strongly recommended to read those articles first.

Enabling Rule Resolution on rules

Rules, as defined in this document, are all instances of classes which descend from the **Rule-** class, including Activities (Rule-Obj-Activity), Streams (Rule-Stream), Models (Rule-Obj-Model), Properties (Rule-Obj-Property), etc.

The ability of a particular rule type (Activity, etc.) to be rule-resolved is dependent upon the definition of that rule, as shown by the Class form.

Important: Not all rules are rule-resolved; Rule Resolution must be enabled on the class definition.



CLASS *Rule-Obj-Activity*

Class	Rule-Obj-Activity
Short Description	Activity

SETTINGS

Type: Concrete — Created in Version: 04-01-01

This class: does not belong to a class group

KEYS

	Name	Caption
1.	pyClassName	Applies To
2.	pyActivityName	Activity Name

RULE INSTANCES OF THIS CLASS

Allow multiple versions with the same key values? (rule resolution)	✓
Allow selection of rules based on property values? (circumstance qualified)	✓
Allow rules that are only valid for a certain period of time? (date range availability)	✓
Use class-based inheritance to arrive at the correct rule to execute?	✓

CLASS INHERITANCE

Find by name first (Pattern)? ✓

Parent class to inherit from (Directed): Rule-Obj-

TEST CONNECTION TO DATABASE TABLE

When classes are defined, different features are enabled which affect inheritance and rule resolution. These features are contained in two sections on the Class form:

- Rule Instances of this Class
- Class Inheritance (described in a later section of this document)

Rule Instances of this class

This section of the class form enables rule resolution for the rule. The different checkboxes control different facets of rule resoution, and are described below so that developers may predict the rule resolution for these classes, based on the options checked in the rule definition.

Important: This option is only available for classes which start with **Rule-.** For existing Rule- classes, these settings *should not be changed* after instances of the class have been created. Changing the settings may make rule instances inaccessible.

Allow multiple versions with the same key values (Rule Resolution)

This is the most important choice for Rule Resolution, as it enables or disables the functionality. If this box is *not* checked, then none of the other checkboxes will appear on the class form.

If checked, this field allows more than one rule with the same name and class to be present in the system. When this box is checked, the following rule qualifiers may be enabled (i.e., Rule Resolution is used):

- RuleSet and Version
- Availability
- Circumstance (*also requires Circumstance checkbox)*
- Circumstance Date *(also requires Circumstance checkbox)*
- Date/Time (*also requires Date Range checkbox)*

Therefore, if this box is checked, Rule Resolution will be used to open instances of this class.

NOTE: It is possible for rules to have this box checked and not have the *Use class-based inheritance to arrive at the correct rule to execute* box checked. If only the *Allow multiple versions* box is checked, then rules (such as Rule-Message) may be saved and modified in different RuleSets/Versions, and rule resolution will be used to determine the correct rule; however, the “defined on” (or “Applies To”) class will not be used as one of the ranking factors, as it is not present.

Allow selection of rules based on property values (Circumstance Qualified)

This checkbox allows the rule to be Circumstance-qualified. If this box is *not* checked, Circumstances and Circumstance Dates may not be used with this rule. (For example, Declarative rules may not use Circumstances, and this feature is missing from those rules.)

NOTE: If the *Allow multiple versions with the same key values* box is not checked, then this choice is not valid.

Allow rules that are only valid for a certain period of time (Date Range Availability)

This checkbox allows the rule to be qualified by a Date Range. If this box is *not* checked, then a Date Range is not valid for this rule. (Declarative rules also may not use Date Ranges.)

NOTE: If the *Allow multiple versions with the same key values* box is not checked, then this choice is not valid.

Use class-based inheritance to arrive at the correct rule to execute

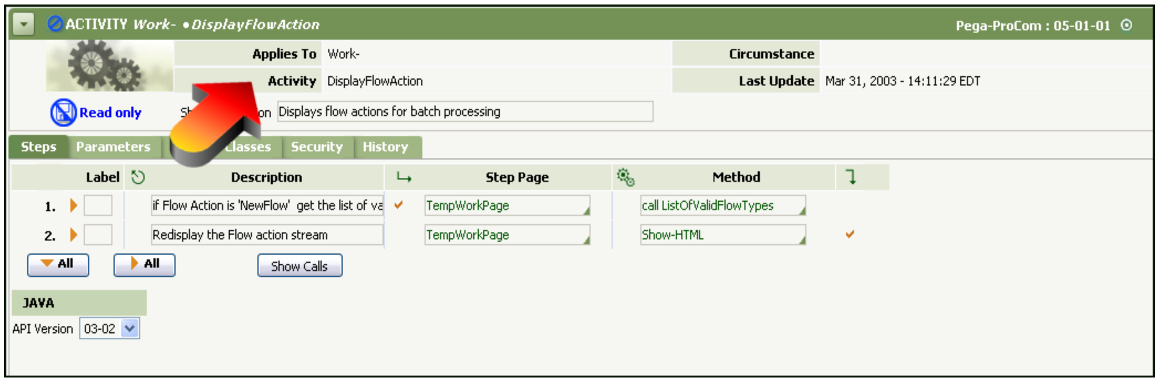
This field defines how inheritance (through the class hierarchy) affects this rule. For rules which are defined on other rules (such as Activities), this checkbox will affect how rule resolution searches for rules in the system. (Details on the functionality enabled by this checkbox are in the next major section of this document.)

Class Inheritance and Rule Resolution

As stated above, the **Use class-based inheritance to arrive at the correct rule to execute** checkbox defines how inheritance (through the class hierarchy) affects this rule. For this functionality, there are two main types of rules:

- those that can be defined as *attributes* or *aspects* of other classes
- those that are *not* aspects of other classes, but stand alone

There are some types of rules, like Activities, Models, Whens, Decision Tables, etc., which may be "applied to" or "defined on" other classes. These rules are considered *attributes* of other classes



ACTIVITY *Work- •DisplayFlowAction* Pega-ProCom : 05-01-01

Applies To	Work-	**Circumstance**	
Activity	DisplayFlowAction	**Last Update**	Mar 31, 2003 - 14:11:29 EDT

Read only

Sh on Displays flow actions for batch processing

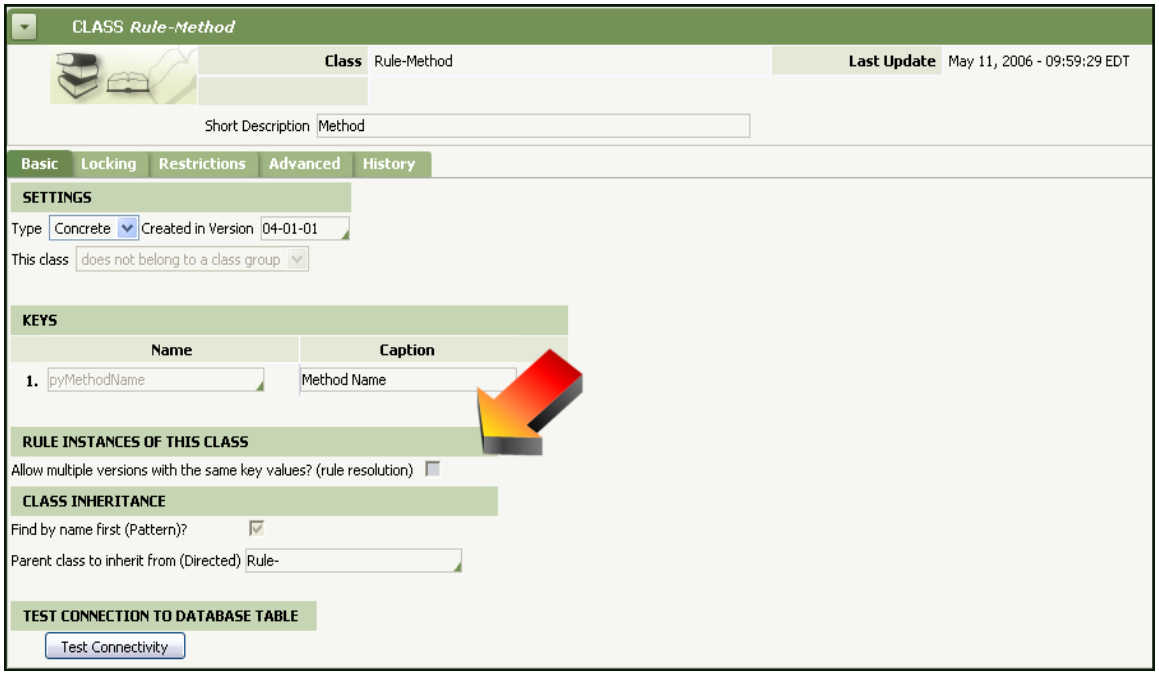
	Label	Description		Step Page	Method	
1.		if Flow Action is 'NewFlow' get the list of va	✓	TempWorkPage	call ListOfValidFlowTypes	
2.		Redisplay the Flow action stream		TempWorkPage	Show-HTML	✓

JAVA

API Version 03-02

In the above example, the activity **DisplayFlowAction** is *defined on* the class **Work-**.

Other rules, such as Rule-Method, do not have the **Use class-based inheritance** box checked.


CLASS *Rule-Method*

Class	Rule-Method	**Last Update**	May 11, 2006 - 09:59:29 EDT

Short Description: Method

SETTINGS

Type: Concrete — Created in Version: 04-01-01

This class: does not belong to a class group

KEYS

	Name	Caption
1.	pyMethodName	Method Name

RULE INSTANCES OF THIS CLASS

Allow multiple versions with the same key values? (rule resolution) ☐

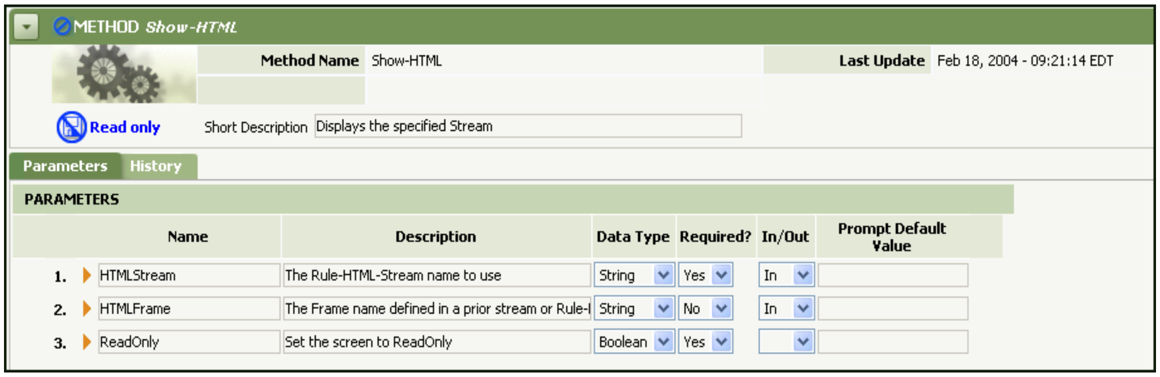
CLASS INHERITANCE

Find by name first (Pattern)? ✓

Parent class to inherit from (Directed): Rule-

TEST CONNECTION TO DATABASE TABLE

Instances of this rule will not be defined on any classes, and will not have the “Applies To” field on its instances.


METHOD *Show-HTML*

Method Name	Show-HTML	**Last Update**	Feb 18, 2004 - 09:21:14 EDT

Read only

Short Description: Displays the specified Stream

PARAMETERS

	Name	Description	Data Type	Required?	In/Out	Prompt Default Value
1.	HTMLStream	The Rule-HTML-Stream name to use	String	Yes	In	
2.	HTMLFrame	The Frame name defined in a prior stream or Rule-I	String	No	In	
3.	ReadOnly	Set the screen to ReadOnly	Boolean	Yes		

Inheritance

For rules such as activities which are attributes of (defined on) classes, *inheritance* applies.

PegaRULES models data by defining it in a class hierarchy, following an object-oriented paradigm (much like Java). As Java methods can be inherited from the Java class's parents, rules in PegaRULES such as activities can be inherited through the defined class hierarchy. Thus, an instance of the class *Work-Cover-GeneralTask* will not only be able to use the activities, properties, etc. which are defined on that class, but also those defined on the class *Work-Cover-*, and the class *Work-.*

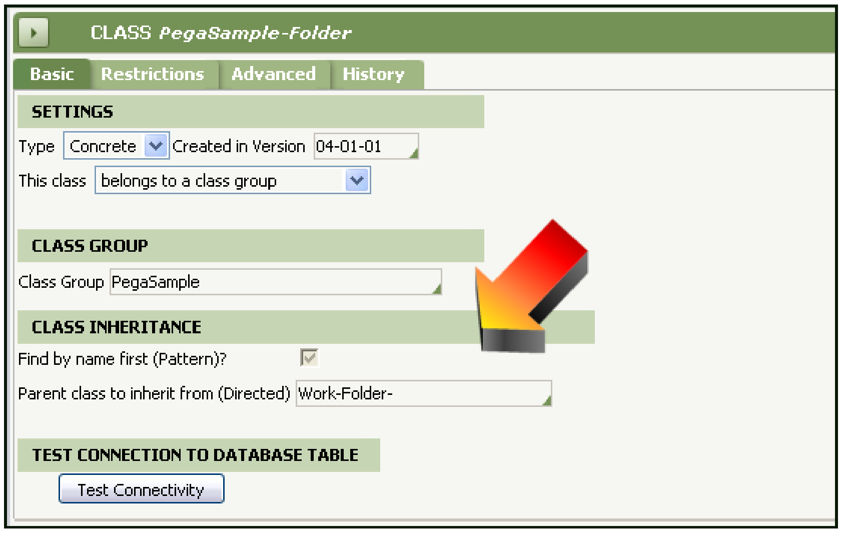
When a particular activity (or property or other such rule) is called during processing, the application will search for that rule in the class where it was called. If the rule is not present in that class, Process Commander will search other classes, based on specific procedures. This section describes the paradigm used to search the class hierarchy during rule resolution.

In PegaRULES, each class that is defined in the system may have one or two parents, depending upon the class specified for each type of inheritance. There are two types of inheritance:

- *pattern* inheritance
- *directed* inheritance

Note that directed inheritance *always* applies, whether pattern inheritance is enabled or not.

The **Class Inheritance** section of the class form defines the class hierarchy and inheritance for this class, which is then used when determining what rule to use in Rule Resolution.


CLASS *PegaSample-Folder*

SETTINGS

Type: Concrete — Created in Version: 04-01-01

This class: belongs to a class group

CLASS GROUP

Class Group: PegaSample

CLASS INHERITANCE

Find by name first (Pattern)? ✓

Parent class to inherit from (Directed): Work-Folder-

TEST CONNECTION TO DATABASE TABLE

Find by name first (Pattern) – determines *pattern* inheritance. When this box is checked, then when the system is searching for a particular rule defined on this class, it will use *pattern* inheritance to search through all the classes linked in the hierarchy.

***Parent class to inherit from (Directed*)** – determines *directed* inheritance. This field holds the immediate parent, by directed inheritance, to use for this class.

Pattern Inheritance

Pattern inheritance follows the *pattern* of the names and dashes in the class name to define inheritance. For example, the class Acme-Auto-ClaimsEntry-Accident might have the following ancestors in the class hierarchy:

Acme-Auto-ClaimsEntry-
Acme-Auto-ClaimsEntry
Acme-Auto-
Acme-Auto
Acme-
Acme

If pattern inheritance is being followed, then all Activities, Whens, Models, etc. that are defined on any of the above classes would be available to be applied to Acme-Auto-ClaimsEntry-Accident (unless there is another more specific version of the instance defined on the Acme-Auto-ClaimsEntry-Accident class itself).

Pattern inheritance is always checked first. Note that pattern inheritance will cause the system to search *all the way through* the class pattern hierarchy before checking directed inheritance.

If an instance of Rule-Obj-Activity called **Display** is defined on the class **Acme-Auto-ClaimsEntry** above, then this Activity could be applied (if there is nothing more specific available) to all classes that inherit from Acme-Auto-ClaimsEntry.

It is possible to independently change the definition of Display in any or all of these other classes, and then, depending upon Rule Resolution - what circumstances are present and what rules are being used at the time - different versions of **Display** will be selected by the system.   Rule Resolution will search for the activity **Display,** and choose the **Display** which is defined on the class closest (in the inheritance path) to the one that is being requested.

Directed Inheritance

Directed inheritance allows the user to specify or *direct* from what classes the current class may inherit.   Directed inheritance classes do not have to have a name which includes the “pattern parent prefix” of the current class. (For example, Acme-Auto-ClaimsEntry could have directed inheritance to Work-Object-; the directed inheritance does not have to follow the “Acme-Auto-“ prefix.)

A directed inheritance class is required for each class being defined. The field will default to the pattern class parent; the developer then may change this if they wish.

By default, all base classes (Work-, Data-, etc.) will have directed inheritance to @baseclass.

> **Important:** The system will not prevent users from creating classes with inheritance that is circular, although this is definitely NOT PegaRULES Best Practice. If this kind of circular inheritance does happen, however, the system knows to go through a class only once.
>
> Example:
>
> Acme-Auto-ClaimsEntry- ("ClaimsEntry Dash") inherits via Pattern inheritance from Acme-Auto-ClaimsEntry.
>
> Acme-Auto-ClaimsEntry has directed inheritance to Acme-Auto-ClaimsEntry- ("ClaimsEntry Dash").
>
> Again, **this is NOT PegaRULES Best Practice, and should never be created.** Developers who are creating classes should be very careful whether they create the class with a "dash" on the end or not.

Class Hierarchy Search Protocol

When the system is searching for a particular rule defined on a class (the activity "Display," for example), it will search by *pattern* inheritance first (unless the pattern inheritance is disabled – unchecked - which should be extremely rare[1]). First, the class itself is checked for the rule; then the class's parent, by pattern inheritance; then *that* class's parent, and so on, up the chain of inheritance. If a match is not found, the *first directed* inheritance link is followed, and then pattern inheritance is used again, until the system finds an appropriate match. (See Scenario 1 in the example.)

When Pattern inheritance is enabled, rule resolution will follow a name pattern all the way up through the pattern hierarchy, no matter what the inheritance settings are for classes higher in the pattern hierarchy.

Directed inheritance is evaluated on a class-by-class basis, and *only one link is followed at a time.* Although pattern inheritance allows the system to search through an entire chain of classes, directed inheritance will only search to the specified directed inheritance class, and then stop and evaluate the next place to search, depending upon the definition of that class.

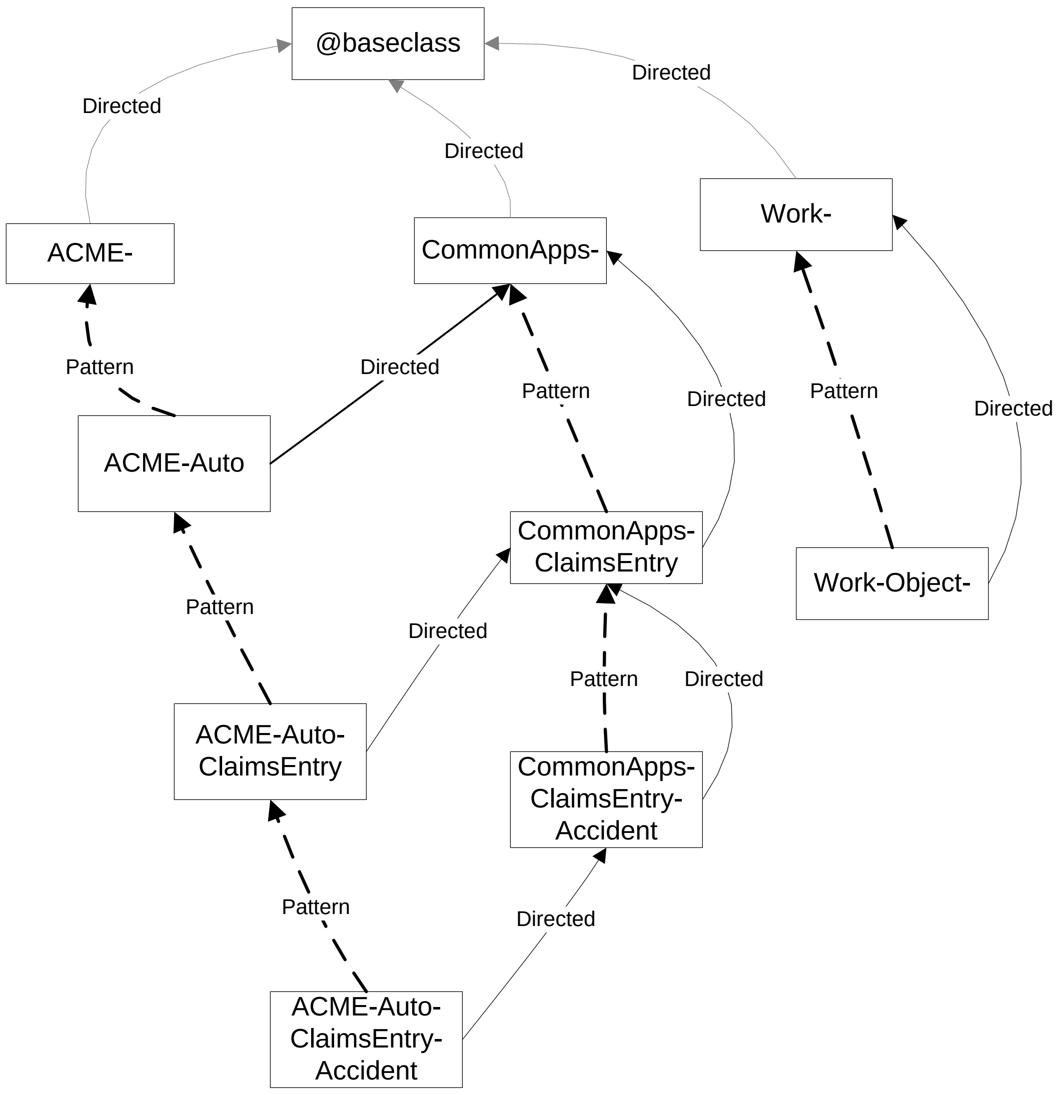
Thus, for the class pictured earlier (PegaSample-Folder), if searching for a specific rule ("Display") defined on that class, the system would check the current class, and then follow pattern inheritance to PegaSample. If the activity were not found there, the system would follow the directed inheritance link and check Work-Folder-, and then, depending upon the settings in Work-Folder-, follow either pattern or directed inheritance from there.

[1] For almost every class, Pattern inheritance should remain enabled (checked). A possible exception might be a situation where the hierarchy was built incorrectly, and now inheritance must be forced into a certain path.

Example: Inheritance paths

The following classes are in the system:

ACME-
Acme-Auto
Acme-Auto-ClaimsEntry
Acme-Auto-ClaimsEntry-Accident
CommonApps-
CommonApps-ClaimsEntry
CommonApps-ClaimsEntry-Accident
Work-
Work-Object-
@baseclass



Inheritance is set up for the above diagram by checking inheritance boxes on Class forms:

Class	**On Class Form:** ***Find by Name Pattern First*** **checked (pattern inheritance) Most Immediate Ancestor:**	***Inherit From (Parent)*** **field entry (directed inheritance) Most Immediate Ancestor:**
Acme-Auto-ClaimsEntry-Accident	Acme-Auto-ClaimsEntry	CommonApps-ClaimsEntry-Accident
Acme-Auto-ClaimsEntry	Acme-Auto	CommonApps-ClaimsEntry
Acme-Auto	Acme-	CommonApps-
Acme-		*@baseclass*
CommonApps-ClaimsEntry-Accident	CommonApps-ClaimsEntry	CommonApps-ClaimsEntry
CommonApps-ClaimsEntry	CommonApps-	CommonApps-
CommonApps-		*@baseclass*
Work-Object	Work-	Work-
Work-		*@baseclass*

In the above setup, the user wishes to use the Activity **Display.** There are several paths that rule resolution for this Activity could take, depending upon what inheritance options are chosen.

NOTE: All of the following scenarios assume the following points:

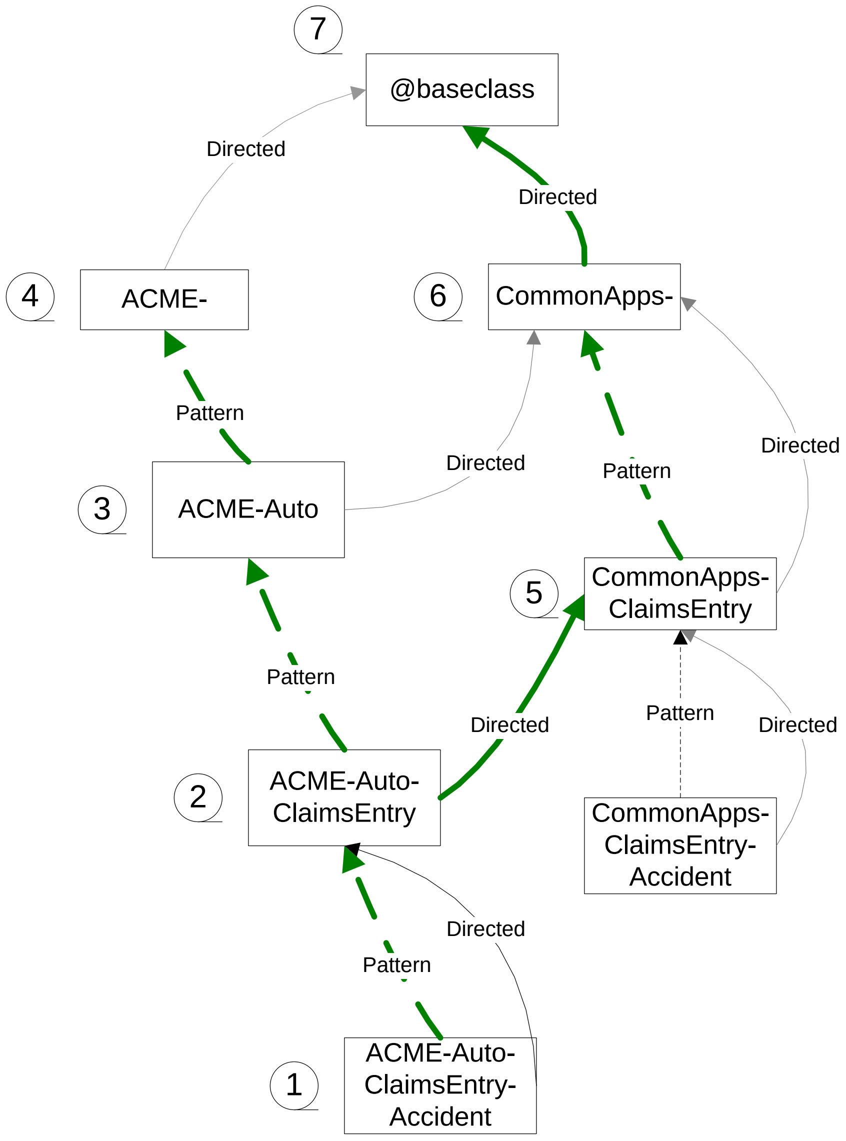
- that Pattern Inheritance on each class is *enabled* (the “Find by name first” checkbox is checked)
- that Pattern Inheritance for each class (the “Find by name first” checkbox on the class form) is shown by the dashed arrows
- that Directed Inheritance for each class (the “Parent Class to Inherit From” field on the class form) is shown by the solid-line arrows

IMPORTANT: The class hierarchies for these examples are meant to illustrate inheritance paths *only* - *not necessarily PegaRULES Best Practice for setting up class hierarchies.* They are set up to show consequences of inheritance choices *only.*

Scenario 1

If the class being requested is Acme-Auto-ClaimsEntry-Accident**,** and the pattern and direct inheritance for this class both point to Acme-Auto-ClaimsEntry, the system will search for **Display** in the following order:

1. Acme-Auto-ClaimsEntry-Accident
2. Acme-Auto-ClaimsEntry
3. Acme-Auto
4. Acme-
5. CommonApps-ClaimsEntry
6. CommonApps-
7. @baseclass



NOTE: Since the class **Acme-Auto-ClaimsEntry** was checked by pattern inheritance after **Acme-Auto-ClaimsEntry-Accident**, when following the directed link, the system knows it has already checked this class. Therefore, the first directed link goes to **CommonApps-ClaimsEntry.**

As stated above, all classes end with directed inheritance to @baseclass, so after checking Acme- and then following directed inheritance from Acme-Auto-ClaimsEntry up through CommonApps-, the last class checked in this pattern is @baseclass.

All the above rules in any available RuleSet and Version will be selected to begin Rule Resolution.

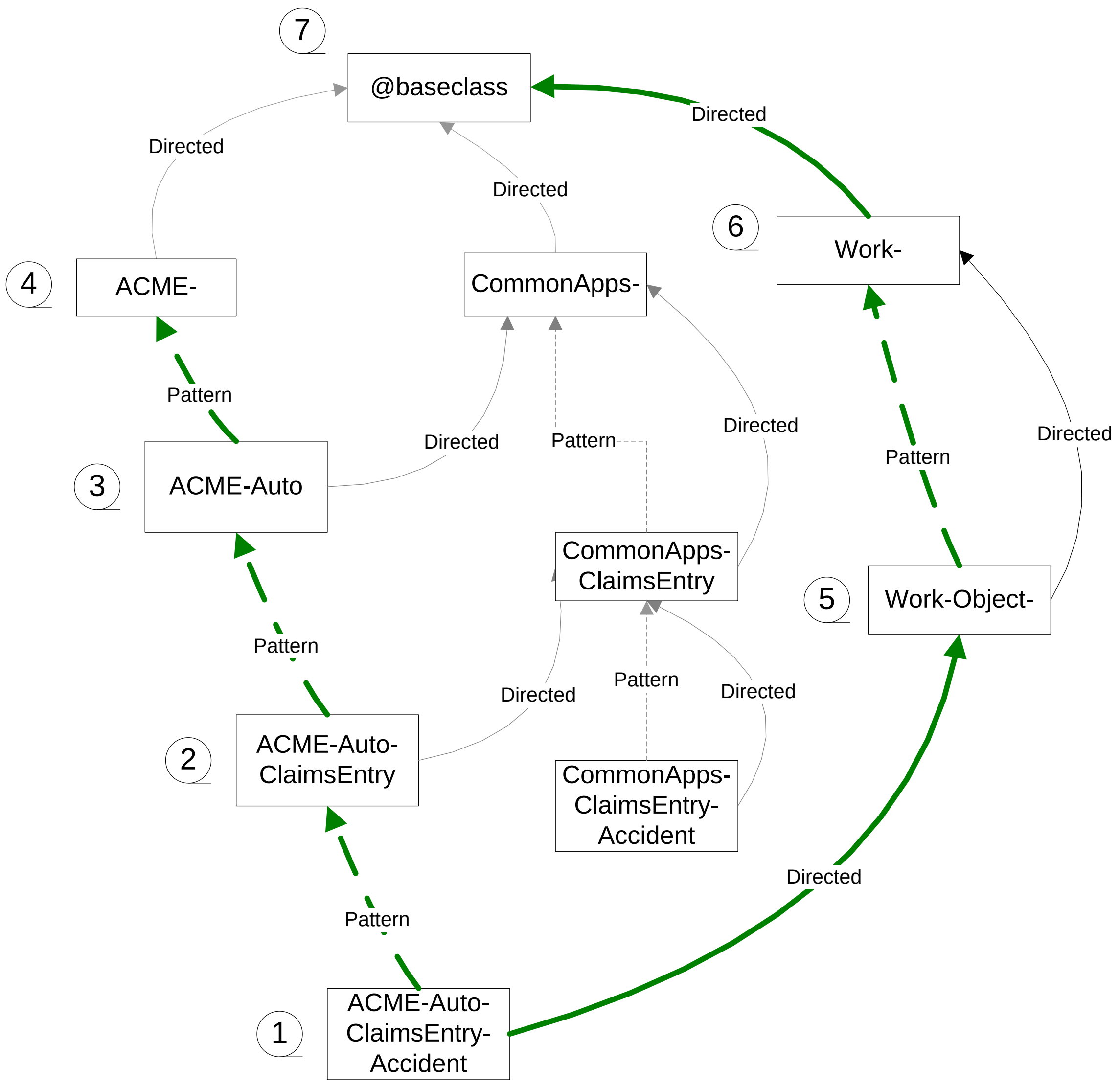
Important: Rule Resolution points to the class closest (in the inheritance path) to the one that is being requested (Acme-Auto-ClaimsEntry) before checking the more general class (Acme-). As soon as the system finds a match for the rule it is searching for, it stops searching and uses the match it has found.

Scenario 2

The example has changed for this scenario – as part of the application development, the developer decided they needed to inherit from Work-, so they have changed directed inheritance for the class **Acme-Auto-ClaimsEntry-Accident** to point to **Work-Object-.**

Due to this change, the first directed link becomes the one from **Acme-Auto-ClaimsEntry-Accident,** up to **Work-Object-.** This link will be followed after all the classes are checked using pattern inheritance. Therefore, the system will search for **Display** in the following order:

1. Acme-Auto-ClaimsEntry-Accident
2. Acme-Auto-ClaimsEntry
3. Acme-Auto
4. Acme-
5. Work-Object-
6. Work-
7. @baseclass



Realize that inheritance can be changed by adding new classes, or by changing the directed inheritance of existing classes. In the first scenario, the system is looking for the Activity **Display** and may find it in the **CommonApps-** class. However, if the inheritance lower in the chain is changed, then CommonApps- isn't even checked.

Since only *the first* directed link off each pattern chain is followed, the CommonApps classes are not reached. Since Acme-Auto-ClaimsEntry-Accident now has a directed link to Work-Object-, that first directed inheritance link is followed, and *no other directed link off the Acme pattern chain is followed.* (Although there are directed links to CommonApps-ClaimsEntry and to CommonApps-, they are not the first directed link, and so are not followed.)

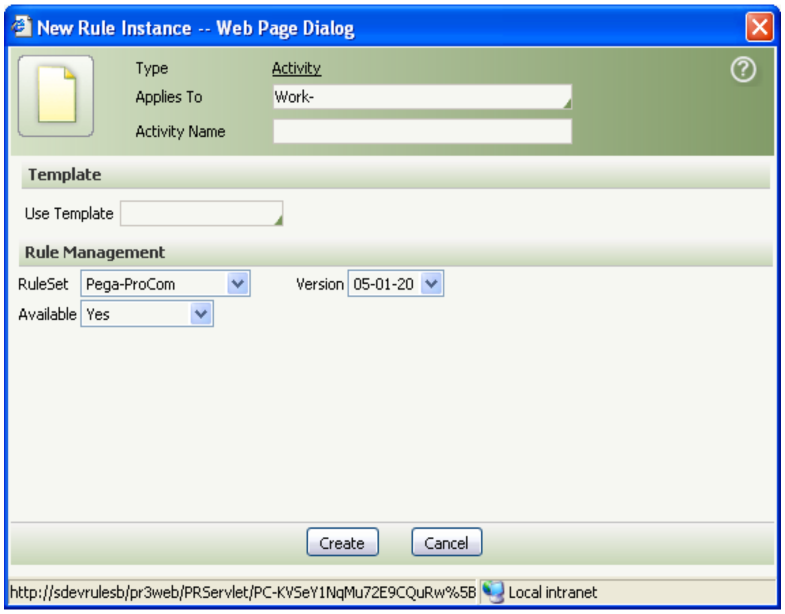
If the required version of **Display** is defined on **CommonApps-**, the system will not find that version. The class **CommonApps-** is "blocked" by **Work-Object-**; any items which were inherited from CommonApps- by Acme-Auto-ClaimsEntry-Accident *will no longer be available.* Instead, items inherited from Work-Object- and Work- will be available.

Rule Resolution Qualifiers

There are several different qualifiers that differentiate between same-named Rules, including:

- Circumstance
- Circumstance Date
- Multivariate Circumstancing
- Date/Time Ranges

When a new Rule is being created, the RuleSet and Version must be selected from the dropdown boxes displayed on the screen. The first definition of a Rule with an entirely new name (creating a New Rule) will not have the qualifier fields:


New Rule Instance -- Web Page Dialog

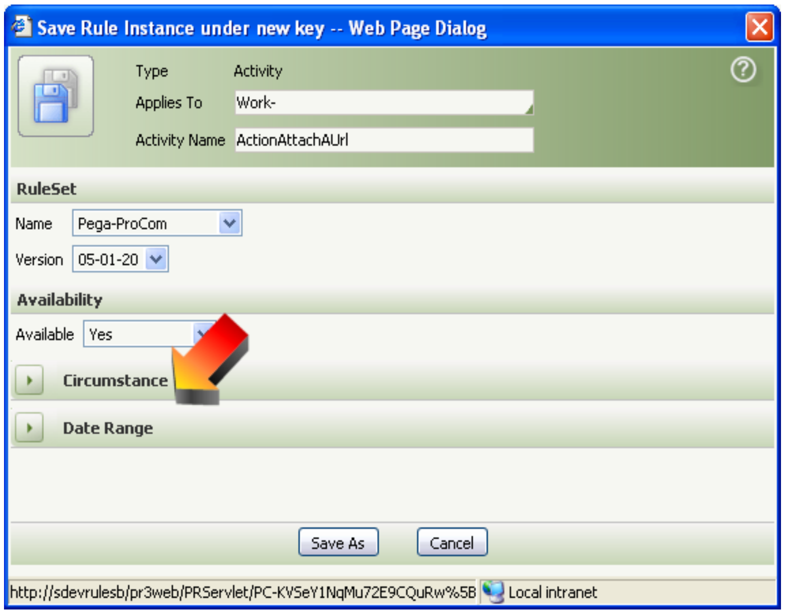
Type Activity
Applies To Work-
Activity Name

Template
Use Template

Rule Management
RuleSet Pega-ProCom
Version 05-01-20
Available Yes

http://sdevrulesb/pr3web/PRServlet/PC-KVSeY1NqMu72E9CQuRw%5B Local intranet

Once a Rule has been created and saved, **Save As** should be used to create a rule with that same name, in order to add desired qualifiers.



Save Rule Instance under new key -- Web Page Dialog

Type Activity

Applies To Work-

Activity Name ActionAttachAUrl

RuleSet

Name Pega-ProCom

Version 05-01-20

Availability

Available Yes

Circumstance
Date Range
Save As
Cancel
http://sdevrulesb/pr3web/PRServlet/PC-KVSeY1NqMu72E9CQuRw%5B
Local intranet

The **Circumstance** and **Date Range** fields will only display on the **Save As** form. Using the **New** form, even if an existing Rule is filled in for the *Name* field, will not allow the user to enter any qualifier information.

This functionality reinforces the fact that if there is a rule (or several rules) with a given name which has one or more qualifiers, then there *must* be at least one default rule of that name present with *no* qualifier, in case none of the qualified rules are appropriate for the run-time situation.

This default rule with no qualifier can be:

- in the *same* RuleSet as the qualified rule
- in a RuleSet which is a *prerequisite* of the RuleSet of the qualified rule

In addition, the default (unqualified) rule may be defined:

- on the *same class* as the qualified rule
- on an *ancestor of the class* of the qualified rule

Single Circumstance

A Circumstance may be added to a Rule by creating a circumstance *property* ("CustomerStatus") and then adding the name of that property and a value for that property to the Rule definition. Circumstances represent facts about customers. They allow the user to specify that different Rules should be used for different types of customers, where the type of customer may be identified by the value of the Circumstance property.

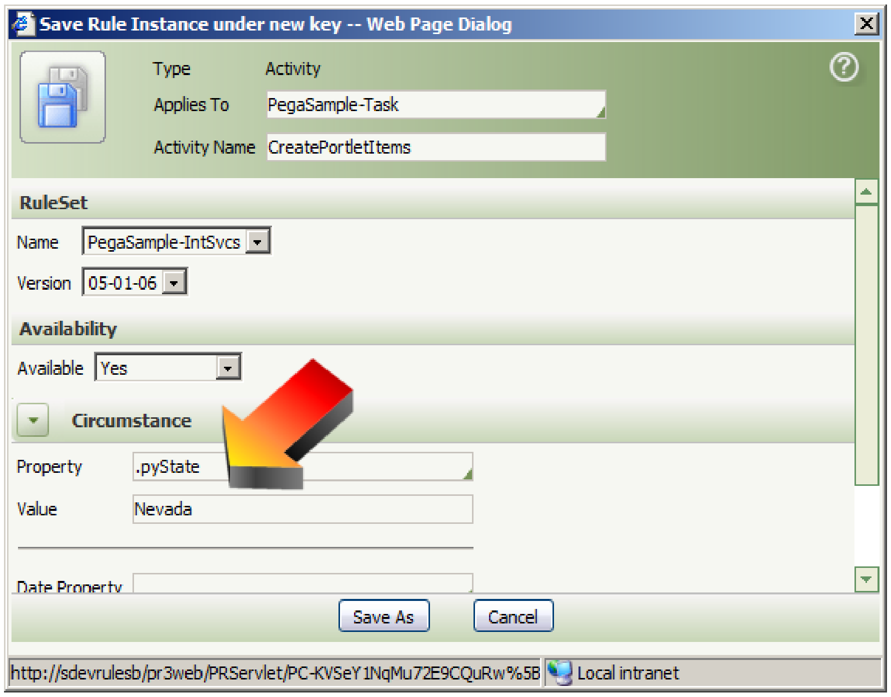
For example, the "StateTax" Rule might be defined to calculate a state tax. There are 50 states:

- 35 of them use a 5% tax rate
- 15 states have tax rates at a value other than 5%

In this situation, the default Rule is defined to calculate a 5% tax, as the majority of the states have that value. A Circumstance property called *pyState* may be defined for this rule; each of the 15 States with a different tax value would have a special Circumstanced rule to replace the default 5% rate:

Circumstance Property	Circumstance Value	Tax Rate (set in rule)
pyState	New Hampshire	2%
	New York	8%
	California	12%
etc.		

When the customer creates an Invoice work item, the activity calculating the invoice pricing would call the StateTax rule. Based on the state where the buyer resides (which is identified in the work item), Rule Resolution would choose either the default Rule or one of the Circumstanced Rules.


Save Rule Instance under new key -- Web Page Dialog

Type Activity
Applies To PegaSample-Task
Activity Name CreatePortletItems

RuleSet
Name PegaSample-IntSvcs
Version 05-01-06

Availability
Available Yes

Circumstance
Property .pyState
Value Nevada

Date Property

Save As
Cancel
http://sdevrulesb/pr3web/PRServlet/PC-KVSeY1NqMu72E9CQuRw%5B
Local intranet

If there are rules with a given name that specify a Circumstance, then there *must* be at least one rule with that name with *no* circumstance property specified, in case none of the Circumstance-qualified rules are appropriate for the run-time situation. This non-circumstanced rule is called the “default” rule. The default rule must be defined:

- on the same class as the circumstanced rules, or on an ancestor of that class
- in the same RuleSet version as the circumstanced rules, or in one of that RuleSet’s prerequisites

A valid circumstanced rule will take precedence over the default rule (class, RuleSet, and Version being equal).

The Circumstance **Property** field (.pyCircumstanceProp) holds the property name for the Circumstance Property (example: .pyStateName). This property can be an absolute or a relative reference, which will be looked for in the context of the current Primary Page[2], and must hold only one value at a time; aggregate properties are not allowed.

The Circumstance **Value** field (.pyCircumstanceVal) holds the specific value that the Circumstance Property must contain for this rule to be selected. (Example: "Nevada")

For rules such as Activities, the Circumstance Property is usually defined on the Class on which the rule is defined; it can also be defined on named pages of other classes.

Valid Entries for Circumstance Property	Examples
(relative reference) property name	.CustomerStatus
fully qualified property name	myPage.CustomerStatus
embedded page with property name	.embeddedPage.CustomerStatus
fully qualified name with page and embedded page	myPage.embeddedPage.CustomerStatus
embedded pagelist or pagegroup with one value specified	.embeddedPage(7).CustomerStatus .embeddedPage(invoice).CustomerStatus

list or group property with one value specified	.CustomerStatus(3)
	.CustomerStatus(Delaware)

If any rule of a given name has a Circumstance Property defined, then all the rules *with that name, across all classes*[3] must share that same Circumstance Property - except for the defaults, which have *no* Circumstance.

Note that it is possible to have more than one non-Circumstance-qualified Rule with the same name in a class hierarchy. If the Activity **Display** is defined on the class Acme-Auto-ClaimsEntry and also defined differently on the class Acme-Auto, then there could be a default rule defined with no Circumstance on Acme-Auto, and again on Acme-Auto-ClaimsEntry; or there could be multiple defaults defined on Acme-Auto-ClaimsEntry with different date ranges.

[2] NOTE: There is a special case, where an Activity is called from within another Activity. In this case, the Primary Page chagnges to th e page on whith the Activity is called. This change occurs *before* the Activity is looked up, so that its circumstance qualification can be checked against the newly-established Primary Page (rather than the initial Primary Page). All other rule-resolved rules with circumstances have those circumstances looked up on the Primary Page which was established by the calling activity - so that a Stream displayed from an Activity uses the Activity's Primary Page, and a Model executed by an Activity also uses that Activity's Primary Page.

[3] There may be two classes that are *not* in the same inheritance tree which have the same-named rule, and those rules may not be in any way related. However, it is not possible for the system to determine whether at some point, a change may be made to the system to link these same-named rules and *make* them related; therefore, all rules of the same name – whether currently in the same inheritance tree or not – must have the same Circumstance Property defined.

Different versions of a named rule may have different Circumstance Property *values*, but they must have the same Circumstance Property. So for the **CreatePortletItems** Activity defined on PegaSample-Task, if the property **CustomerStatus** was entered into the **Circumstance Property** field, then all other **CreatePortletItems** activities defined on PegaSample-Task with a Circumstance *must* have only **CustomerStatus** in that field. (A blank value – i.e., no property and no values filled in - for Circumstance would also be correct.) There can be different values filled in (“Platinum”, “Gold”, “Silver”, etc.), but they can only use that one property.

In order to *change* the Circumstance Property from one property to another, all rules with that Circumstance Property must be deleted; then they may be recreated with the new Circumstance Property.

At run-time, when doing Rule Resolution of Circumstances, the system checks the primary page (of the activity being executed) for the Circumstance Property (or, for a fully-qualified property name, the specified page).

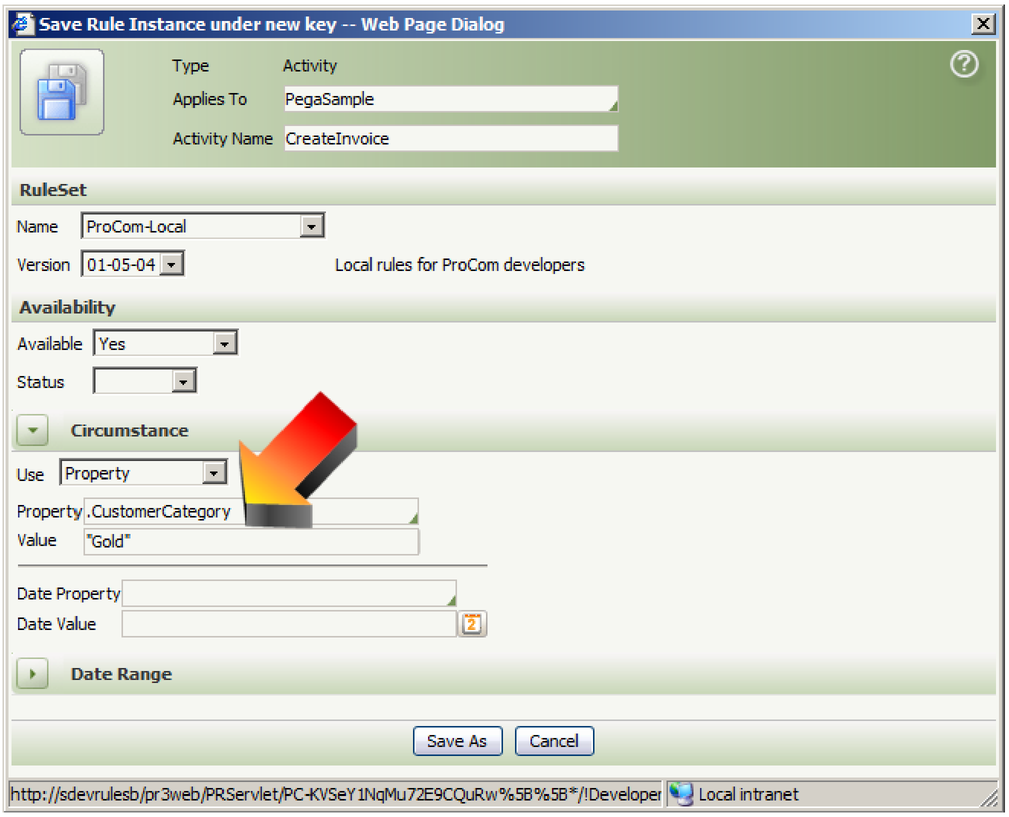
- If the Circumstance Property is blank or does not exist on the page for the activity (or, for a fully-qualified property name, if the property is not on the specified page in the property name), then the system will *ignore* all the Circumstanced rules and choose one of the default (non-Circumstanced) rules.

- If the property *does* exist on the page in question, then the system must evaluate the value of that property and compare it to the Circumstance Values for each rule being considered by Rule Resolution.

Example: Circumstance property

Circumstance property: CustomerCategory

Possible values:	“Platinum”
	“Gold”
	“Silver”
	“Bronze”


Save Rule Instance under new key -- Web Page Dialog

Type Activity
Applies To PegaSample
Activity Name CreateInvoice

RuleSet
Name ProCom-Local
Version 01-05-04 Local rules for ProCom developers

Availability
Available Yes
Status

Circumstance
Use Property
Property .CustomerCategory
Value "Gold"

Date Property
Date Value

Date Range

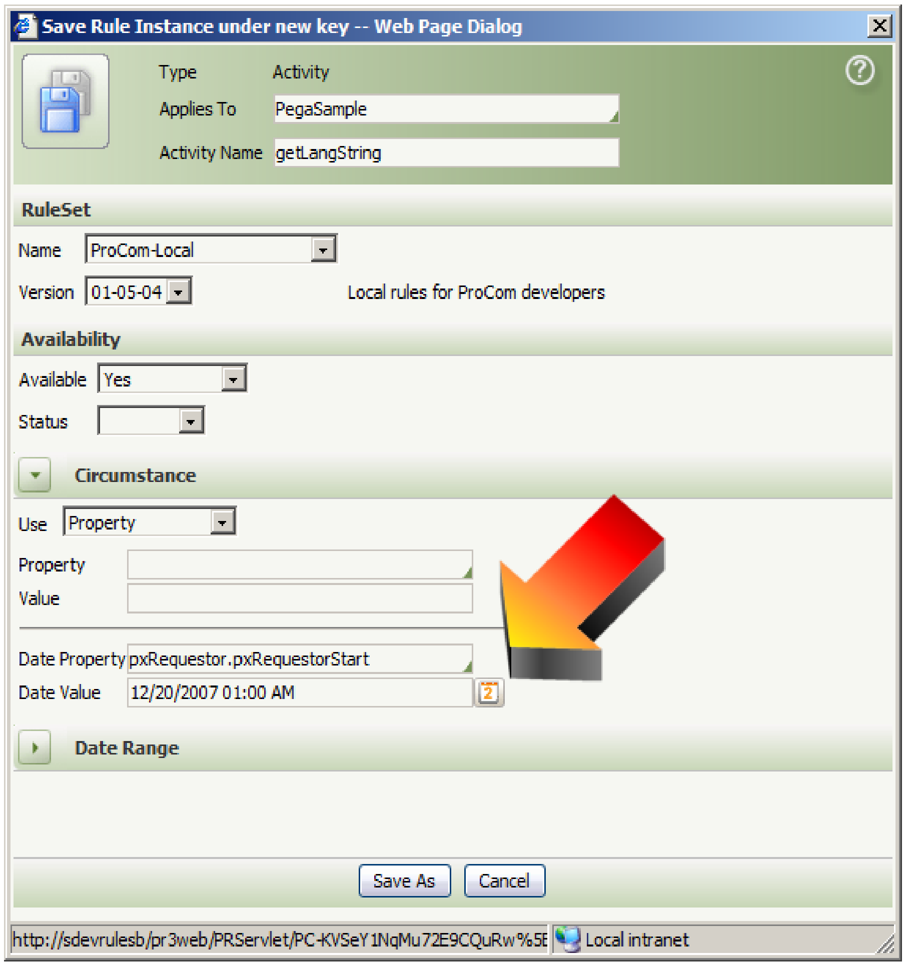
Save As Cancel

http://sdevrulesb/pr3web/PRServlet/PC-KVSeY1NqMu72E9CQuRw%5B%5B*/!Developer Local intranet

For the above example rule, the Circumstance Property (CustomerCategory) and the value (Gold) are specified. If when the Rule Resolution is executed, the current value of CustomerCategory on the Primary Page is “Silver,” this rule would not be selected, as the CustomerCategory of this rule (“Gold”) does not match the value of the Circumstance Property (“Silver”).

Circumstance Date

A Circumstance Date is a qualifier much like a Circumstance. A Circumstance Date may be added to a rule by creating or identifying a Circumstance Date *property* and then entering the name of that property and a value for that property to the rule definition.



Save Rule Instance under new key -- Web Page Dialog

Type: Activity
Applies To: PegaSample
Activity Name: getLangString

RuleSet
Name: ProCom-Local
Version: 01-05-04
Local rules for ProCom developers

Availability
Available: Yes
Status:

Circumstance
Use: Property
Property:
Value:
Date Property: pxRequestor.pxRequestorStart
Date Value: 12/20/2007 01:00 AM

Date Range

Save As
Cancel
http://sdevrulesb/pr3web/PRServlet/PC-KVSeY1NqMu72E9CQuRw%5E
Local intranet

If there are rules with a given name that specify a Circumstance Date, then there *must* be at least one rule with that name with *no* Circumstance Date property, in case none of the Circumstance Date-qualified rules are appropriate for the run-time situation. This non-circumstanced rule is called the "default" rule. The default rule must be defined:

- on the same class as the circumstanced rules, or on an ancestor of that class
- in the same RuleSet version as the circumstanced rules, or in one of that RuleSet's prerequisites

A valid circumstanced rule will take precedence over the default rule (class, RuleSet, and Version being equal).

The Circumstance **Date Property** field (.pyCircumstanceDateProp) holds the name of the property on the clipboard that contains the date/time to which the Circumstance Date value is compared (example: .StartDate). The Circumstance Date Property can be an absolute or a relative reference, which will be looked for in the context of the Primary Page of the Activity currently being executed[4], and must hold only one value at a time; aggregate properties are not allowed.

The Circumstance **Date Value** field (.pyCircumstanceDate) holds the specific value to which the value of the Circumstance Date Property must be compared.

The date value entered into the **Date Value** field is a constant, and could be thought of as a *starting date.* Rules containing a Circumstance Date value closest to (but not after) the value in pyCircumstanceDate (stored on the clipboard) will be ranked superior in the list to rules with Circumstance Date values that are further in the past. (See Example 3 below for an illustration of this point.)

For rules such as Activities, the Circumstance Date Property is usually defined on the Class on which the rule is defined (Work- in the above example); it can also be defined on named pages of other classes.

Valid Entries for Circumstance Date Property	Example
(relative reference) property name	.CreateDate
fully qualified property name	myPage.CreateDate
embedded page with property name	.embeddedPage.CreateDate
fully qualified name with page and embedded page	myPage.embeddedPage.CreateDate
embedded pagelist or pagegroup with one value specified	.embeddedPage(7).CreateDate .embeddedPage(invoice).CreateDate

list or group property with one value specified	.CreateDate(3)
	.CreateDate("CustAccountOpened")

If any rule of a given name has a Circumstance Date Property defined, then all the rules *with that name* must share that same Circumstance Date Property - except for the defaults, which have *no* Circumstance Date.

Note that it is possible to have more than one non-Circumstance-qualified rule with the same name in a class hierarchy. If the Activity **Display** is defined on the class Acme-Auto-ClaimsEntry, and also defined differently on the class Acme-Auto, then there could be a default rule defined with no Circumstance Date on Acme-Auto, and again on Acme-

[4] NOTE: There is a special case, where an Activity is called from within an Activity. In this case, the option of changing the Primary Page to the step page is available. This change occurs *before* the Activity is looked up, so that its circumstance qualification can be checked against the newly-established Primary Page (rather than the initial Primary Page). All other rule-resolved rules with circumstances have those circumstances looked up on the Primary Page which was established by the calling activity - so that a Stream displayed from an Activity uses the Activity's Primary Page, and a Model executed by an Activity also uses that Activity's Primary Page.

Auto-ClaimsEntry; or there could be multiple defaults defined on Acme-Auto-ClaimsEntry with different date ranges.

Different versions of a named rule may have different Circumstance Date Property *values*, but they must have the same Circumstance Date property. So for the **ValidateProperties** Activity defined on Data-Admin-Operator-AccessGroup, if the property **AccountCreateDate** was entered into the **Circumstance Date Property** field, then all other **ValidateProperties** activities defined on Data-Admin-Operator-AccessGroup with a Circumstance Date *must* have only **AccountCreateDate** in that field. (A blank value – i.e., no property and no values filled in - for Circumstance Date would also be correct.) There can be different values filled in ("01/01/2002", "06/03/2000 12:30:24", "01/01/1980", etc.), but they can only use that one property.

In order to *change* the Circumstance Date Property from one property to another, all rules with that Circumstance Date Property must be deleted; then they may be recreated with the new Circumstance Date Property.

At run-time, when doing Rule Resolution of Circumstance Dates, the system checks the primary page (of the activity being executed) for the Circumstance Date Property (or, for a fully-qualified property name, the specified page).

- If this property does not exist on the page for the activity (or, for a fully-qualified property name, if the property is not on the specified page in the property name), then the system will *ignore* all the Circumstance Date rules and choose one of the **default** (non-Circumstance-Dated) rules.

- If the property *does* exist on the page in question, then the system must evaluate the value of that property and compare it to the Circumstance Date Value for each rule being considered by Rule Resolution.

NOTE: If the value of the property on the clipboard is *not* a valid Date/Time value, then the system will report an error.

Example: Circumstance Date Property implementation

Scenario 1: Rule Changes After the Circumstance Date Value

The customer, an insurance company, has a default Rule that is used in a flow that processes home insurance claims. New regulations were passed after Hurricane Katrina, requiring homeowners whose homes are in a flood zone to carry flood insurance. All policies written *before* the date of this regulation (01/01/2006) are not required to have the flood insurance, but all policies *after* that date must carry flood protection. Rather than creating two different rules, the customer wishes use a circumstanced version of this rule. They would set up a copy of this rule using Circumstance Date:

Circumstance Date Property:	CustPage.AccountCreateDate
Circumstance Date value:	01/01/06[5]

[5] NOTE: As shown in the Web Page Dialog box above, the Date Value is generally a date/timestamp, including a date *and a time.* For ease of reading in this tech note, circumstance date values will be shown as dates; in a production system, the time should also be specified.

Each time the flow is executed, the system would compare the homeowner's account creation date with the value in Circumstance Date:

- all homeowners whose account was opened (property value of .AccountCreateDate) *after* 01/01/06 would use the Rule qualified by the Circumstance Date

- all homeowners whose account was opened *before* 01/01/06 would use the default rule

Scenario 2: Rule Changes Before the Circumstance Date Value

Sometimes a company has to change a rule going back into the past. For example, another customer, an auto insurance company, has a default rule that is used in a flow to process accident claims; this rule states that the insurance cost for the auto is 10% of its "Blue Book" value. (Don't we wish!)

Suddenly, a federal report comes out stating that cars without a certain type of side air bag (put into wide use in January 2000) are 50% more likely to have fatal crashes. Congress swings into action, and passes a law which states: All new cars which were sold *prior to* January 1, 2000, must pay 5% extra insurance.

Since the Circumstance Date is a *starting* date, the original 10% rule cannot be circumstanced with a 1/1/2000 date, because then *all* cars will have to pay the new higher 15% rate. Therefore, in this scenario, the customer would have to set up a Circumstance Date version of the original rule with the 10% rate, that would be an *exact copy* of the original behavior.

Circumstance Date Property:	CustPage.DateCarPurchased
Circumstance Date value:	01/01/2000

Then the customer would have to change the *default* (unCircumstance-Dated) rule to reflect the new 15% rate. The result:

- car owners who purchased their new car *after* 01/01/2000 use the Circumstance Date rule with the 10% rate (previously the default rule)
- car owners who purchased their new car *before* 01/01/2000 use the (new) default rule with the 15% rate

Thus, the Circumstance-Dated Rule contains the "standard" behavior, and the default Rule becomes the "exception" case.

Multivariate Circumstancing

Beginning in Version 5.4, rules may be circumstanced by one property, or by multiple properties. If a rule is circumstanced by one property, the name of that property and a property value for that rule would be defined when the circumstanced rule is created (as described above).

However, customers may need to circumstance a rule by more than one property. As an example, an insurance company has to meet various state regulations regarding communication about privacy notices and other customer disclaimers. The laws vary by state, and may require the insurance company to communicate different disclaimers depending upon the amount of the claim and the type of customer communication (phone call, email, letter).

In order to allow multiple circumstance values to be evaluated while choosing rules, two different rule types must be configured:

- Circumstance Templates (Rule-Circumstance-Template)
- Circumstance Definitions (Rule-Circumstance-Definition)

NOTES:

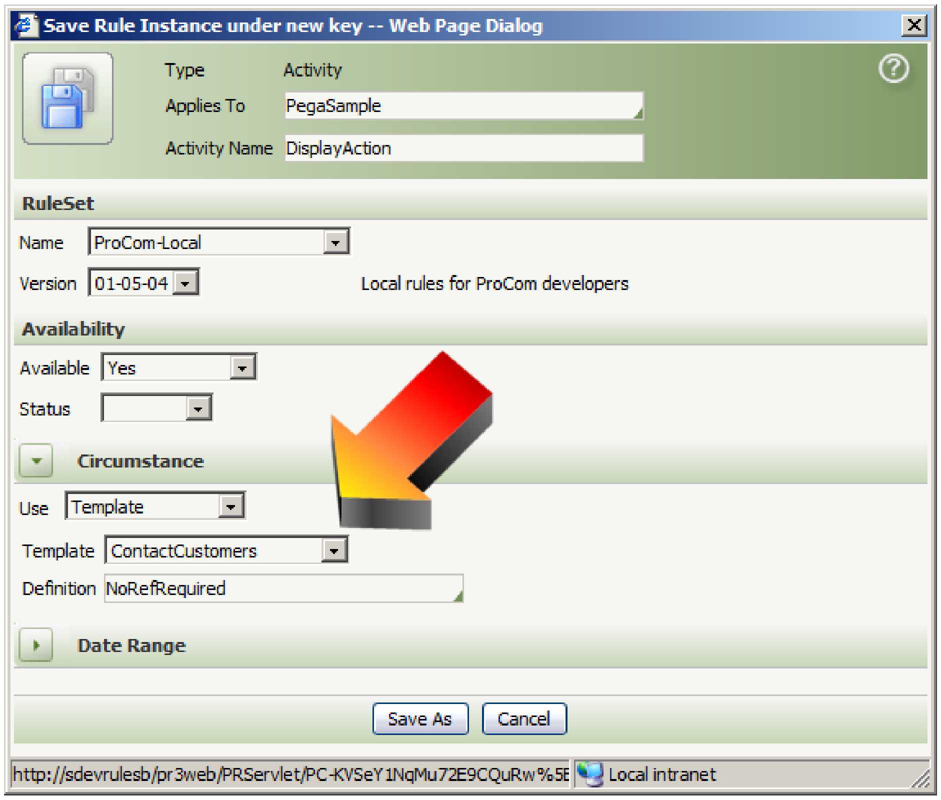
- Do not confuse the Circumstance Template rule with a *template* rule. Template rules may be used as a pattern to copy for creating new rules; a Circumstance Template rule is a template specifically designed to implement the multiple-circumstance-property functionality.

- Circumstance Template and Circumstance Definition rules are not circumstanceable!

Please see the KB article #25272, *How Multivariate Circumstancing Works,* for details on this functionality.

Property information for multi-circumstanced rules

When doing a Save As of a rule qualified by multiple circumstance, choose *Template* in the **Use** drop-down field, and then enter the Template Name and Definition Name.


Save Rule Instance under new key -- Web Page Dialog

Type Activity
Applies To PegaSample
Activity Name DisplayAction

RuleSet
Name ProCom-Local
Version 01-05-04 Local rules for ProCom developers

Availability
Available Yes
Status

Circumstance
Use Template
Template ContactCustomers
Definition NoRefRequired

Date Range

Save As Cancel

http://sdevrulesb/pr3web/PRServlet/PC-KVSeY1NqMu72E9CQuRw%5E Local intranet

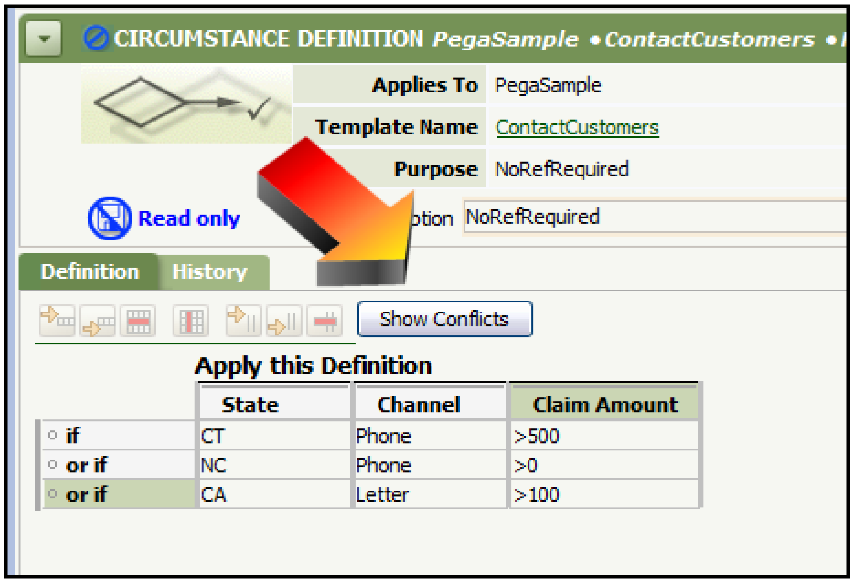
The property names for these fields are as follows:

Field Name	Property Name	Example Value
Use	.pyCircumstanceType	Template
Template	.pyCircumstanceProp	*ContactCustomers* (Rule-Circumstance-Template instance)
Definition	.pyCircumstanceVal	*NoRefRequired* (Rule-Circumstance-Definition instance)

Note that these fields use the same property names for a rule with a single circumstance value; the only difference is that .pyCircumstanceType would be set to "Property."

Creating the combined decision table

As stated in the KB article *How Multivariate Circumstancing Works,* the **Show conflicts** button is used to combine all the Circumstance Definitions together, and highlight any incorrect configurations.   This button is available on the **Definitions** tab of the Circumstance Template, and also the **Definition** tab of each Circumstance Definition rule.

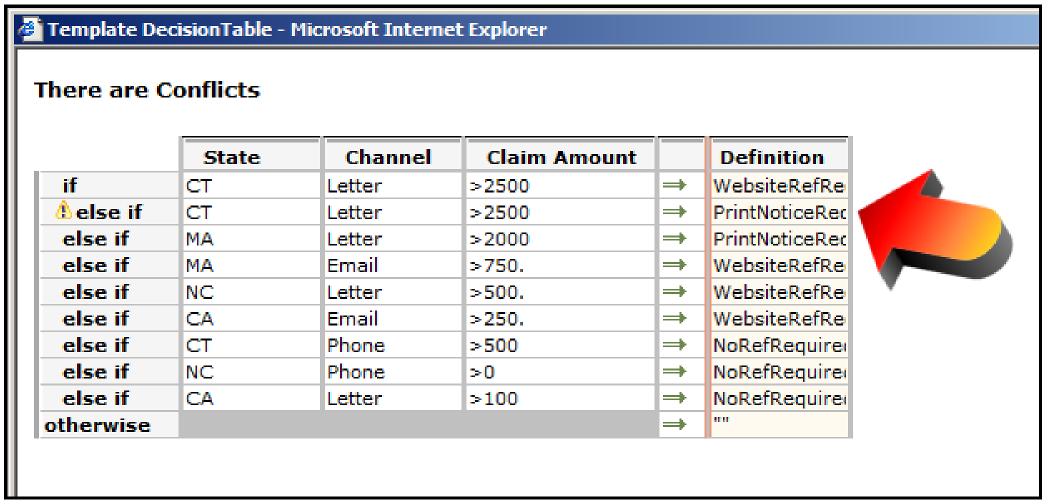


Applies To: PegaSample
Template Name: ContactCustomers
Purpose: NoRefRequired
ption: NoRefRequired

Apply this Definition

	State	Channel	Claim Amount
if	CT	Phone	>500
or if	NC	Phone	>0
or if	CA	Letter	>100

Clicking this button from the Template rule or *any* of the Definition rules will display the combined decision table for the entire Template, and show any conflicts.


Template DecisionTable - Microsoft Internet Explorer

There are Conflicts

	State	Channel	Claim Amount		Definition
if	CT	Letter	>2500	⇒	WebsiteRefRe
⚠else if	CT	Letter	>2500	⇒	PrintNoticeRec
else if	MA	Letter	>2000	⇒	PrintNoticeRec
else if	MA	Email	>750.	⇒	WebsiteRefRe
else if	NC	Letter	>500.	⇒	WebsiteRefRe
else if	CA	Email	>250.	⇒	WebsiteRefRe
else if	CT	Phone	>500	⇒	NoRefRequire
else if	NC	Phone	>0	⇒	NoRefRequire
else if	CA	Letter	>100	⇒	NoRefRequire
otherwise				⇒	""

The chart states “There are conflicts” and the first *else* line has a “!” icon, showing that this line is where the problem is. The developer must research the cause of this conflict among the Circumstance Definitions and resolve it before the current Circumstance Definition may be saved.

If possible, the system will automatically order the value lines so that there aren’t logic errors. For example, entries in the following order would present a logic error:

State	Channel	ClaimAmount	Circumstance Definition Rule
CT	Letter	>1500	WebNotice
CT	Letter	>2000	PrintNotice

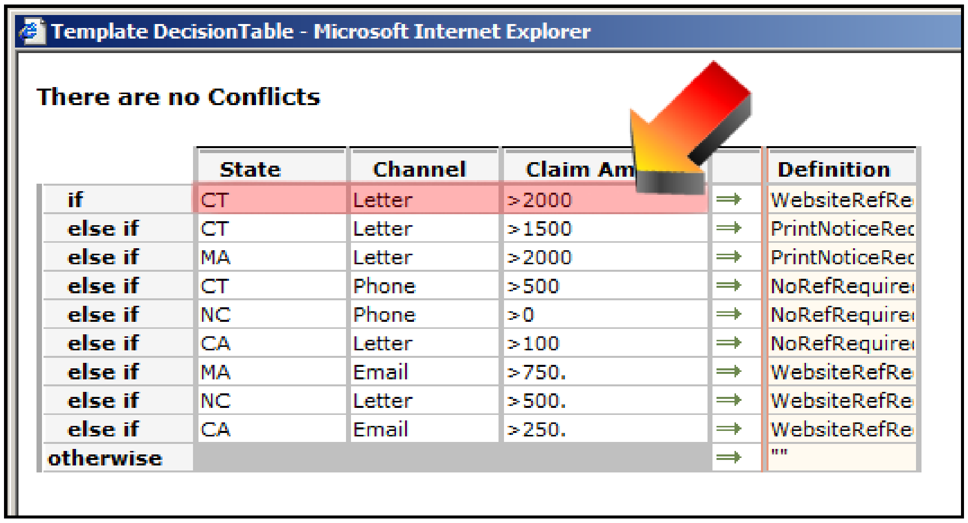
Since anything that would be greater than 2000 would *also* be greater than 1500, the second line would never be reached (i.e., all rules where the claim amount was greater than 1500 would automatically be sent to the web notice). In order to properly handle this situation, the system would reorder these as follows:

State	Channel	ClaimAmount	Circumstance Definition Rule
CT	Letter	>2000	PrintNotice
CT	Letter	>1500	WebNotice

Now, all claims above $2000 will get the print notice, and claims less than $2000 but greater than $1500 get the web notice.

Empty fields are evaluated as “matches all,” and are thus automatically ordered to be below specific data.

For example, the Template Decision Table for the *ContactCustomers* Circumstance Template looks like this:

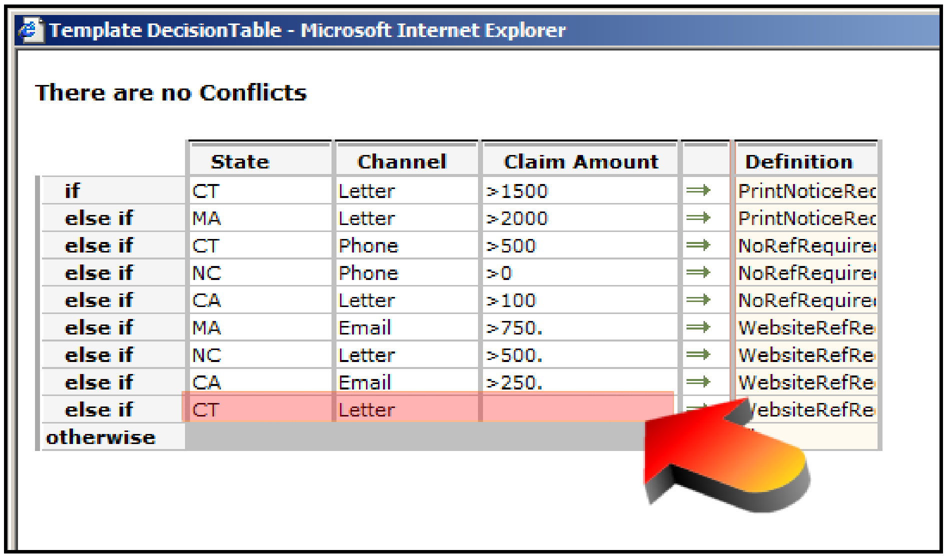


Template DecisionTable - Microsoft Internet Explorer

There are no Conflicts

	State	Channel	Claim Am		Definition
if	CT	Letter	>2000	⇒	WebsiteRefRe
else if	CT	Letter	>1500	⇒	PrintNoticeRec
else if	MA	Letter	>2000	⇒	PrintNoticeRec
else if	CT	Phone	>500	⇒	NoRefRequire
else if	NC	Phone	>0	⇒	NoRefRequire
else if	CA	Letter	>100	⇒	NoRefRequire
else if	MA	Email	>750.	⇒	WebsiteRefRe
else if	NC	Letter	>500.	⇒	WebsiteRefRe
else if	CA	Email	>250.	⇒	WebsiteRefRe
otherwise				⇒	""

If one of the rows in one of the Circumstance Definitions is blanked out, the order of the values in the table changes:



Template DecisionTable - Microsoft Internet Explorer

There are no Conflicts

	State	Channel	Claim Amount		Definition
if	CT	Letter	>1500	⇒	PrintNoticeRec
else if	MA	Letter	>2000	⇒	PrintNoticeRec
else if	CT	Phone	>500	⇒	NoRefRequire
else if	NC	Phone	>0	⇒	NoRefRequire
else if	CA	Letter	>100	⇒	NoRefRequire
else if	MA	Email	>750.	⇒	WebsiteRefRe
else if	NC	Letter	>500.	⇒	WebsiteRefRe
else if	CA	Email	>250.	⇒	WebsiteRefRe
else if	CT	Letter		⇒	ebsiteRefRe
otherwise					

Note that if a Circumstance Definition is created where all the values are blank, this is essentially another base rule. If there is a line in a Circumstance Definition with all blank values, then the base (uncircumstanced) version of this rule will never be executed, because the blank version will “match” all values for those properties.

During the Circumstance Definition save process, the system does automatic conflict resolution by moving empty field rows below field rows with specified values. Then any priority definitions are moved to the top of a virtual decision table which contains *all* values from *all* Circumstance Definition rules associated with the given Circumstance Template. Finally, the system tests for conflicts.

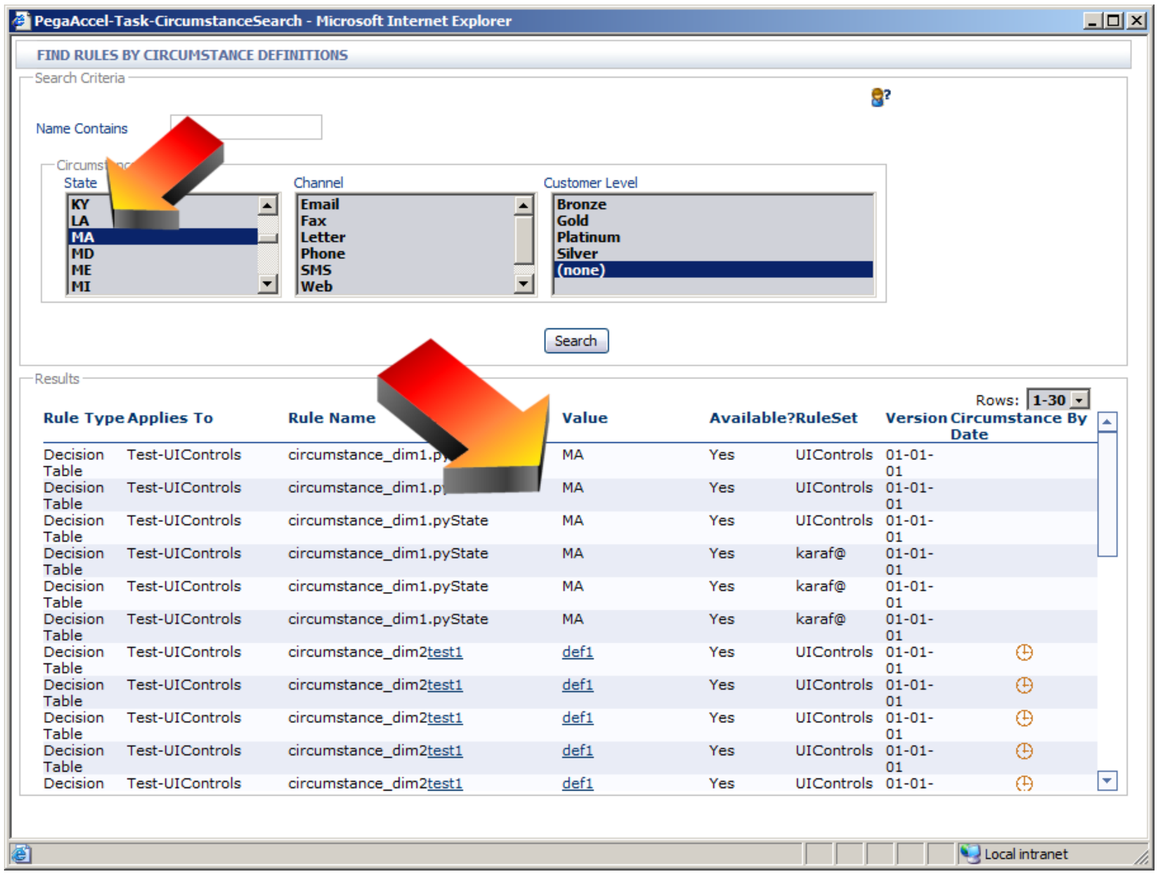
- If there are no conflicts, the system completes the save of this Circumstance Definition.
- If there are *no* priority definitions and there is a conflict, the system displays an error in the ruleform, and the developer must resolve the conflict in order to save the Circumstance Definition.

- If priority definitions are specified, and a conflict occurs, then a warning is displayed; the developer must decide whether they wish to leave the conflict, or redefine the priority definition.

Note that a conflict could span several rows. Property value data in one row might conflict with one row in one way, and another row in another way. Again, it is up to the developer to resolve these conflicts so the application reflects how the company wishes to process their work.

Reporting multivariate circumstancing

As explained in KB #25220 ("How to Report Rules with a Particular Circumstance Value"), the *Find by Circumstance Value* report allows allows users to choose from multiple properties, to print reports showing data on queries like "Show all rules circumstanced for the state of Massachusetts".


PegaAccel-Task-CircumstanceSearch - Microsoft Internet Explorer

FIND RULES BY CIRCUMSTANCE DEFINITIONS

Search Criteria

Name Contains

State	Channel	Customer Level
KY	Email	Bronze
LA	Fax	Gold
MA	Letter	Platinum
MD	Phone	Silver
ME	SMS	(none)
MI	Web	

Search

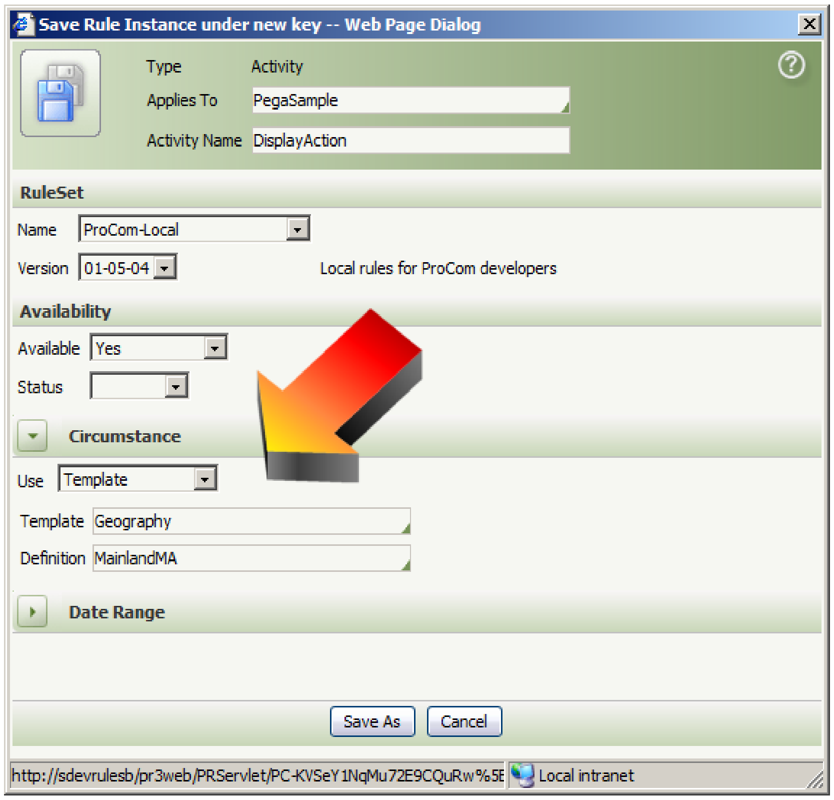
Results

Rows: 1-30

Rule Type	Applies To	Rule Name	Value	Available?	RuleSet	Version	Circumstance By Date
Decision Table	Test-UIControls	circumstance_dim1.p	MA	Yes	UIControls	01-01-01	
Decision Table	Test-UIControls	circumstance_dim1.p	MA	Yes	UIControls	01-01-01	
Decision Table	Test-UIControls	circumstance_dim1.pyState	MA	Yes	UIControls	01-01-01	
Decision Table	Test-UIControls	circumstance_dim1.pyState	MA	Yes	karaf@	01-01-01	
Decision Table	Test-UIControls	circumstance_dim1.pyState	MA	Yes	karaf@	01-01-01	
Decision Table	Test-UIControls	circumstance_dim1.pyState	MA	Yes	karaf@	01-01-01	
Decision Table	Test-UIControls	circumstance_dim2test1	def1	Yes	UIControls	01-01-01	◷
Decision Table	Test-UIControls	circumstance_dim2test1	def1	Yes	UIControls	01-01-01	◷
Decision Table	Test-UIControls	circumstance_dim2test1	def1	Yes	UIControls	01-01-01	◷
Decision Table	Test-UIControls	circumstance_dim2test1	def1	Yes	UIControls	01-01-01	◷
Decision	Test-UIControls	circumstance_dim2test1	def1	Yes	UIControls	01-01-	◷

Local intranet

This report is trickier to produce than it looks from the menu. Unlike single-circumstance rules, which use a Circumstance Property and a single Circumstance Value, Multiple-circumstanced rules are defined by the Circumstance Template and Circumstance Definition:


Save Rule Instance under new key -- Web Page Dialog
Save As
Cancel
http://sdevrulesb/pr3web/PRServlet/PC-KVSeY1NqMu72E9CQuRw%5E
Local intranet

Type: Activity
Applies To: PegaSample
Activity Name: DisplayAction

RuleSet
Name: ProCom-Local
Version: 01-05-04 — Local rules for ProCom developers

Availability
Available: Yes
Status:

Circumstance
Use: Template
Template: Geography
Definition: MainlandMA

Date Range

The values for the properties are defined *in* the Circumstance Definition, not on the rule itself. Therefore, in order to find "all rules circumstanced for Massachusetts," it is necessary to look at those values from a Declare Index.

Whenever a Circumstance Definition is saved, the properties and their values are also saved into the database table **pr_index_circumstance_def** (associated with the class Index-CircumstanceDefinition). Each entry in this table contains the name of the Circumstance Template, the Circumstance Definition, and one line of the values for the Definition properties:

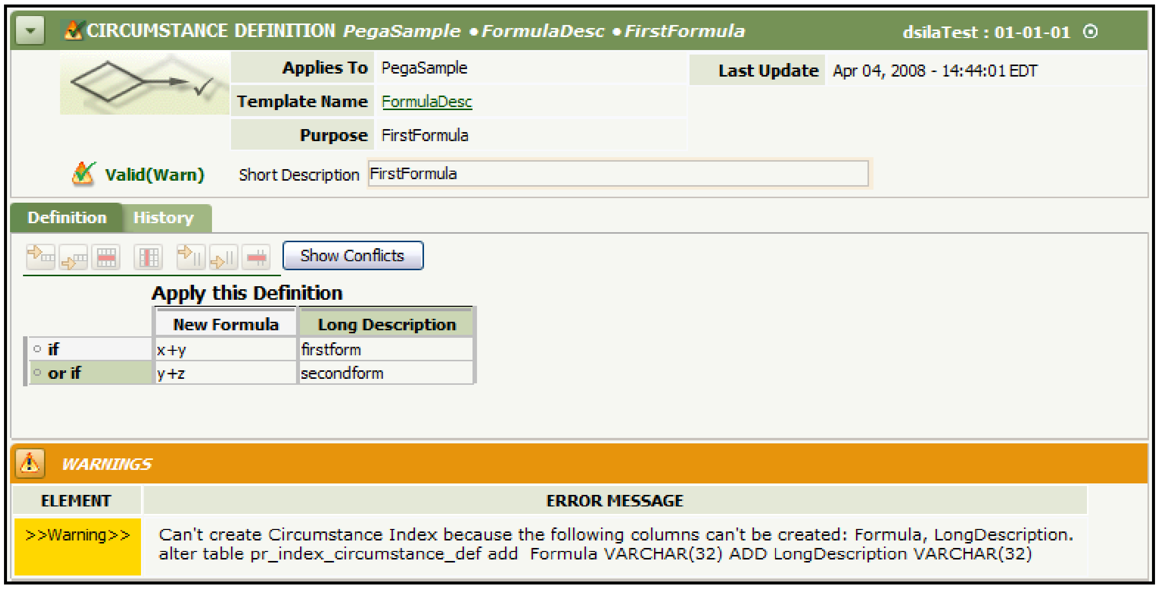
Template	Definition	State	Channel	Amount
Geography	IncludePrintedNotice	CT	Letter	>$1500
Geography	IncludePrintedNotice	MA	Letter	>$2000

To produce the report, the system runs a JOIN on the Rules table and this index table, and then:

- searches for the appropriate value in the index table (".pyState = MA")
- from these entries, gets the Circumstance Template & Definition values
- chooses the rules by their association with the Circumstance Template value and Circumstance Definition value

If the properties for the Circumstance Definition already exist in the table, the system will use the existing column (rather than creating a duplicate column). If the columns do not exist, those properties will be added into this table; hence, the need for ALTER TABLE privileges. The new properties will also be created in the Index-CircumstanceDefinition class.

All the index information is entered into the table when the Circumstance Definition rule is saved. If the user does *not* have ALTER TABLE privileges, then when they try to save the Circumstance Definition, they will receive a warning stating that the column could not be added to the table:



Applies To	PegaSample
Template Name	FormulaDesc
Purpose	FirstFormula
Last Update	Apr 04, 2008 - 14:44:01 EDT
Short Description	FirstFormula

Apply this Definition

	New Formula	Long Description
if	x+y	firstform
or if	y+z	secondform

ELEMENT	ERROR MESSAGE
>>Warning>>	Can't create Circumstance Index because the following columns can't be created: Formula, LongDescription. alter table pr_index_circumstance_def add Formula VARCHAR(32) ADD LongDescription VARCHAR(32)

NOTE: The Circumstance Definition rule itself will save; however, the index entry will not.

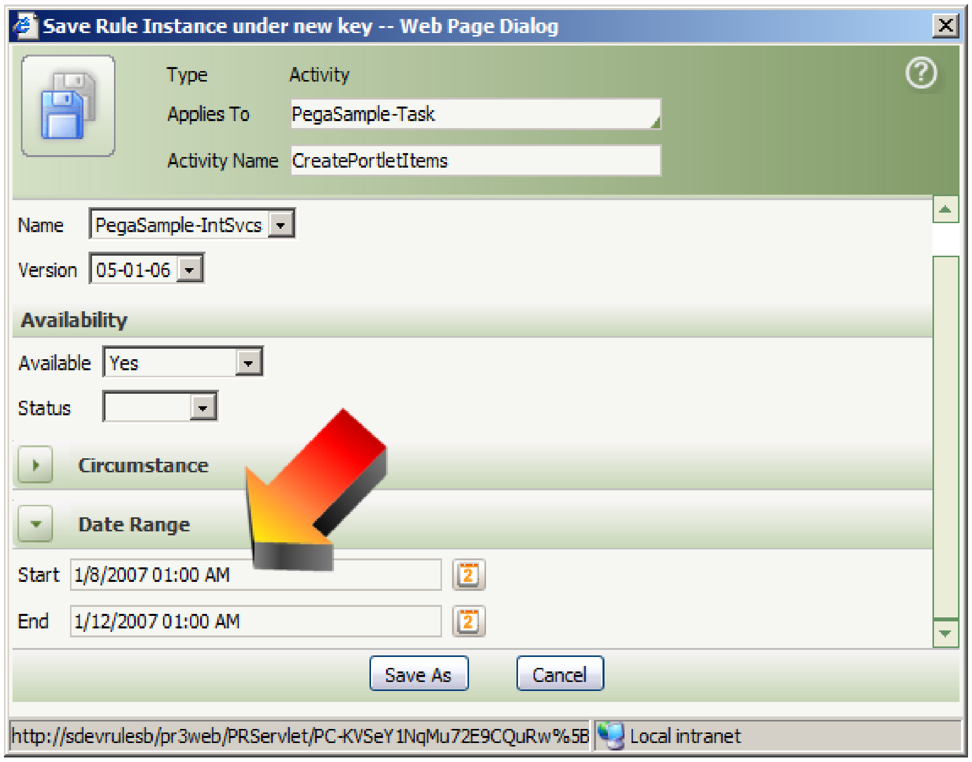
The concept behind this error message is that if the customer's DBA doesn't want to provide ALTER TABLE privileges, then the user who needs it can bring the SQL statement to the DBA, who can use it to create the appropriate column in the table.

After the column is created, the user must *resave* the Circumstance Definition, in order to populate the index table. If they do not resave the rule, it will not be included in the report.

Date/Time Ranges

When creating a rule, it is possible to set a date/time range during which a rule is valid. There may be several rules of a particular type, all with the same name, defined on the same class, with the same qualifiers (the same RuleSet/Version, Circumstance, and Circumstance Date), that differ by this set time range.

The Date/Time qualifier ties the execution of a Rule to a period of time, based on the current (system) date/time. For example, a Rule might be created to calculate customer discounts. For the month of July 2007, a special 10% discount might apply. The system will use the current date to determine whether the 10% discount is valid.


Save Rule Instance under new key -- Web Page Dialog

Type	Activity
Applies To	PegaSample-Task
Activity Name	CreatePortletItems
Name	PegaSample-IntSvcs
Version	05-01-06

Availability

Available	Yes
Status	

Circumstance

Date Range

Start	1/8/2007 01:00 AM
End	1/12/2007 01:00 AM

Save As
Cancel
http://sdevrulesb/pr3web/PRServlet/PC-KVSeY1NqMu72E9CQuRw%5B
Local intranet

If there are rules with a given name that specify a date range, then there *must* be at least one rule with that name with *no* date range specified, in case none of the date-qualified rules are appropriate for the run-time situation. This non-circumstanced rule is called the "default" rule. The default rule must be defined:

- on the same class as the circumstanced rules, or on an ancestor of that class
- in the same RuleSet version as the circumstanced rules, or in one of that RuleSet's prerequisites

A valid date-range rule will take precedence over the default rule (class, RuleSet, and Version being equal).

Unlike circumstance, there is no property associated with a Date/Time range; PegaRULES will automatically retrieve the current date/time information (to compare to the specified range) from the system clock.

It is also possible to have a *blank* (open-ended) Start Date or End Date.

- If the Start Date is blank, the Rule is considered to have been "always" in force (except when a more specific Rule applied), until the specified End Date.

- If the End Date is blank, then the Rule will be in force forever (or until a more specific Rule takes effect), beginning with the specified Start Date.

If there is more than one valid time range for a Rule, then the rule with the *end date* closest to the system date is chosen. If there are several dates where the end date is equivalent, then the rule with the *start date* closest to the system date is chosen.

	RuleSet/Version	Start Date	End Date
1.	Pega-ProCom:05-02-03	04/01/05[6]	
2.	Pega-ProCom:05-02-03	01/01/06	04/30/06
3.	Pega-ProCom:05-02-03	02/01/06	04/15/06
4.	Pega-ProCom:05-02-03	04/10/06	04/20/06
5.	Pega-ProCom:05-02-03	04/01/06	04/30/06

If the system contains rules of the same given name with the Rulesets, Versions, and Date/Time Ranges above, and if the system date were **04/18/06,** then the fourth rule would be chosen as the most accurate.

If the system date were **04/11/06**, then the third rule would be chosen. Even though this date fits into all the ranges except the first, it is closest to the end date of the third rule.

NOTE: If a rule is qualified by a Date range, but that rule is eliminated by an invalid Circumstance or Circumstance Date, the Date range will never be tested, as the rule will have been discarded before then.

[6] NOTE: As shown in the Web Page Dialog box above, the Date Range values are generally a date/timestamp, including a date *and a time.* For ease of reading in this tech note, these date values will be shown simply as dates; in a production system, the time should also be specified.

Standard Rule Resolution Process

As stated previously, there are a number of steps that the Rule Resolution process follows to determine which one rule should be used in a particular situation. Beginning with a large group of possible rules for the situation, the Rule Resolution process winnows down the rules until only one is available, using the following process:

1. Check the cache. If the rule is there, go to Step 8.
2. Choose all instances with correct purpose
3. Discard rules where Availability = No
4. Discard inapplicable RuleSets and Versions
5. Discard all candidates *not* defined on a class in the “ancestor tree”
6. Rank remaining candidates by: Override, Class, RuleSet, Circumstance, Circumstance Date, date/time; remove all that are withdrawn or hidden by other withdrawn candidates
    6a. Discard all choices that occur in the ranked list *after* the first “default” rule
7. Set the cache
8. Find best instance (and check to make sure there is not a duplicate)
9. Check that Availability does not show BLOCKED
10. Security – Verify that the Requestor is authorized to see the Rule

1. Check the Cache

The Rule Resolution process begins by checking to see if the cache has a list of Rule candidates for the given request. The cache is used to avoid the database lookups that would otherwise have to be done for Rule Resolution (choosing the class, checking availability, choosing the RuleSets and Versions).

If an appropriate entry is found in the cache, the system jumps ahead in the Rule Resolution process to **Step 8**.

A full description of the cache is beyond the scope of this document. In brief, the cache is used by PegaRULES to hold onto class instances which were recently retrieved from the database. Rule Resolution extends the cache to include the ability to alias entries in the cache that were actually returned by rule resolution. (Entries are aliased because the request made for rules is not always returned precisely as requested: if the system is searching for the activity **Display** for Acme-Auto-ClaimsEntry-Accident, and finds this activity on Acme-Auto, that is the entry that is aliased.) The alias records the original request, along with all the rules that might be considered for that request, based on the current RuleSet List and the requested class on which the rule is defined.

The alias also includes the applicable RuleSet List used in making the request, since different RuleSet Lists would lead to different Rule Resolution results.

2. Choose Instances with correct purpose

The “purpose” (also known as “Family Name”) is the combination of the values of all the key properties of a rule, *excluding* the “defined on” class. For all instances which belong to a class that uses inheritance, then the Class Name (pyClassName) is *not* part of the purpose. (Instances of a class which does *not* use inheritance generally do not have the Class Name as part of their key).

So for an Activity rule, the key properties include:

- the “Applies To” class (the class it is defined on)
- the Activity name

The purpose would be the Activity name (“Display”).

Likewise, for a Field Value, the key properties include:

- Applies To
- Field Name
- Field Value

The purpose would be the Field Name plus the Field Value (“pyActionPrompt.ViewHistory”).

During Rule Resolution, the system will begin by choosing *all* items of the appropriate purpose, and putting them in a temporary list. (So for the **Display** Activity example, all instances of Rule-Obj-Activity named **Display** would be chosen, regardless of the class they are defined on.)

3. Availability

The list of possible rules is filtered based on the availability of rules. In this step, Rule Resolution removes all rules from the list where **Availability = No**. (Further into the Rule Resolution process, there will be other Availability checks.)

4. RuleSets and Versions

Next, the Rule Resolution process looks through the list of possible Rules and discards the RuleSets and Versions that are not enabled for that user/requestor.

Example: RuleSet and Version evaluation

The example user’s profile is as follows:

AcmeInsuranceAuto: 05-02
AcmeInsurance: 05-02
Acme: 05-01

Thus, the only RuleSet Versions which would be considered valid for this user would be:

- in that major version (05 only; any rulesets that start with 04- or 06- will be ignored)
- equal to or below the listed minor version (05-03 is higher, so would be ignored for AcmeInsuranceAuto and AcmeInsurance; 05-02 would be ignored for Acme)

Instances of the Activity **Display** are defined on several classes, found in the following RuleSets and Versions in an example system.

Class	RuleSet	Version
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-03-01
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-03-01
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-52
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-02-10
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-02-10
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-01-52
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-01-52
Acme-Auto-ClaimsEntry	AcmeInsuranceAuto	05-01-52
Acme-Auto-ClaimsEntry	AcmeInsuranceAuto	05-01-52
Acme-Auto-ClaimsEntry	AcmeInsuranceAuto	05-01-52
Acme-Auto-ClaimsEntry	AcmeInsurance	05-02-10
Acme-Auto-ClaimsEntry	Acme	05-02-40
Acme-Auto-ClaimsEntry	Acme	05-02-40
Acme-Auto-ClaimsEntry	Acme	05-02-40
Acme-Auto-ClaimsEntry	Acme	05-01-01
Acme-Auto-ClaimsEntry	Acme	05-01-01
Acme-Auto-ClaimsEntry	Acme	05-01-01
Acme-Auto	AcmeInsurance	05-02-01
Acme-Auto	AcmeInsurance	05-02-01
Acme-Auto	AcmeInsurance	05-01-11
Acme-Auto	AcmeInsurance	05-01-10
Acme-Life	AcmeInsurance	05-02-01
Acme-Life	AcmeInsurance	05-02-01
Acme-Life	AcmeInsurance	05-01-11
Acme-Life	AcmeInsurance	05-01-10

When Rule Resolution begins, the highlighted RuleSets and Versions will be removed, due to the user’s profile.

5. Discard all candidates not in Ancestor Tree of Class

In this step, all the classes that are *not* in the *ancestor tree* of the Class being used are removed from the list. For example, if the Activity being Rule Resolved is **Display**, and the Class being used is Acme-Auto-ClaimsEntry-Accident, then only those classes from which Acme-Auto-ClaimsEntry-Accident descends via either *pattern* or *direct* inheritance will be considered. (For full details and an example, see the *Inheritance* section above.)

Based on the type of inheritance specified in the various classes, the Rule Resolution process will take different paths through the structural hierarchies of the classes to find applicable rules.

NOTE: For rules which do not have the *Use class-based inheritance to arrive at the correct rule to execute* box checked in their class definition, this step will not be used.

Example: Discard all not in Ancestor Tree

In this example, the class **Acme-Auto-ClaimsEntry-Accident** is the class where the user is calling the activity **Display.** The ancestors of this class include:

Acme-Auto-ClaimsEntry
Acme-Auto
Acme-

The class **Acme-Life** is *not* in this ancestor tree, so although the activity **Display** is defined on that class, the highlighted classes will not be considered for Rule Resolution of this rule.

Class	RuleSet	Version
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-52
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-02-10
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-02-10
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-01-52
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-01-52
Acme-Auto-ClaimsEntry	AcmeInsuranceAuto	05-01-52
Acme-Auto-ClaimsEntry	AcmeInsuranceAuto	05-01-52
Acme-Auto-ClaimsEntry	AcmeInsuranceAuto	05-01-52
Acme-Auto-ClaimsEntry	AcmeInsurance	05-02-10
Acme-Auto-ClaimsEntry	Acme	05-01-01
Acme-Auto-ClaimsEntry	Acme	05-01-01
Acme-Auto-ClaimsEntry	Acme	05-01-01
Acme-Auto	AcmeInsurance	05-02-01
Acme-Auto	AcmeInsurance	05-02-01
Acme-Auto	AcmeInsurance	05-01-11
Acme-Auto	AcmeInsurance	05-01-10
Acme-Life	AcmeInsurance	05-02-01
Acme-Life	AcmeInsurance	05-02-01
Acme-Life	AcmeInsurance	05-01-11
Acme-Life	AcmeInsurance	05-01-10

6. Rank Remaining Rules

During Rule Resolution, for a given named rule, the system searches across all the possible Rules in the list, and ranks them in the order of:

- Override
- Class
- RuleSet
- Version
- Circumstance
- Circumstance Date
- Date/Time Range

Important notes:

- The RuleSet and Version are based on the ordered list in the user's profile.
- A rule with a specific qualifier is ranked higher than one with no qualifier.
- Circumstanced rules are ranked alphabetically by Circumstance value.
- Circumstance Dates are ranked by descending value.
- Date/Time ranges are ranked first by their end-date (in ascending order), and then by their start-date (in descending order).
- Rules which do not have the *Use class-based inheritance to arrive at the correct rule to execute* box checked in their class definition will not be ranked by Class.

A default Rule (no qualifiers defined) is considered a match for *any* Circumstance, Circumstance Date, or Date/Time Range. Therefore, the system will discard any rules lower in the list than the first default rule found.

Although in general (as stated above) a rule with a specific qualifier is ranked higher than one with no qualifier, due to the ranking process, some of the qualifications may cause rules with inferior qualifiers to be chosen over other rules. Examples:

- A rule with an inferior date range or inferior Circumstance Date in a better RuleSet will be chosen over a rule with a better date range or Circumstance Date in an inferior RuleSet.
- A rule with an inferior date range or inferior Circumstance Date in a better class will be chosen over a rule with a better date range or Circumstance Date in an inferior class.
- A rule defined on a better class in an inferior RuleSet will be chosen over a rule defined on an inferior class in a better RuleSet.

Withdrawn Rules

Beginning in Version 5.2, Withdrawn rules are also considered. When Rule Resolution encounters a Withdrawn rule during the process of gathering up all the appropriate rules, that Withdrawn rule is placed on a separate holding list. All the rule candidates are referenced against the *Withdrawn* list:

- *after* the rules are ranked, but
- *before* the list is truncated at the first default rule

If any rule candidates match any rules on the Withdrawn list, they are removed from the candidate list.

A match occurs when the rules have the same:

- "defined-on" class
- RuleSet Name
- RuleSet major version
- Purpose (name)
- Qualifications (Circumstance Property, Circumstance Property Value, Date Property, Date Property Value, Start Date, End Date)

Override Rules

Override rules are rules which are stored in an *override* RuleSet. These rules are always moved to the top of the ranking, no matter what class or version they are.

For a description and examples of these rules, please see the *Rule Resolution with Override Rules Defined* section later in this doc.

Ranking Default vs. Circumstanced Rules

Beginning in Version 5.2, functionality was added to Process Commander to allow changes to a default rule which would not hide Circumstanced rules.

As stated above, the standard functionality has Circumstanced versions of a rule ranked higher than a non-circumstanced version of that same rule:

Class	RuleSet	RuleSet Version	Circumstance = State
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-11	NY
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	CA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MI
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	FL

Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	AZ
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	NV
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	

In the example above, there are many circumstanced versions of this rule, and one non-qualified or "default" rule, which is ranked at the bottom of the list.

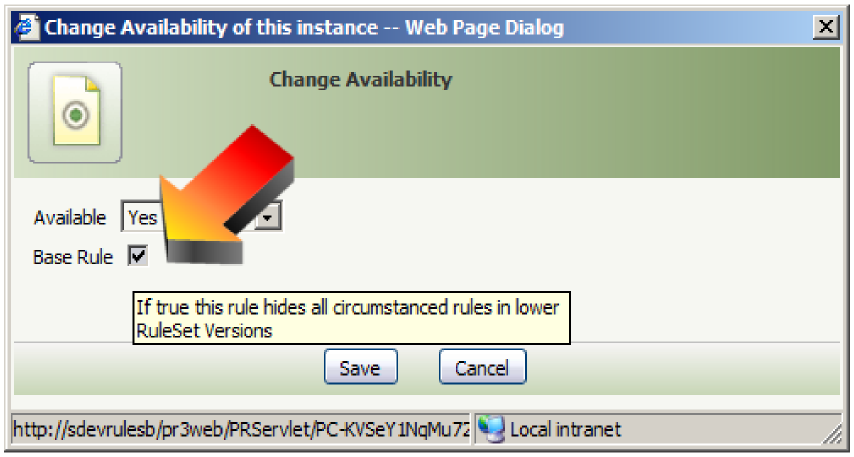
Prior to Version 5.2, because RuleSet Version is generally evaluated before circumstances, if the default rule is changed and then saved to a higher version, that default rule would rank higher in the list than the lower-versioned circumstanced rules. In order to continue to consider all the circumstanced rules in rule resolution, they would also have to be resaved to the higher RuleSet Version. This is not a problem if they all have to be changed to reflect the change from the default rule. If the change to the default rule does not affect the circumstanced rules, however (the "auto insurance rate" is changed for the states in the base rule, but not for the states in the circumstanced rules), then resaving all the circumstanced rules means a lot of extra work.

Therefore, beginning in Version 5.2, a higher-versioned default rule will *still* rank below all the lower-versioned circumstanced rules:

Class	**RuleSet**	**RuleSet Version**	**Circumstance = State**
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-11	NY
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	CA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MI
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	FL
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	AZ
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	NV
Acme-Auto-ClaimsEntry-Accident	**AcmeInsuranceAuto**	**05-02-20**	
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	

The highlighted rule has a higher RuleSet Version than the circumstanced rules, but ranks below them.

If the developer wished to have the higher-versioned default rule *hide* all the lower-versioned circumstanced rules (which was Version 4.2 functionality), they would create a **Base Rule**. A Base Rule is a non-qualified rule which, by virtue of a higher RuleSet Version, *hides* all the circumstanced rules in a lower RuleSet version.


Change Availability of this instance -- Web Page Dialog

Change Availability

Available Yes

Base Rule ☑

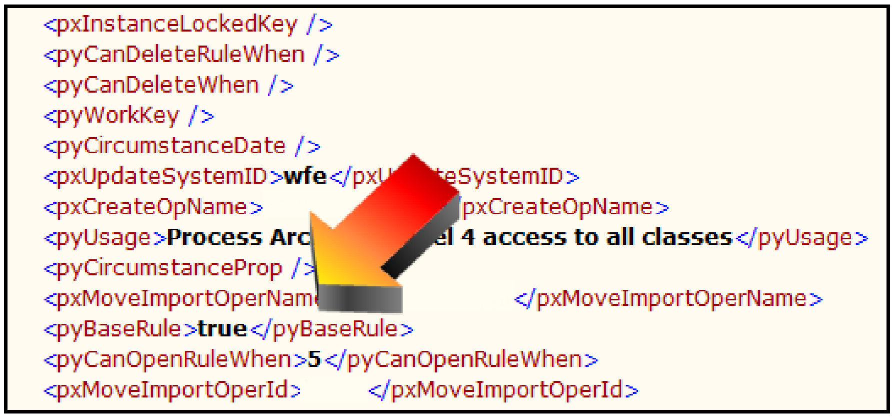
If true this rule hides all circumstanced rules in lower RuleSet Versions

In the **Availability** dialog box, the **Base Rule** checkbox is displayed.

Checking this box means that this is a Base Rule, and that this rule, being the higher-versioned default rule, will be ranked *above* the circumstanced versions of that rule, effectively “hiding” them from being chosen during the Rule Resolution process.

By default, this box is *unchecked* for rules which were created beginning in Version 5.2, meaning that the default base rule does *not* hide the circumstanced rules.

For rules in locked RuleSets, the **Availability** icon on the rule is grayed out, as it is not possible to change the Availability or the Base Rule status. In this case, the Base Rule status may be viewed by checking the **.pyBaseRule** property in the **Rule Data** screen:


```
<pxInstanceLockedKey />
<pyCanDeleteRuleWhen />
<pyCanDeleteWhen />
<pyWorkKey />
<pyCircumstanceDate />
<pxUpdateSystemID>wfe</pxU      teSystemID>
<pxCreateOpName>                pxCreateOpName>
<pyUsage>Process Arc            el 4 access to all classes</pyUsage>
<pyCircumstanceProp />
<pxMoveImportOperNam                    </pxMoveImportOperName>
<pyBaseRule>true</pyBaseRule>
<pyCanOpenRuleWhen>5</pyCanOpenRuleWhen>
<pxMoveImportOperId>        </pxMoveImportOperId>
```

In the above example, .pyBaseRule is set to *true*, meaning that the Base Rule box is checked. Therefore, this rule will hide lower-versioned circumstanced rules.

For rules created prior to Version 5.2, .pyBaseRule is set to *true*. For examples of Rule Resolution behavior when the box is checked (.pyBaseRule is set to true), or for Process Commander versions prior to Version 5.2, please reference Appendix A.

IMPORTANT: In order for this new feature to apply, the new default rule *must* have the same definition as the old default rule. This includes:

- defined on the same class
- defined in the same RuleSet
- defined within the same major RuleSet Version (minor version/revision may be different)
- not circumstanced

Example: Ranking the Rules – Version 5.2

For the activity **Display** on the class **Acme-Auto-ClaimsEntry-Accident,** the remaining rules resulting from the previous examples (with qualifiers now added in) are ranked as follows:

Class	**RuleSet**	**Version**	**Circum-stance = Customer Level**	**Circum-stance Date**	**Date/Time Range**
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze	10/24/2006	09/01/05 - 09/30/05
Acme-Auto-ClaimsEntry-Accident	***AcmeInsuranceAuto***	***05-02-10***	***Bronze***	***11/10/2004***	***04/01/06 - 04/09/06***
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze		02/01/06 - 04/15/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze		04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Gold		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-52	Gold		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01		4/1/2005	04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident**	AcmeInsuranceAuto	05-02-20			
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10			
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01			
Acme-Auto-ClaimsEntry-Accident	***AcmeInsurance***	***05-02-10***	***Gold***		***04/01/06 - 04/09/06***
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-02-10	Gold		04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-01-52	Bronze		02/01/06 - 04/15/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-01-52	Silver		
Acme-Auto-ClaimsEntry	AcmeInsuranceAuto	05-01-52	Bronze		
Acme-Auto-ClaimsEntry	AcmeInsurance	05-02-10	Gold	4/1/2005	02/01/06 - 04/15/06
Acme-Auto-ClaimsEntry	***Acme***	***05-01-01***	***Bronze***		***02/01/06 - 04/15/06***
Acme-Auto-ClaimsEntry	Acme	05-01-01	Gold		
Acme-Auto-ClaimsEntry	Acme	05-01-01			
Acme-Auto	AcmeInsurance	05-02-01	Gold		
Acme-Auto	AcmeInsurance	05-01-11			04/01/06 - 04/30/06
Acme-Auto	AcmeInsurance	05-01-10	Gold		
Acme-Auto**	AcmeInsurance	05-02-01			

NOTES:

- The asterisked rules are the Version 5.2 “non-hiding default” rules; even though the RuleSet Version is higher than the circumstanced rules above it, the circumstanced rules are ranked higher.

- ***The rules which are italicized and bolded*** match rules which are the Withdrawn Rule list, so they will be withdrawn from consideration at this point in the process.

7. Add List to Cache

After the Withdrawn rules are processed, the list is truncated at the first completely non-qualified or default rule. (Since a default rule is considered a “match” for all circumstances, all rules ranked below that default rule would not be considered.)

Class	**RuleSet**	**Version**	**Circum-stance = Customer Level**	**Circum-stance Date**	**Date/Time Range**
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze	10/24/2006	09/01/05 - 09/30/05
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze		02/01/06 - 04/15/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze		04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Gold		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-52	Gold		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01		4/1/2005	04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident*	AcmeInsuranceAuto	05-02-20			

At this point, the list of rules will be added to the Cache. The list will contain the as-yet unqualified rules (not yet filtered for Circumstance, Circumstance Date, or Date/Time) that are potentially selectable. (Caching the precise rule for the request at the very end of this process, and *including* the Circumstance, Circumstance Date, and Date/Time information, would mean that too few other requests would be able to use that cached information; there would be too much non-reusable information cached, and the point of a cache is to be able to re-use the data.)

The cache is keyed by RuleSet List and the key of the Class that was requested (thus, the cache would be keyed on Acme-Auto-ClaimsEntry-Accident if that were requested, even if the instances are all in Acme-Auto). NOTE: The key does *not* include any of the qualifiers.

8. Find First Appropriate Instance

After ranking the possible Rules, the system searches down this list for the first match – since the rules are ranked with the “best” match higher in the list - where the first match is either an exact match to a Circumstanced Rule, with the correct Circumstance Date, in the correct Date/Time Range, or the default Rule, with no qualifiers. As soon as either of these situations is found, the search is stopped and that rule is chosen. (See Example 8.)

After the one rule is chosen, the next rule in the list is checked to see if it is equally as correct, according to the Rule Resolution process. If it is equally correct (same qualifier values, same RuleSet and Version), then the system asserts that it is actually a *duplicate* rule of the chosen rule. At this point, neither rule is chosen; an exception is thrown stating that there are duplicate rules in the system.

Example: Choosing the Best Rule

Scenario 1:

The date that this Rule Resolution is being run is **April 5, 2006**
The Circumstance property is “Customer Level,” with a value of **Silver**.
The Circumstance Date property is “PolicyDate,” with a value of **01/01/2006**.

Rule Resolution would begin processing the list of rules by removing all rules which do not match the **Circumstance** property value of “Silver” (leaving only those rules which have either “Silver” or no value for Circumstance). In this case, there are *no* Circumstanced rules with a value of “Silver,” so the default rule is chosen.

Class	RuleSet	Version	Circum-stance = Customer Level	Circum-stance Date	Date/Time Range
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze	10/24/2006	09/01/05 - 09/30/05
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze		02/01/06 - 04/15/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze		04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Gold		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-52	Gold		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01		4/1/2005	04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident*	AcmeInsuranceAuto	05-02-20			

Scenario 2:

The date that this Rule Resolution is being run is **April 5, 2006**
The Circumstance property is “Customer Level,” with a value of **Bronze**.
The Circumstance Date property is “PolicyDate,” with a value of **01/01/2006**.

- All rules which do not match the **Circumstance** property value of “Bronze” were removed (leaving only those rules which have either “Bronze” or no value for Circumstance).

- All rules where the **Circumstance Date** is *after* the Policy Date of 01/01/06 were removed.

The following rules remained in the list:

Class	RuleSet	Version	Circum-stance = Customer Level	Circum-stance Date	Date/Time Range
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze	10/24/2006	09/01/05 - 09/30/05
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze		02/01/06 - 04/15/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze		04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Gold		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-52	Gold		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01		4/1/2005	04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident*	AcmeInsuranceAuto	05-02-20			

Both Date/Time Ranges are valid for April 5. The highlighted rule has an end date closest to the system date, so that rule is chosen.

Scenario 3:

The date that this Rule Resolution is being run is **May 5, 2006**
The Circumstance property is “Customer Level,” with a value of **Bronze**.
The Circumstance Date property is “PolicyDate,” with a value of **01/01/2006**.

- All rules which do not match the **Circumstance** property value of “Bronze” were removed (leaving only those rules which have either “Bronze” or no value for Circumstance).

- All rules where the **Circumstance Date** is *after* the Policy Date of 01/01/06 were removed.

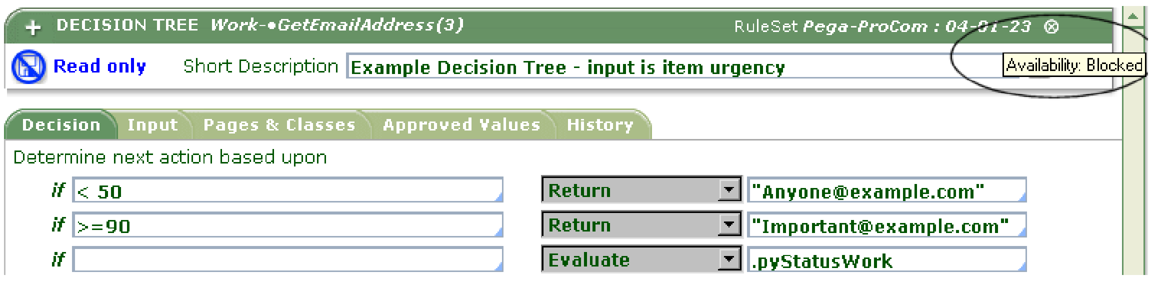
- All rules which have a Date-Time range ending *before* May 5 were removed.

Class	RuleSet	Version	Circum-stance = Customer Level	Circum-stance Date	Date/Time Range
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze	10/24/2006	09/01/05 - 09/30/05
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze		02/01/06 - 04/15/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze		04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Gold		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-52	Gold		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01		4/1/2005	04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident*	AcmeInsuranceAuto	05-02-20			

Once again, the only rule which meets all the criteria is the default rule.

9. Availability

Availability is checked again on the remaining rule. The first check was to be sure that Availability did *not* equal NO. This check sees if this rule is BLOCKED for this user in this situation. If **Available = Blocked** for this rule, then the system reports "not found".



DECISION TREE *Work-•GetEmailAddress(3)* RuleSet *Pega-ProCom : 04-01-23*

Read only Short Description: Example Decision Tree - input is item urgency

Availability: Blocked

Decision | Input | Pages & Classes | Approved Values | History

Determine next action based upon

if	< 50	Return	"Anyone@example.com"
if	>=90	Return	"Important@example.com"
if		Evaluate	.pyStatusWork

If the rule is not marked "Blocked," then the final check is made for security, and then the rule is used in processing.

10. Security

Security is involved in Rule Resolution through Rule-Access-Role-Obj. A full discussion of this topic is outside the scope of this document. Please reference the ***Administration and Security Guide*** for details on Security.

Rule Resolution with “Base Rule” defined

Beginning in Version 5.2, functionality was added to Process Commander to allow changes to a default rule which would not hide Circumstanced rules.

Prior to Version 5.2, the standard functionality has Circumstanced versions of a rule ranked higher than a non-circumstanced version of a rule – class and RuleSet Version being equal:

Class	RuleSet	RuleSet Version	Circumstance = State
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-11	NY
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	CA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MI
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	FL
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	AZ
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	NV
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	

New Default Rule

In the example above, there are many circumstanced versions of this rule, and one default rule which is ranked below the circumstanced rules.

Because RuleSet Version is evaluated before circumstances, if the default rule is changed and then saved to a higher version, that default rule will then rank higher in the list than the lower-versioned circumstanced rules.   In Version 4.2 and 5.1 implementations, all of the circumstanced rules would now be ignored:

Class	RuleSet	RuleSet Version	Circumstance = State
Acme-Auto-ClaimsEntry-Accident	**AcmeInsuranceAuto**	**05-02-20**	
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-11	NY
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	CA
Acme-Auto-	AcmeInsuranceAuto	05-02-10	MI

ClaimsEntry-Accident			
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	FL
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	AZ
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	NV
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	

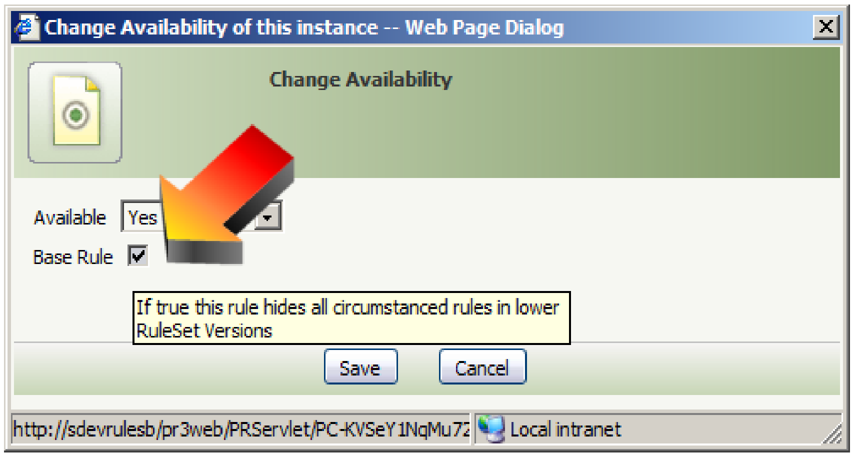
In order to continue to consider all the circumstanced rules in rule resolution, they would also have to be resaved to the higher RuleSet Version. This is not a problem if they all have to be changed to reflect the change from the default rule. If the change to the default rule does not affect the circumstanced rules, however (the “auto insurance rate” is changed for the states in the base rule, but not for the states in the circumstanced rules), then resaving all the circumstanced rules means a lot of extra work.

Therefore, beginning in Version 5.2, a higher-versioned default rule will *still* rank below all the lower-versioned circumstanced rules:

Class	RuleSet	RuleSet Version	Circumstance = State
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-11	NY
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	CA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MI
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	FL
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	AZ
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	NV
Acme-Auto-ClaimsEntry-Accident	**AcmeInsuranceAuto**	**05-02-20**	
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	

The highlighted rule has a *higher* RuleSet Version than the circumstanced rules, but ranks *below* them.

If the developer wished to have the higher-versioned default rule *hide* all the lower-versioned circumstanced rules (which was Version 4.2 functionality), they would create a **Base Rule**. A Base Rule is a non-qualified rule which, by virtue of a higher RuleSet Version, *hides* all the circumstanced rules in a lower RuleSet version.


Change Availability of this instance -- Web Page Dialog
Change Availability
Save
Cancel
http://sdevrulesb/pr3web/PRServlet/PC-KVSeY1NqMu72
Local intranet

Available Yes

Base Rule ✓

If true this rule hides all circumstanced rules in lower RuleSet Versions

In the **Availability** dialog box, the **Base Rule** checkbox is displayed.

Checking this box means that the higher-versioned default rule will be ranked *above* the circumstanced versions of that rule, effectively “hiding” them from being chosen during the Rule Resolution process – pre-5.2 behavior.

Leaving this checkbox *unchecked* means that the higher-versioned default rule will be ranked *below* the circumstanced versions of that rule.

New default rule defined on "ancestor" class

There can be additional variations on the default functionality. The existing default rule may be defined on an "ancestor" class:

Class	RuleSet	RuleSet Version	Circumstance = State
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-11	NY
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	CA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MI
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	FL
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	AZ
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	NV
Acme-Auto	**AcmeInsuranceAuto**	**05-02-10**	

Because class is used as the first ranking criteria, rule resolution ranking behavior is not changed in this example:

Class	RuleSet	RuleSet Version	Circumstance = State
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-11	NY
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	CA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MI
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	FL
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	AZ
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	NV
Acme-Auto	**AcmeInsuranceAuto**	**05-02-20**	
Acme-Auto	AcmeInsuranceAuto	05-02-10	

However, if some of the circumstanced classes were defined on an "ancestor" class, the ranking might change. The existing rules may be defined as follows:

Class	**RuleSet**	**RuleSet Version**	**Circumstance = State**
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-11	NY
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	CA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MI
Acme-Auto	AcmeInsuranceAuto	05-02-10	FL
Acme-Auto	AcmeInsuranceAuto	05-02-10	AZ
Acme-Auto	AcmeInsuranceAuto	05-02-10	NV
Acme-Auto	AcmeInsuranceAuto	05-02-10	

Adding in the new default rule with **Base Rule** *checked* (the 5.1 and prior behavior) would result in the following ranking:

Class	RuleSet	RuleSet Version	Circumstance = State
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-11	NY
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	CA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MI
Acme-Auto	**AcmeInsuranceAuto**	**05-02-20**	
Acme-Auto	AcmeInsuranceAuto	05-02-10	FL
Acme-Auto	AcmeInsuranceAuto	05-02-10	AZ
Acme-Auto	AcmeInsuranceAuto	05-02-10	NV
Acme-Auto	AcmeInsuranceAuto	05-02-10	

Adding in the new default rule with **Base Rule** *unchecked* (5.2 functionality) would result in the following ranking:

Class	RuleSet	RuleSet Version	Circumstance = State
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-11	NY
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	CA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MI
Acme-Auto	AcmeInsuranceAuto	05-02-10	FL
Acme-Auto	AcmeInsuranceAuto	05-02-10	AZ
Acme-Auto	AcmeInsuranceAuto	05-02-10	NV
Acme-Auto	**AcmeInsuranceAuto**	**05-02-20**	
Acme-Auto	AcmeInsuranceAuto	05-02-10	

New default rule defined on "inferior" RuleSet

In addition to "ancestor" class, the existing default rule may be defined on an "inferior" ("lower") RuleSet. Using the above example again, the rules may be defined as follows:

Class	RuleSet	RuleSet Version	Circumstance = State
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-11	NY
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	CA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MI
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	FL
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	AZ
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	NV
Acme-Auto-ClaimsEntry-Accident	**Acme**	**05-02-10**	

Again, because RuleSet is used as a higher ranking criteria than circumstance, rule resolution ranking behavior is not changed in this example:

Class	RuleSet	RuleSet Version	Circumstance = State
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-11	NY
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	CA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MI
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	FL
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	AZ
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	NV
Acme-Auto-ClaimsEntry-Accident	**Acme**	**05-02-20**	
Acme-Auto-ClaimsEntry-Accident	Acme	05-02-10	

However, if some of the circumstanced classes were also defined on the "inferior" RuleSet, the ranking might change. The existing rules may be defined as follows:

Class	RuleSet	RuleSet Version	Circumstance = State
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-11	NY
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	CA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MI
Acme-Auto-ClaimsEntry-Accident	Acme	05-02-10	FL
Acme-Auto-ClaimsEntry-Accident	Acme	05-02-10	AZ
Acme-Auto-ClaimsEntry-Accident	Acme	05-02-10	NV
Acme-Auto-ClaimsEntry-Accident	**Acme**	**05-02-10**	

Adding in the new default rule with **Base Rule** *checked* (the 5.1 and prior behavior) would result in the following ranking:

Class	RuleSet	RuleSet Version	Circumstance = State
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-11	NY
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	CA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MI
Acme-Auto-ClaimsEntry-Accident	**Acme**	**05-02-20**	
Acme-Auto-ClaimsEntry-Accident	Acme	05-02-10	FL
Acme-Auto-ClaimsEntry-Accident	Acme	05-02-10	AZ
Acme-Auto-ClaimsEntry-Accident	Acme	05-02-10	NV
Acme-Auto-ClaimsEntry-Accident	Acme	05-02-10	

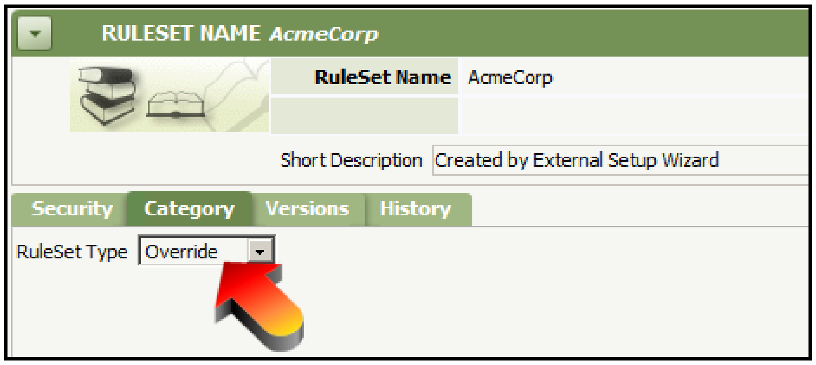
Adding in the new default rule with **Base Rule** *unchecked* (5.2 functionality) would result in the following ranking:

Class	RuleSet	RuleSet Version	Circumstance = State
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-11	NY
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	CA
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	MI
Acme-Auto-ClaimsEntry-Accident	Acme	05-02-10	FL
Acme-Auto-ClaimsEntry-Accident	Acme	05-02-10	AZ
Acme-Auto-ClaimsEntry-Accident	Acme	05-02-10	NV
Acme-Auto-ClaimsEntry-Accident	**Acme**	**05-02-20**	
Acme-Auto-ClaimsEntry-Accident	Acme	05-02-10	

Rule Resolution with “Override Rules” defined

Override rules are rules which are saved into an Override RuleSet.

An Override RuleSet is designated on the **Category** tab of the RuleSet Name rule.

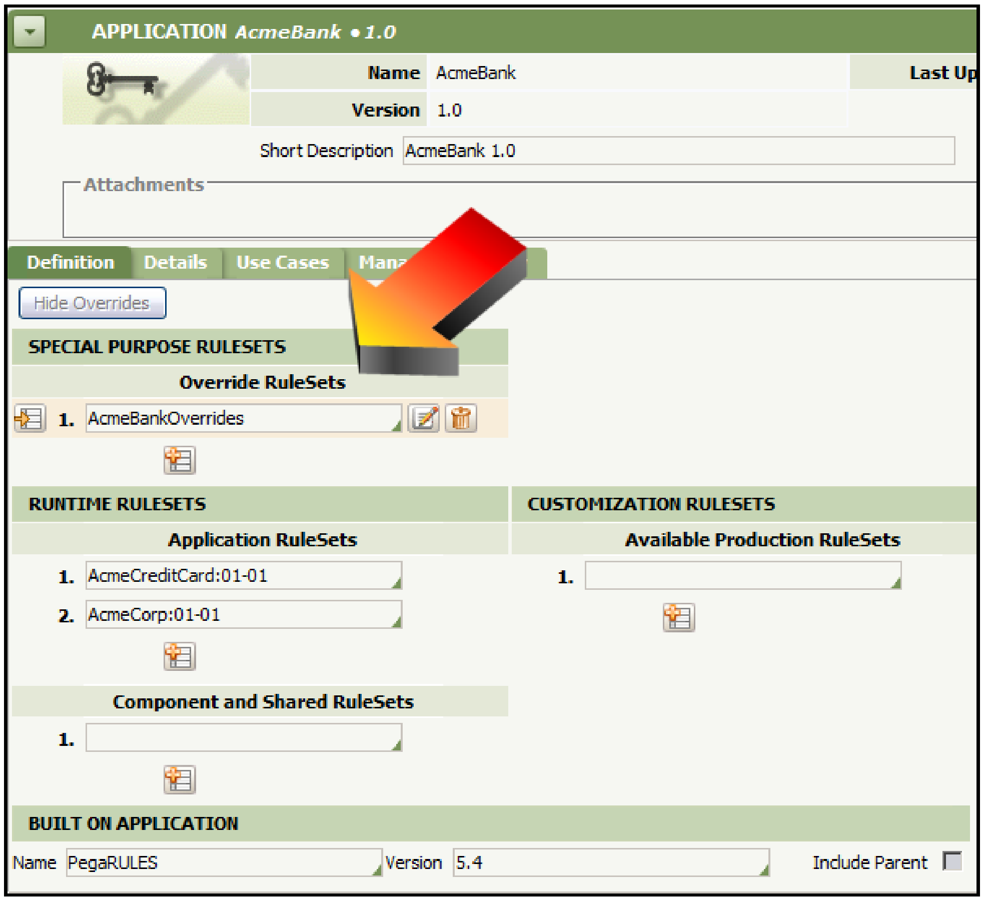

RULESET NAME *AcmeCorp*

RuleSet Name AcmeCorp

Short Description Created by External Setup Wizard

RuleSet Type Override

Override RuleSets are always listed at the top of any application RuleSet List. Therefore, they are displayed at the top of the Application ruleform, as a *Special Purpose RuleSet.*


APPLICATION *AcmeBank • 1.0*

Name AcmeBank

Version 1.0

Last Up

Short Description AcmeBank 1.0

SPECIAL PURPOSE RULESETS

Override RuleSets

1. AcmeBankOverrides

RUNTIME RULESETS

Application RuleSets

1. AcmeCreditCard:01-01
2. AcmeCorp:01-01

Component and Shared RuleSets

1.

CUSTOMIZATION RULESETS

Available Production RuleSets

1.

BUILT ON APPLICATION

Name PegaRULES Version 5.4 Include Parent

Override rules do exactly as their name implies: they override other rules of that same *name* and in that class inheritance path (examples below). They should be used *only* in extenuating circumstances, to handle emergencies. For example, suppose a bank has a mortgage application which approves mortgage loans all over the US. Suddenly a major disaster hits some part of the country (like Mount St. Helens erupting, or Hurricane Katrina). The bank would want to immediately stop approving mortgages for that area, so they would override the appropriate rules which involve the approval process. By using an override rule (rather than making some other change), they can be certain that all versions of the approval rules are changed.

IMPORTANT NOTES:

- Override rules *cannot be checked out*. Since the Override RuleSet is listed "higher" in the RuleSet List than all other rules, it is also higher than users' personal RuleSets, which are where the checked out rules are staged.

- Override rules *cannot be localized*. It is not possible to have one Override RuleSet dependent on another, which is how localized RuleSets are structured.

- Override rules *can* be marked as "withdrawn", using the same process as withdrawing a normal rule.

- Once a RuleSet has been designated as Override, it *cannot be changed* to any other RuleSet Type.

- Only the override RuleSet for the top-level application will be included in that application's RuleSet. Override RuleSets defined in the "built-on" application will *not* be included. (See the KB article #25158, *How Process Commander Assembles a RuleSet List.*)

Override Examples

As stated earlier in this document, the following process is used in rule resolution:

1. Check the cache. If the rule is there, go to Step 8.
2. Choose all instances with correct purpose
3. Discard rules where Availability = No
4. Discard inapplicable RuleSets and Versions
5. Discard all candidates *not* defined on a class in the "ancestor tree"
6. **Rank remaining candidates by: Override, Class, RuleSet, Circumstance, Circumstance Date, date/time; remove all that are withdrawn or hidden by other withdrawn candidates**
    6a. Discard all choices that occur in the ranked list *after* the first "default" rule
7. Set the cache
8. Find best instance (and check to make sure there is not a duplicate)
9. Check that Availability does not show BLOCKED
10. Security – Verify that the Requestor is authorized to see the Rule

Override rules are checked in Step 6 of this process. Thus, in the below example, the system is looking for the activity *Display* for the class *Acme-Auto-ClaimsEntry-Accident*. Inappropriate purposes, RuleSets, Versions, and classes have been discarded.

The rules which are left might include:

Class	RuleSet	Version	Circum-stance = Customer Level	Circum-stance Date	Date/Time Range
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze	10/24/2006	09/01/05 - 09/30/05
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze		02/01/06 - 04/15/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze		04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Gold		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-52	Gold		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01		4/1/2005	04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident**	AcmeInsuranceAuto	05-02-20			
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10			
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01			
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-02-10	Gold		04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-01-52	Bronze		02/01/06 - 04/15/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-01-52	Silver		
Acme-Auto-ClaimsEntry	AcmeInsuranceAuto	05-01-52	Bronze		
Acme-Auto-ClaimsEntry	AcmeInsurance	05-02-10	Gold	4/1/2005	02/01/06 - 04/15/06
Acme-Auto-ClaimsEntry	Acme	05-01-01	Gold		
Acme-Auto-ClaimsEntry	Acme	05-01-01			
Acme-Auto	AcmeInsurance	05-02-01	Gold		
Acme-Auto	AcmeInsurance	05-01-11			04/01/06 - 04/30/06
Acme-Auto	AcmeInsurance	05-01-10	Gold		
Acme-Auto**	AcmeInsurance	05-02-01			

Since a default rule (no qualifiers defined) is considered a match for any Circumstance, Circumstance Date, or Date/Time Range, the list is truncated at the first default rule. So normally the possible list of matches would be confined to the following:

Class	**RuleSet**	**Version**	**Circum-stance = Customer Level**	**Circum-stance Date**	**Date/Time Range**
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze	10/24/2006	09/01/05 - 09/30/05
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze		02/01/06 - 04/15/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze		04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Gold		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-52	Gold		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01		4/1/2005	04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-20			

However, an override rule, no matter the class, is always considered first, so the rule resolution is different.

Example 1: Override rule present

In this example, the rule in the Override Ruleset (in **bold** type) overrides *all* other rules for the class Acme-Auto-ClaimsEntry-Accident – even those in other RuleSets, a higher RuleSet, or circumstanced.

Class	RuleSet	Version	Circum-stance = Customer Level	Circum-stance Date	Date/Time Range
Acme-Auto-ClaimsEntry-Accident	***AcmeInsurance Override***	***01-01-01***			
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-02-20			
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze	10/24/2006	09/01/05 - 09/30/05
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze		04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Gold		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-52	Gold		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01		4/1/2005	04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01			
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-02-10	Gold		04/01/06 - 04/09/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-01-52	Bronze		02/01/06 - 04/15/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-01-52	Silver		
Acme-Auto-ClaimsEntry	AcmeInsuranceAuto	05-01-52	Bronze		
Acme-Auto-ClaimsEntry	AcmeInsurance	05-02-10	Gold	4/1/2005	02/01/06 - 04/15/06
Acme-Auto-ClaimsEntry	Acme	05-01-01			
Acme-Auto-ClaimsEntry	Acme	05-01-01	Gold		
Acme-Auto	AcmeInsurance	05-02-01	Gold		
Acme-Auto	AcmeInsurance	05-02-01			
Acme-Auto	AcmeInsurance	05-01-11	Gold		
Acme-Auto	AcmeInsurance	05-01-11			04/01/06 - 04/30/06

Truncating at the first "default" rule produces the following "list" of available rules:

Class	**RuleSet**	**Version**	**Circum-stance = Customer Level**	**Circum-stance Date**	**Date/Time Range**
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance Override	01-01-01			

Only the override rule is available.

Example 2: Circumstanced override rule

In this example, the override rule is circumstanced.

Class	**RuleSet**	**Version**	**Circum-stance = Customer Level**	**Circum-stance Date**	**Date/Time Range**
Acme-Auto-ClaimsEntry-Accident	***AcmeInsurance Override***	***01-01-01***	***Gold***		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze	10/24/2006	09/01/05 - 09/30/05
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze		04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-52	Gold		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01		4/1/2005	04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10			
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01			
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-02-10	Gold		04/01/06 - 04/09/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-01-52	Bronze		02/01/06 - 04/15/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-01-52	Silver		
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-02-05			
Acme-Auto-ClaimsEntry	AcmeInsuranceAuto	05-01-52	Bronze		
Acme-Auto-ClaimsEntry	AcmeInsurance	05-02-10	Gold	4/1/2005	02/01/06 - 04/15/06
Acme-Auto-ClaimsEntry	Acme	05-01-01			
Acme-Auto-ClaimsEntry	Acme	05-01-01	Gold		

In this case, the default rule is a "standard" rule, and the list is truncated somewhat lower:

Class	**RuleSet**	**Version**	**Circum-stance = Customer Level**	**Circum-stance Date**	**Date/Time Range**
Acme-Auto-ClaimsEntry-Accident	***AcmeInsurance Override***	***01-01-01***	***Gold***		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze	10/24/2006	09/01/05 - 09/30/05
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze		04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01		4/1/2005	04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10			

If the run-time value of the .CustomerLevel property is "Gold," then the override rule will be run. If the value is "Bronze" or some other value ("Silver"), then the override rule will *not* be used.

Example 3: Override rule present in "less specific" class in the hierarchy

In this example, the rule inheritance is followed not only through the pattern inheritance, but also through directed inheritance. Acme-Auto-ClaimsEntry has directed inheritance to class Work-Object, and a version of the *Display* activity is present in some additional classes in the directed hierarchy:

Class	**RuleSet**	**Version**	**Circum-stance = Customer Level**	**Circum-stance Date**	**Date/Time Range**
Work-	***AcmeInsurance Override***	***01-01-01***			
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-20			
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze	10/24/2006	09/01/05 - 09/30/05
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze		04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01			
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-02-10	Gold		04/01/06 - 04/09/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-01-52	Bronze		02/01/06 - 04/15/06
Acme-Auto-ClaimsEntry	Acme	05-01-01			
Acme-Auto-ClaimsEntry	Acme	05-01-01	Gold		
Acme-Auto	AcmeInsurance	05-02-01	Gold		
Acme-Auto	AcmeInsuranceAuto	05-02-01			
Acme-Auto	AcmeInsurance	05-01-11	Gold		
Work-Object	Pega-ProCom	05-04-01			
Work-	Pega-ProCom	05-04-01			
@baseclass	Pega-RULES	05-04-01			

Here, the override rule is defined on the class Work-. Again, it is moved to the top of the list, resulting in the following truncated list:

Class	**RuleSet**	**Version**	**Circum-stance = Customer Level**	**Circum-stance Date**	**Date/Time Range**
Work-	AcmeInsurance Override	01-01-01			

The override rule defined on Work- would again be the only one available to be chosen, even though there were many other rules with more specific classes that were present.

Example 4: Multiple override rules

In this example, there are several override rules. *Within* the override RuleSet, rule resolution works normally.

Class	RuleSet	Version	Circum-stance = Customer Level	Circum-stance Date	Date/Time Range
Acme-Auto-ClaimsEntry-Accident	***AcmeInsurance Override***	***01-01-01***	***Gold***		
Acme-Auto-ClaimsEntry	***AcmeInsurance Override***	***01-01-01***			
Work-	***AcmeInsurance Override***	***01-01-01***			
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze	10/24/2006	09/01/05 - 09/30/05
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10	Bronze		04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-52	Gold		
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01		4/1/2005	04/01/06 - 04/30/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-02-10			
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-02-05			
Acme-Auto-ClaimsEntry-Accident	AcmeInsuranceAuto	05-01-01			
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-02-10	Gold		04/01/06 - 04/09/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-01-52	Bronze		02/01/06 - 04/15/06
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance	05-01-52	Silver		
Acme-Auto-ClaimsEntry	AcmeInsuranceAuto	05-01-52	Bronze		
Acme-Auto-ClaimsEntry	AcmeInsurance	05-02-10	Gold	4/1/2005	02/01/06 - 04/15/06
Acme-Auto-ClaimsEntry	Acme	05-01-01	Gold		
Acme-Auto	AcmeInsurance	05-02-01	Gold		
Acme-Auto	AcmeInsurance	05-02-01			
Acme-Auto	AcmeInsurance	05-01-11	Gold		
Acme-Auto	AcmeInsurance	05-01-11			04/01/06 - 04/30/06
Work-Object	Pega-ProCom	05-04-01			
Work-	Pega-ProCom	05-04-01			
@baseclass	Pega-RULES	05-04-01			

In this situation, after truncating at the first default rule, the following rules are available:

Class	**RuleSet**	**Version**	**Circum-stance = Customer Level**	**Circum-stance Date**	**Date/Time Range**
Acme-Auto-ClaimsEntry-Accident	AcmeInsurance Override	01-01-01	Gold		
Acme-Auto-ClaimsEntry	AcmeInsurance Override	01-01-01			

If the .CustomerLevel property have a value of “Gold” at runtime, the first rule would be used; otherwise, the rule with no circumstancing would be used.