Using Business Rules with Ensemble

Version 2012.2
31 August 2012
# Table of Contents

**About This Book** .................................................................................................................................... 1

1 About Business Rules .............................................................................................................................................. 3
  1.1 Rules as Classes ................................................................. 5

2 Introduction to the Rule Editor .............................................................................................................................. 7
  2.1 Business Rule List ............................................................... 7
  2.2 Business Rule Editor ............................................................ 7
    2.2.1 Business Rule Wizard ...................................................... 9

3 Creating and Editing Rule Sets .......................................................................................................................... 11
  3.1 Adding Rules ............................................................................. 12
  3.2 Adding Actions ........................................................................... 12
  3.3 Using the Associated Editors ..................................................... 13
    3.3.1 Using the Rule Constraint Editor ....................................... 14
    3.3.2 Using the Production Configuration Item Selector .............................. 15
    3.3.3 Selecting the Transformation and Target of a Send Action ................. 15
    3.3.4 Using the Expression Editor ................................................. 16
  3.4 Adding Business Rule Notification ............................................ 21

4 Invoking a Rule from BPL ................................................................................................................................. 23
  4.1 Using the BPL <rule> Element ................................................ 23
    4.1.1 Identifying the Rule ......................................................... 24
    4.1.2 BPL <rule> Example ......................................................... 24
  4.2 BPL Business Process Designer .................................................. 25

5 Example Rule Classes ....................................................................................................................................... 29
  5.1 General Business Rule Example .............................................. 29
  5.2 HL7 Message Routing Rule Example .......................................... 30
  5.3 General Message Routing Rule Example ....................................... 32
  5.4 Virtual Document Message Routing Rule Example ............................ 32

6 Viewing the Business Rule Log ...................................................................................................................... 35
  6.1 Enter Search and Purge Criteria ................................................ 35
    6.1.1 Quick Search ....................................................................... 36
    6.1.2 Search Rules By ................................................................. 36
    6.1.3 Purge Rule Log ....................................................................... 36
  6.2 View the Executed Rule List ......................................................... 36
  6.3 View Rule Execution Details ....................................................... 37

**Index** ......................................................................................................................................................... 39
List of Figures

Figure 1–1: How Developers and Business Analysts Work with Business Rules ........................................ 3
Figure 4–1: BPL Business Process Containing a <rule> Activity ................................................................. 26
Figure 5–2: HL7 Message Routing Rule — Demo.HL7.MsgRouter.XYZRoutingRule ...................... 31
Figure 5–3: General Message Routing Rule — Demo.HL7.MsgRouter.AlertRule ........................... 32
Figure 5–4: Virtual Document Routing Rule — Demo.HL7v3.Rule.RouteAndTransform .................. 33
List of Tables

Table 2–1: Business Rule Editor Icons .................................................................................................... 9
Table 3–1: Editors Associated with Rule Properties ............................................................................... 13
Table 3–2: Expression Editor Icons ..................................................................................................... 16
Table 3–3: Examples of Valid Expressions for Business Rules ............................................................. 20
Table 4–1: Attributes for the BPL Element <rule> .............................................................................. 23
About This Book

This book explains how to define business rules that direct business process logic at decision points.

The book contains the following chapters:

- About Business Rules
- Introduction to the Rule Editor
- Creating and Editing Rule Sets
- Invoking a Rule from BPL
- Example Rule Classes
- Viewing the Business Rule Log

There is also a detailed table of contents.

The following books provide related information:

- *Developing Ensemble Productions* describes how to generate the class code, XML, and custom software elements that may comprise an Ensemble solution.

- *Managing Ensemble* describes how to use the browser-based Management Portal.

If you are using an Ensemble production to route and transform messages in an Electronic Data Interchange (EDI) format such as HL7, X12, or ASTM, the following books are also helpful:

- *Ensemble HL7 Version 2 Development Guide* provides full details about routing rule sets, using HL7 Version 2 message routing as its example. See the “Routing Rule Sets for HL7” section in the chapter “Elements of a Routing Production.”


- *Ensemble X12 Development Guide* explains how to add X12 interfaces to a routing production.

Business rules allow non-technical users to change the behavior of Ensemble business processes at specific decision points. You can change the logic of the rule instantly, using the Ensemble Rule Editor in the Management Portal. There is no need for programming or diagramming skills to change the rule, and there is no need to modify or compile production code for changes to take effect. The following figure shows how business rules work.

*Figure 1–1: How Developers and Business Analysts Work with Business Rules*
Suppose an enterprise runs an Ensemble production that processes loan applications consistently across an international enterprise. The decision process is consistent worldwide, but banks must adjust the local acceptance criteria from country to country. Business rules support this division of responsibility as follows:

1. The developer of the business process identifies a decision point, by naming the business rule that will make the decision on behalf of the business process. The developer leaves a placeholder for that business rule in the Business Process Language (BPL) code by invoking the Business Process Language (BPL) element <rule>. The <rule> element specifies the business rule name, plus parameters to hold the result of the decision and (optionally) the reason for that result. Suppose we call this rule LoanDecision.

2. Wherever the <rule> element appears in a BPL business process, a corresponding rule definition must exist within the production. A user at the enterprise, typically a business analyst, may define the rule using a browser-based online form called the Ensemble Rule Editor. This form prompts the user for the simple information required to define the business rule called LoanDecision. Ensemble saves this information in its configuration database.

   Any enterprise user who is familiar with the Ensemble Rule Editor and who has access to it in the Management Portal can modify the rule definition. Modifications are simply updates to the database and can be instantly applied to a production while it is running. Therefore, it is possible for business analysts at various regional locations to run the Ensemble Rule Editor to modify their copies of the rule to provide different specific criteria appropriate to their locales.

3. At runtime, upon reaching the BPL <rule> statement the business process invokes the rule named LoanDecision. The rule retrieves its decision criteria from the configuration database, which may be different at different locales. Based on these criteria, the rule returns an answer to the business process. The business process redirects its execution path based on this answer.

4. For ongoing maintenance purposes, the business process developer need not be involved if a rule needs to change. Any rule definition is entirely separate from business process code. Rule definitions are stored in the Ensemble configuration database as classes and are evaluated at runtime. Additionally, rule definitions can be exported and imported from one Ensemble installation to another.

   In this way, enterprise users such as business analysts can change the operation of the business process at the decision point, without needing the programming expertise that would be required to revise the BPL or class code for the business process.

Ensemble offers four types of business rule definition:

- **General Business Rule**
- **HL7 Message Routing Rule**
- **General Message Routing Rule**
- **Virtual Document Message Routing Rule**

Each type has an associated rule assist class that controls how the editor works and provides a Rules Assistant in the right pane while you are editing.

You create rules for a definition that are grouped into rule sets. Each rule definition has one or more rule sets. Each rule set is associated with a beginning and ending effective date and time. Each time a process invokes a rule, one and only one rule set is executed.

The remaining chapters in this book describe how to define business rules including how to create and use rule sets using the Ensemble Rule Editor as well as how to invoke rules using BPL and using business process routing engines.
1.1 Rules as Classes

The [Ensemble] > [Ensemble Rule Editor] provides a structured way for enterprise business analysts to shape the logical decisions made by a business process, without needing any programming skills.

However, a business process developer can instead work with business rule definitions as classes, using Studio. The following figure shows this format.

```xml
/// Business rule responsible for mapping an input location
/// to "F" for Fahrenheit or "C" for Celsius temperature scale.
{
  Parameter RuleAssistClass = "Ens.Rule.GeneralBusinessRuleAssist";

  XData RuleDefinition [ XMLNamespace = "http://www.intersystems.com/rule" ]
  {
      <ruleSet name="" effectiveBegin="" effectiveEnd="">
        <rule name="" disabled="false">
          <when condition=""(Location=&quot;US&quot;)||(Location=&quot;USA&quot;)">
            <return>"F"</return>
          </when>
          <otherwise>
            <return>"C"</return>
          </otherwise>
        </rule>
      </ruleSet>
    </ruleDefinition>
  }
}
```

You can open a business rule as a class in Studio, edit the document, and save the changes. Changes saved in Studio are immediately visible in the Ensemble Rule Editor. If you do not see your changes, simply refresh the page.

**Package Mapping Rule Classes**

Since they are classes, you can map rules to other namespaces. If you do map rule classes, you must recompile all the mapped rule classes in each namespace where you use them to insure the local metadata is available in each namespace.

If you are using rules that have been upgraded from legacy rules, you may encounter compile errors due to rule aliases not being unique. You must fix these compiler errors manually.

For details, see the “Package Mapping” section of the “Packages” chapter of *Using Caché Objects*. 
Introduction to the Rule Editor

This chapter introduces the Rule Editor in the Management Portal. It is divided into the following sections:

- Business Rule List
- Business Rule Editor

2.1 Business Rule List

The [Ensemble] > [Business Rule List] page displays a list of the business rule classes defined in the active Ensemble namespace. Navigate to this page from the Business Rules item of the Ensemble List menu.

Select a rule class to be the target of one of the following commands in the ribbon bar:

- **Edit** — Click to change or view the rule definition using the Business Rule Editor.
- **Delete** — Click to permanently delete the rule definition class.
- **Export** — Click to export the selected rule class as an XML file.
- **Import** — Click to import an XML file into a rule class.

You can also export and import rule classes as you do any other class in Ensemble. You can use the [System] > [Globals] page of the Management Portal or use the Export and Import commands on the Tools menu in Studio.

2.2 Business Rule Editor

The [Ensemble] > [Ensemble Rule Editor] page is where you create and edit business rule class definitions for all types of business rule. The page opens with the last rule you had open in the namespace. The tab at the left of the title bar contains the name of the business rule definition class. If this is the first time on the page for this namespace, the working pane is empty and you must either create a new rule or open an existing one.

The “Creating and Editing Rule Sets” chapter describes the details of how to use the editor to define business and routing rules; the rest of this chapter describes how the user interface works with the Ensemble rule structure.

The ribbon bar of the Ensemble Rule Editor page contains the following commands:

- **New** — Click to launch the Business Rule Wizard to create a new business rule definition.
General Tab

Once you have a rule definition in the working pane you see tabs of information. The general tab contains the summary information for the rule definition:

**Description**

Class description of the rule definition and its purpose.

**Rule Type / Rule Assist Class**

Each rule type has an associated rule assist class which controls the constraints of the rule and provides information in the right pane of the page to guide you when editing rules.

The table shows the four rule types and their associated Ens.Rule.Assist class:

<table>
<thead>
<tr>
<th>Rule Type</th>
<th>Rule Assist Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Message Routing Rule</td>
<td>EnsLib.MsgRouter.RuleAssist</td>
</tr>
</tbody>
</table>

**Context Class**

The class that contains the information to tell the Ensemble Rule Editor which object properties to offer as choices in the Value editor while you are editing a rule. For general rules, it is generated from the business process BPL class and ends in .Context. For routing rules without a BPL process, it is usually the routing engine business process class.

**Rule Alias**

This field may have a value if a rule was created in older releases before Ensemble stored rule definitions as classes. When the rule is upgraded, if the old name does not conform to class naming conventions, it becomes the alias. You can use this field for continuing such naming conventions, but if you are creating a new rule you do not need to use this field.

**Production Name**

The name of the production in which you expect to use this routing rule.

**Rule Set List**

List of Rule Sets with the following information:

- **Rule Set Name** — Name to identify the particular set.
- **Beginning Date and Time** — The time from which the rule becomes active. The exact time is included in the active interval. The format is **YYYY-MM-DDTHH:MM:SS**. The time portion is optional and defaults to **00:00:00**.
- **Ending Date and Time** — The time when the rule stops being active. The exact time is excluded from the active interval. The format is *YYYY-MM-DDTHH:MM:SS*. The time portion is optional and defaults to 24:00:00

**Rule Set Tabs**

Each rule set has its own tab for editing its list of rules. For details, see the “Creating and Editing Rule Sets” chapter.

**Editor Icons**

Both tabs contain following set of icons. The following icons are available to edit the rule definition, rule sets, rules, and clauses within a rule set:

- ![Up](icon.png)
- ![Down](icon.png)
- ![Add](icon.png)
- ![Delete](icon.png)
- ![Undo](icon.png)
- ![Redo](icon.png)
- ![Function](icon.png)

The specific action may differ depending on the entity you are editing; the following table describes the action of each icon in general.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="icon.png" alt="Up" /></td>
<td>Click the <em>Up</em> icon to move the selected item up in the list.</td>
</tr>
<tr>
<td><img src="icon.png" alt="Down" /></td>
<td>Click the <em>Down</em> icon to move the selected item down in the list.</td>
</tr>
<tr>
<td><img src="icon.png" alt="Add" /></td>
<td>Click the <em>Add</em> icon to add the type of item you choose from the list or beneath the icon in the Rule Assistant.</td>
</tr>
<tr>
<td><img src="icon.png" alt="Delete" /></td>
<td>Click the <em>Delete</em> icon to delete the item next to it.</td>
</tr>
<tr>
<td><img src="icon.png" alt="Undo" /></td>
<td>Click the <em>Undo</em> icon to undo the last change.</td>
</tr>
<tr>
<td><img src="icon.png" alt="Redo" /></td>
<td>Click the <em>Redo</em> icon to redo the last change that was undone.</td>
</tr>
<tr>
<td><img src="icon.png" alt="Function" /></td>
<td>Click the <em>Function</em> icon to open the associated editor for the selected action.</td>
</tr>
</tbody>
</table>

If an action is not available, its icon appears dimmed.

**Rule Assistant**

You can hide or show the Rule Assistant using the double-arrow in the right pane of the Ensemble Rule Editor. When you are editing a rule set, the expanded *Rule Assistant* pane provides you with help throughout the editing process. It describes the item you have selected and offers you a list of options based on your assist class.

**2.2.1 Business Rule Wizard**

This wizard helps you create a new business rule definition based on the `Ens.Rule.Definition` class with an XData block named `RuleDefinition`. Enter values for the following fields:

**Package**

Enter a package name or use the arrow to select an existing package name.
Name

Enter the name of the business rule class.

Alias

(Optional) Enter the alias name for this rule. Do not use any of the following characters:

; , : | ! * - $ '

Generally, this exists for some rules created in older releases before Ensemble stored rule definitions as classes and for continuing such naming conventions. If you are creating a new rule you do not need to use this field.

Description

(Optional) Enter a description for this rule definition. This becomes the class description.

Type

Enter one of four rule types:

- General Business Rule
- HL7 Message Routing Rule
- General Message Routing Rule
- Virtual Document Message Routing Rule

Each rule type has an associated rule assist class which provides information in the right pane of the page to guide you in entering rules and controls which options the editor presents.

Context Class

(Optional)

This field tells the Ensemble Rule Editor which object properties to offer as choices in the Value field when you are editing a rule. For general rules, this class is generated from the BPL business process class that invokes the <rule>. The naming convention of the class is the business process class name plus the .Context extension, as in Demo.ZenService.Bproc.WeatherReport.Context.
Creating and Editing Rule Sets

This chapter describes how to use the Ensemble Rule Editor to develop rule sets.

There are two types of rule set:

- **General business rule set** — A list of rules that are evaluated sequentially until one of them is found to be true. The “true” case determines the next action of the business process that invoked the rule. If none of the rules is true, the rule set returns a default value. This is the type of rule that you invoke using the BPL `<rule>` element.

- **Routing rule set** — A rule set for use in message routing productions. Based on the type and contents of incoming messages (constraint), the routing rule set determines the correct destination for each message and how to transform the message contents prior to transmission. You use a routing engine business process to invoke this rule set.

**Rule Set Properties**

When you add a rule set, you see three property fields that you can use to distinguish rule sets in a rule definition: *Name*, *Effective begin date and time*, and *Effective end date and time*. These are the values the rule set list shows on the general tab.

Most business rule definitions have only one rule set that is always in effect. You can, however, have more than one version of a rule defined and become active at different times using a beginning and ending effective date and time. Each time a process invokes a rule, one and only one rule set is executed.

You can add a rule set from the general tab by clicking the add icon with the rule set list selected. You can then begin to edit the rule set by clicking on its tab or double-clicking its row in the list. When you are editing a rule set, you can click the icons near the top of the rule set tab or you can click the ones (or in the case of the add icon, the labeled rectangles beneath) in the expanded *Rule Assistant* pane.

As you become more familiar with the rule set editor and the rule sets you are developing, you may find it unnecessary to view the property names throughout the display of the rule set. You can toggle the viewing of property names by clicking the green square in the top right of the editor pane.

Also throughout the editing process, if a property does not contain a valid value you see a small red circle containing an exclamation point at the top right of the property box. If you double-click this warning mark, a helpful error message displays.

The following sections describe the editing tasks involved in creating a rule set:

- Adding Rules
- Adding Actions
- Using the Associated Editors
- Adding Business Rule Notification
3.1 Adding Rules

Each rule set contains one or more rules that you define to satisfy a specific function in a business process.

Rule Properties

When you add a rule, you see the following properties:

Name
You can give this rule an optional name to help you identify it.

Internally, Ensemble names the rules in sequential order in the form rule#n. If you enter a value in the Name property, it appears in the class definition and also appears in parentheses next to the internal rule name in the rule log. The value of n changes if you reorder the rules in a rule set.

Disabled
You can double-click this item to toggle between disabling and enabling the rule. A value of true means the rule is disabled and therefore skipped when the rule set is executed.

Constraint
(For routing rules only) — The constraint property distinguishes a routing rule. As a message makes its way through the rule set, if it matches the constraint you define for the rule, that rule logic is executed. See the “Using the Rule Constraint Editor” section for details on defining the constraint.

Each rule consists of a series of one or more when clauses and an optional otherwise clause, along with some optional actions. When you add a rule, the editor starts you with a when clause and if it is a routing rule, it also provides a return action for the clause.

Some general considerations to keep in mind when developing rules in your rule set:

• Once the execution through a rule set encounters a return action, the execution of the rule set ends and returns to the business process that invoked the rule definition class.

• You can control the execution of more than one rule in a rule set by omitting the returns. In other words, if you want to check all rules, do not provide a return action within any of the rule clauses. You may then provide a value in a return action at the end of the rule set for the case where no rule clauses evaluate to true.

• When a rule contains multiple when clauses, only the actions indicated by the first when that evaluates to true are performed. You can use an otherwise clause to perform an action if no when conditions are true.

• Each when clause has a condition property. A common design for a general business rule set is one that contains one rule with a series of when conditions and returning a value depending on which condition is true. If you want to return a default value if none of the conditions is true, you can use the otherwise clause with a return.

• A common design for a routing rule set is one that contains several rules each with a different constraint defined and each with one when clause describing how and where to route the message that matches the constraint.

3.2 Adding Actions

Every clause within a rule can have zero or more actions associated with it. Actions are executed if and only if the associated when condition is true. You can add the following actions to a rule set or a when or otherwise clause within a rule:
You can add some actions at the rule set level, but they do not always logically make sense. You should contain most actions within the `when` clauses of rules. A time when it may make sense to provide an action outside of these clauses is to set a default return value if no rules are executed in a rule set.

In addition, you can add the following actions to a routing rule:

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>send</strong></td>
<td>Sends the message to a particular target after optionally transforming it. See the “Selecting the Transformation and Target of a Send Action” section for details.</td>
</tr>
<tr>
<td><strong>delete</strong></td>
<td>Deletes the current message.</td>
</tr>
<tr>
<td><strong>delegate</strong></td>
<td>Delegates the message to a different rule.</td>
</tr>
</tbody>
</table>

### 3.3 Using the Associated Editors

When you select a property of any of the items in a rule definition, $\text{fx}$ (the function icon) becomes enabled if the property has an associated editor. The following table shows which properties exist for a item, clause, or action and which editor opens when you double-click the property or click $\text{fx}$.

**Table 3–1: Editors Associated with Rule Properties**

<table>
<thead>
<tr>
<th>Item</th>
<th>Property</th>
<th>Associated editor or edit action</th>
</tr>
</thead>
<tbody>
<tr>
<td>rule set</td>
<td>name</td>
<td><em>Enter text.</em></td>
</tr>
<tr>
<td>rule set</td>
<td>effective begin</td>
<td>Date and Time Selector</td>
</tr>
<tr>
<td>rule set</td>
<td>effective end</td>
<td></td>
</tr>
<tr>
<td>rule</td>
<td>name</td>
<td><em>Enter text.</em></td>
</tr>
<tr>
<td>rule</td>
<td>disabled</td>
<td>Double-click to toggle between true and false.</td>
</tr>
<tr>
<td>rule</td>
<td>constraint</td>
<td>Rule Constraint Editor</td>
</tr>
<tr>
<td>when</td>
<td>condition</td>
<td>Expression Editor</td>
</tr>
<tr>
<td>assign</td>
<td>property</td>
<td>Enter the name of a context property that is the target of this assignment. This must be a property in an execution context object.</td>
</tr>
<tr>
<td>assign</td>
<td>value</td>
<td>Expression Editor</td>
</tr>
<tr>
<td>return</td>
<td>value (general rule set only)</td>
<td>Expression Editor</td>
</tr>
</tbody>
</table>
The otherwise clause and delete action have no properties to edit.

### 3.3.1 Using the Rule Constraint Editor

Routing rules have a constraint property you use to determine which messages to route through which rules. Use this editor to configure the rule constraint values, which can be made up of the following properties:

**Source**

Configuration name of one of the following items:

- A business service (for a routing interface)
- A message routing process (if another rule chains to this routing rule set)

Click (...) next to the Source field to invoke the Production Configuration Item Selector which displays a list of possible source items in the production you indicate on the general tab. If you have not yet prepared the item you need as a Source, you may leave this field blank and return to it when the item is ready.

**Message Class**

Identifies the Ensemble message object that is being routed by this rule. The value of this field depends on the routing rule type:

- **HL7 Message Routing Rule** — Defaults to EnsLib.HL7.Message; you do not have the option of entering this property in a new rule definition.
- **General Message Routing Rule** — Click ... next to the Message Class field to invoke the Finder Dialog to select the appropriate message class. You can choose the category of message class to narrow your choices.
- **Virtual Document Message Routing Rule** — Choose from the list of defined virtual document classes.

The following fields in the editor apply only when the you are editing an HL7 or virtual document routing rule class, such as X12 or ASTM. For general message routing rules, you are finished entering the constraint fields.

**Schema Category**

Identifies the message category for the particular message class:

- **HL7 Message Routing Rule** — Choose from the built-in schema category list such as 2.1, 2.2, 2.3, 2.3.1, 2.4, 2.5, 2.5.1, 2.6, or the name of a custom schema definition.
- **Virtual Document Message Routing Rule** — Choose from the list of category types defined for your chosen virtual document class; they could be built-in or from an imported custom schema.

**Document Name**

Identifies the message structure; the acceptable values depend on the message class.
• **HL7 Message Routing Rule** — The HL7 message structure that the source application identifies in the MSH:9 field, such as ADT_A08 or ORM_O01. To allow easy retrieval, this MSH:9 value resides in the EnsLib.HL7.Message property called *Name*.

• **Virtual Document Message Routing Rule** — Choose from the list of category types defined for your chosen virtual document class; they could be built-in or imported.

Enter more than one value in the **Document Name** text entry field. This causes the rule to match any of the specified **Document Name** values, and no others.

If you leave any of the fields blank, Ensemble considers *all* values to be a match for that rule.

The **Constraint Editor** behaves somewhat differently when you are editing a rule set converted from a version older than Ensemble 2012.1. You may see the **Schema Doc Type** field with a box to select an item to append to the list.

**Schema Category** and **Schema Doc Type** represent the actual HL7 message structure. These values reside in the EnsLib.HL7.Message property called DocType. DocType is a two-part string separated by a colon, such as 2.4:ADT_A08.

• At left is the **Schema Category**. This is the name of a built-in schema category such as 2.1, 2.2, 2.3, 2.3.1, 2.4, 2.5, 2.5.1, 2.6, or the name of a custom schema definition.

• At right is the **Schema Doc Type**. This is an HL7 message structure within the identified schema, such as ADT_A08 or ORM_O01.

### 3.3.2 Using the Production Configuration Item Selector

This editor helps you choose a configuration item as a source of a message or a routing target of a message. You choose from a list of production configuration items defined in the production you enter in the general tab. If you do not choose a production in the general tab of your rule definition, you receive the following warning when you invoke this editor:

No production name has been specified in the General tab of your rule. Be careful and ensure that your chosen target(s) exist in your production.

In this case you choose from a list of production configuration items defined in the current namespace. While developing your production rules, be careful to verify the names of your configuration items.

### 3.3.3 Selecting the Transformation and Target of a Send Action

When you add a **send** action, you also enter the following properties:

• **Transform** — *(optional)* To transform the message before sending it to the **Target**, enter the full package and class name of one or more DTL data transformations. You can double-click the **Transform** field to invoke the **Data Transform Selector** to choose one or more defined data transformations in the namespace.

Multiple data transformations are chained in the order in which they appear, from left to right.

• **Target** — Enter the configured name of one or more of the following production items:
  
  – A business operation, to route the message to an external application
  
  – A routing process, to chain to another routing rule set

  Double-click the **Target** field to invoke the **Production Configuration Item Selector** to choose one or more production configuration items.

If you enter items for these fields that do not exist yet, make sure to verify you have entered the correct name when the production does contain them.
3.3.4 Using the Expression Editor

When you select a condition or a value and click \( \Rightarrow \), you invoke the Expression Editor. There are four properties that activate the expression editor:

- **when condition** — See Defining When Conditions for details.
- **assign value**
- **return value** of a general business rule
- **trace value** — This is the text for the trace message. It can be a literal text string or an expression to be evaluated. If an expression, it must use the scripting language specified by the `language` attribute of the containing `<process>` element.

When defining an expression, you can nest several conditions by using the icons described in the following table.

**Table 3–2: Expression Editor Icons**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>Click to move the selected node up in the expression.</td>
</tr>
<tr>
<td>↓</td>
<td>Click to move the selected node down in the expression.</td>
</tr>
<tr>
<td>←</td>
<td>Click to merge the selected node with the parent node.</td>
</tr>
<tr>
<td>( \Rightarrow )</td>
<td>Click to choose from a list of operators of which to make the selected node an operand.</td>
</tr>
<tr>
<td>( \Rightarrow )</td>
<td>Click to choose from a list of Ensemble functions to make the selected node an argument of the selected function.</td>
</tr>
<tr>
<td>+</td>
<td>Click to add a sibling node.</td>
</tr>
<tr>
<td>×</td>
<td>Click to delete the selected node.</td>
</tr>
</tbody>
</table>

If an action is not available, its icon appears dimmed. As you add conditions and values to the expression diagram, you see the text of the expression in the blue bar at the top of the editor.

The following sections provide greater detail for entering expression values:

- Expression Values
- Expression Operators
- Expression Functions
- Expression Examples

3.3.4.1 Defining When Conditions

Within a rule definition, a condition consists of two values and a comparison operator between these values. For example:

\[
\text{Amount} \ <= \ 5000
\]
If a condition is not true, it is false. There are no other possible values for a condition. This type of result is called a \textit{Boolean} result. Ensemble stores a Boolean result either as the integer value 1 (if true) or 0 (if false). In most cases you do not need to be concerned with this internal representation; however, when using the constraint property in a routing rule, you may want to always execute the associated \textit{when} clause when the constraints are satisfied. In this case, enter a value of 1 in the \textit{when} condition property.

There can be more than one condition within a rule. If so, all of the conditions must be evaluated and compared before the rule (as a whole) can be found to be true or false. The logic between each condition is controlled using \textit{AND} or \textit{OR} operators. For example:

\begin{verbatim}
IF Amount <= 5000
AND CreditRating > 5
OR CurrentCustomer = 1
\end{verbatim}

For this example, the \textbf{Expression Editor} dialog appears as follows:

The preceding rule has three conditions: Amount <= 5000, CreditRating > 5, CurrentCustomer = 1. Each of these conditions could be true or false. All of these conditions are evaluated before the \textit{AND} or \textit{OR} operators come into play.

\textit{AND} and \textit{OR} operate on true and false values only. The operator is positioned between two Boolean values, and returns a single Boolean result based on those two values, as follows.

\begin{tabular}{|c|l|}
\hline
\textbf{Operator} & \textbf{Result is true when...} \\
\hline
\textit{AND} & Both values are true. \\
\hline
\textit{OR} & At least one of the values is true. If one of the values is false and the other is true, and the result (as a whole) is still true. \\
\hline
\end{tabular}

If there are multiple \textit{AND} or \textit{OR} operators within a rule, \textit{AND} operators take precedence over \textit{OR} operators. This means that all \textit{AND} operations in the rule are performed first. Only then are the \textit{OR} operations considered. Thus, logic such as this:

\begin{verbatim}
IF Amount <= 5000
AND CreditRating > 5
OR CurrentCustomer = 1
AND CreditRating >= 5
\end{verbatim}

is handled as follows:

\begin{verbatim}
IF (Amount <= 5000 AND CreditRating > 5)
OR (CurrentCustomer = 1 AND CreditRating >= 5)
\end{verbatim}

For this example, the \textbf{Expression Editor} dialog appears as follows:
The preceding rule is true if anyone requests an amount less than 5,000 and has a credit rating better than average. The rule is true for any current bank customer requests any amount and has a credit rating greater than or equal to the average. Both conditions may be true, or only one or the other of them may be true. If both conditions are false, then the rule (as a whole) is false.

In detail, the preceding rule works as follows:

1. **This AND expression:**
   
   IF \(\text{Amount} \leq 5000\) AND \(\text{CreditRating} > 5\)
   
   Gives a result, true or false. Call this result “SafeBet.”

2. **This AND expression:**
   
   IF \(\text{CurrentCustomer} = 1\) AND \(\text{CreditRating} \geq 5\)
   
   Gives a result, true or false. Call this result “KnownEntity.”

3. **Once the AND operations in the rule have completed, the OR operation begins, as follows:**
   
   IF SafeBet is true OR KnownEntity is true

4. From what we know about OR logic, we know that this rule (as a whole) is true if the customer is a SafeBet but not a KnownEntity, or if the customer is not a SafeBet but is a KnownEntity. Additionally, this rule is true if the customer is both a SafeBet and a KnownEntity.

### 3.3.4.2 Expression Values

Within a condition or the assign, return, or trace actions, the values can be any of the following items:

- A numeric value (integer or decimal), such as 1.1 or 23000.
- A string value, which must be enclosed in double quotes: "NY"
- If your production invokes the rule from a BPL business process using the <rule> element, you can specify a property in the general-purpose, persistent variable context, which is defined using the <context> and <property> elements in BPL. A property name is case-sensitive, and must not be enclosed in quotes, as in:
• An expression using various permitted operators, literal values, and properties of context, as for example:

\[((2+2)*5)/154.3\]
"hello" & "world"
\.Age \* 4 \n\(((x=1) \mid (x=3)) \&\& (y=2)\)

• A built-in Ensemble function such as \texttt{Min()}, \texttt{Max()}, \texttt{Round(n,m)}, or \texttt{SubString()}. The function name must include parentheses. It must also include any input parameters, such as the numeric values \textit{n} and \textit{m} for \texttt{Round}. If there are no input values for the function, then the open and close parentheses must be present, but empty.

When you want to insert a value into a field on the rule set tab, you may type it into the text box directly. If the syntax is incorrect or inappropriate for the type of data expected for that field, you see the red error symbol when you try to save your changes.

If you correctly entered a \textit{Context Class} in your rule definition, when you select a property in the Expression Editor, the text box offers choices of properties from the business process execution context of the identified BPL business process \textit{Context Class}.

### 3.3.4.3 Expression Operators

In the Expression Editor, you can click \(\texttt{op} \) to choose from a list of operators of which to make the selected node an operand.

You may use the following arithmetic operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Plus (binary and unary)</td>
</tr>
<tr>
<td>–</td>
<td>Minus (binary and unary)</td>
</tr>
<tr>
<td>*</td>
<td>Times</td>
</tr>
<tr>
<td>/</td>
<td>Divide</td>
</tr>
</tbody>
</table>

You may use the following logical operators, which return an integer value of 1 (true) or 0 (false):

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
<th>Expression is true when...</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{AND} (&amp;&amp;)</td>
<td>And</td>
<td>Both values are true.</td>
</tr>
<tr>
<td>\texttt{OR} (</td>
<td></td>
<td>)</td>
</tr>
<tr>
<td>!</td>
<td>Not (unary)</td>
<td>The value is false.</td>
</tr>
<tr>
<td>=</td>
<td>Equals</td>
<td>The two values are equal.</td>
</tr>
<tr>
<td>!=</td>
<td>Does not equal</td>
<td>The two values are \textit{not} equal.</td>
</tr>
<tr>
<td>&gt;</td>
<td>Is greater than</td>
<td>The value to the left of the operator is \textit{greater} than the value to the right of the operator.</td>
</tr>
<tr>
<td>&lt;</td>
<td>Is less than</td>
<td>The value to the left is \textit{less} than the value to the right.</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Is greater than or equal to</td>
<td>The value to the left is greater than the value to the right, \textit{or} if the two values are equal.</td>
</tr>
</tbody>
</table>
Expression is true when...

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
<th>Expression is true when...</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=</td>
<td>Is less than or equal to</td>
<td>The value to the left is less than the value to the right, or if the two values are equal.</td>
</tr>
<tr>
<td>[</td>
<td>Contains</td>
<td>The string contains the substring to the right. Pattern matching for Contains is exact. If the value at left is “Hollywood, California” and the value at right is “od, Ca”, there is a match, but a value of “Wood” does not match.</td>
</tr>
</tbody>
</table>

You may use the following string operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;</td>
<td>Concatenation operator for strings.</td>
</tr>
<tr>
<td>_</td>
<td>Binary concatenation to combine string literals, expressions, and variables.</td>
</tr>
</tbody>
</table>

When more than one operator is found in an expression, the operators are evaluated in the following order, from first to last:

1. Any of these logical operators: ! = != < > <= >=
2. Multiplication and division: * /
3. Addition and subtraction: + –
4. String concatenation: &
5. Logical AND: &&
6. Logical OR: ||

### 3.3.4.4 Expression Functions

Within a rule definition, an expression can include a call to one of the Ensemble utility functions. These include mathematical or string processing functions such as you may be accustomed to using in other programming languages.

In the Expression Editor, you can click \( \mathbb{f} \) to choose from a list of functions of which to make the focused node an argument.

For a list of the available utility functions and the proper syntax for using them in business rules or DTL data transformations, see the “Utility Functions” section in the “Creating a New Production” chapter of Developing Ensemble Productions. The section also explains how to extend the list of built-in utility functions by adding custom code.

### 3.3.4.5 Expression Examples

Within a rule definition, an expression is a formula for combining values and properties to return a value. For example:

**Table 3–3: Examples of Valid Expressions for Business Rules**

<table>
<thead>
<tr>
<th>Expression</th>
<th>Computed value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((2+2)*5)/154.3</td>
<td>0.129617628</td>
</tr>
<tr>
<td>&quot;hello&quot; &amp; &quot;world&quot;</td>
<td>&quot;helloworld&quot;</td>
</tr>
<tr>
<td>Age * 4</td>
<td>When Age is a context property (a property in the general-purpose, persistent variable context, which is defined using the &lt;context&gt; and &lt;property&gt; elements in BPL) and has the numeric value 30, the value of this expression is 120.</td>
</tr>
<tr>
<td>1+2.5*2</td>
<td>6</td>
</tr>
</tbody>
</table>
### Adding Business Rule Notification

It is possible for you to set up rule notification, so that specific actions are taken each time a rule is fired. Unlike most activities related to rules, notification requires programming. You must subclass Ens.Rule.Notification and override the `%OnNotify` method in the subclass. The signature of this method is:

```lisp
ClassMethod %OnNotify(pReason As %String, pRule As Ens.Rule.RuleDefinition) As %Status
```

Possible `pReason` values are:

```
Using Business Rules with Ensemble
21
```
Creating and Editing Rule Sets

- BeforeSave
- AfterSave
- Delete

At runtime, the Ensemble framework automatically finds your subclass of Ens.Rule.Notification and uses the code in %OnNotify to determine what to do upon firing a rule.
4

Invoking a Rule from BPL

A business process developer leaves a placeholder for a business rule by invoking the Business Process Language (BPL) element `<rule>`. The `<rule>` element specifies the business rule name, plus parameters to hold the result of the decision and (optionally) the reason for that result.

The parameters for the `<rule>` element can include any property in the general-purpose execution context variable called `context`. Therefore, a typical design approach, for a business process that invokes a rule, is to ensure that the business process accomplishes the following:

1. Provides `<property>` and `<context>` elements so that the `context` object contains properties with appropriate names and types.
   For example, if the rule determines eligibility for a state education loan, you might add properties such as Age, State, and Income.

2. Gathers values for the properties in whatever way you wish, for example by sending requests to business operations or business processes, and as responses return, assigning values to the properties in `context`.

3. Provides a `<rule>` element that invokes a business rule that returns an answer based on these input values.

4.1 Using the BPL `<rule>` Element

The `<rule>` element invokes a business rule from a business process. When a `<rule>` executes, it invokes its associated business rule (named by the `rule` attribute) and gets its response immediately (in the same manner as a `<code>` or `<assign>` activity).

The following table lists the required and optional attributes for the `<rule>` element.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>name</strong></td>
<td>Required. The name of the <code>&lt;rule&gt;</code> element.</td>
<td>A string of one or more characters.</td>
</tr>
<tr>
<td><strong>rule</strong></td>
<td>Required. The name of the business rule to be executed. This must be a valid rule within the namespace; see the section “Identifying the Rule.” If the rule is not defined or otherwise cannot be found at runtime, the rule will return a default value of “” (an empty string).</td>
<td>A string of one or more characters.</td>
</tr>
</tbody>
</table>
### 4.1.1 Identifying the Rule

When you use the `<rule>` element in BPL, the value of the `rule` attribute can be either of the following:

- A simple **Rule Name**:
  
  MyRule

- A full **Package Name** plus **Rule Name** combination:
  
  MyClassPackage.Organization.Levels.MyRule

If any `<rule>` element identifies a simple **Rule Name**, Ensemble automatically prepends a **Package Name** that is equal to the full package and class name of the BPL business process that contains that `<rule>` element. That is:

`BPLFullPackageAndClassName.MyRule`

This combination must identify a valid rule within the namespace, or the return value of the `<rule>` will be a null string.

### 4.1.2 BPL `<rule>` Example

The following is a BPL excerpt showing the use of the `<rule>` activity with a `<switch>` element to process the results from the rule:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>resultLocation</code></td>
<td>Optional. The location in which to store the return value of the rule. Typically this is a property within the business process execution context; that is, <code>context.MyValue</code>.</td>
<td>The name of a valid property and object, usually within the business process execution context.</td>
</tr>
<tr>
<td><code>reasonLocation</code></td>
<td>Optional. The location in which to store the reason returned by the rule. The rule reason is a string indicating why a business rule reached its decision. For example, “Rule 1” or “Default”. If the business rule is empty (for example, it is a rule set that contains no rules) then the reason given for the decision is “Rule Missing.”</td>
<td>A string of zero or more characters.</td>
</tr>
<tr>
<td><code>disabled</code></td>
<td>Optional. You can temporarily disable the <code>&lt;rule&gt;</code> element by setting its <code>disabled</code> attribute to 1 (true). To re-enable the <code>&lt;rule&gt;</code> element, either remove the <code>disabled</code> attribute or set it to 0 (false).</td>
<td>A Boolean value: 1 (true) or 0 (false).</td>
</tr>
<tr>
<td><code>xpos</code></td>
<td>Optional. Sets the x coordinate of the graphic that represents this element in BPL diagrams. Ignored by the BPL compiler.</td>
<td>A positive integer.</td>
</tr>
<tr>
<td><code>ypos</code></td>
<td>Optional. The y coordinate.</td>
<td>A positive integer.</td>
</tr>
<tr>
<td><code>xend</code></td>
<td>Optional. If the graphic that represents this element has two icons (start and end), then <code>xend</code> sets the x coordinate for the ending icon. Ignored by the BPL compiler.</td>
<td>A positive integer.</td>
</tr>
<tr>
<td><code>yend</code></td>
<td>Optional. The ending y coordinate.</td>
<td>A positive integer.</td>
</tr>
</tbody>
</table>
The `<rule>` activity in this example returns a Boolean value (true or false) according to Ensemble conventions. That is, an integer value of 1 means true; 0 means false. All rules return a single value, as in this example, but the type need not be Boolean. The single value returned from a rule may be *any literal value* such as an integer number, decimal number, or text string.

**Return Values**

The result and reason for the result are stored in the variables identified by the `resultLocation` and `reasonLocation` attributes, respectively. Usually, these attributes give the names of properties in the `context` variable. This is the general-purpose, persistent variable that you define at the beginning of the BPL business process using `<context>` and `<property>` elements.

**The `<annotation>` Element**

Like other BPL activities, a `<rule>` element can contain the optional `<annotation>` element, which provides a text string describing the activity. If provided, this string appears as a comment in the BPL diagram.

### 4.2 BPL Business Process Designer

A `<rule>` element is an activity in BPL, so you can add or remove it from a BPL diagram using the BPL Business Process Designer. To add a rule, click **Rule** on the **Add Activity** list in the ribbon bar.

Within a BPL diagram, a `<rule>` uses the rounded rectangle shape, like any other activity. Its identifying icon is the $f(x)$ symbol, as shown in the *Temp Scale* box in the following diagram:
The corresponding BPL code looks like this:

```plaintext
/// BPL business process responsible for getting a weather report and providing the results
/// in the appropriate language and temperature scale for the locale.
  /// BPL Definition
  XData BPL [ XMLNamespace = "http://www.intersystems.com/bpl" ] {
    <process language='objectscript' request='Ens.StringRequest'
      <context>
        <property name='Location' type='%String' >
        </property>
        </property>
        <property name='TempScale' type='%Library.String' >
        </property>
        <property name='OutputTemp' type='%String' >
        </property>
        <property name='Language' type='%Library.String' >
        </property>
      </context>
      ...
      <sequence name='Convert Temp' xpos='200' ypos='450' xend='200' yend='800' >
        <annotation>Invoke the TempScale rule. Depending on location the temperature should be
                    reported in Fahrenheit or Celsius. Store the result in the context property TempScale. </annotation>
        <rule name='Temp Scale' rule='TempScale' resultLocation='context.TempScale' xpos='200' ypos='250' >
          <annotation>Invoke the TempScale rule. If the Location is US or USA then the temperature scale is F; otherwise it is C. Put the result in the context property TempScale. </annotation>
        </rule>
        <if name='Convert?' condition='context.TempScale="F"' xpos='200' ypos='350' xend='200' yend='700' >
          26
        </if>
      </sequence>
    </process>
  }
}
```

---

**Figure 4-1: BPL Business Process Containing a <rule> Activity**

![Diagram of BPL Business Process with <rule> Activity]
<annotation>Test whether the context property TempScale is F. If so, do the conversion to Fahrenheit.
</annotation>
<true>
<call name='Convert Min Temp' target='Convert Temperature' async='0' xpos='335' ypos='500' >
<annotation>Call the Convert Temperature business operation to convert the day’s minimum

temperature from Celsius to Fahrenheit. Put the result in the context property OperationReport.
</annotation>
<request type='Demo.ZenService.Msg.ConvertTempRequest' >
<assign property="callrequest.Direction" value="\"CToF\"" action="set" />
</request>
</response>
</call>
<call name='Convert Max Temp' target='Convert Temperature' async='0' xpos='335' ypos='600' >
<annotation>Call the Convert Temperature business operation to convert the day’s maximum

temperature from Celsius to Fahrenheit. Put the result in the context property OperationReport.
</annotation>
<request type='Demo.ZenService.Msg.ConvertTempRequest' >
<assign property="callrequest.Direction" value="\"CToF\"" action="set" />
</request>
<assign property="context.OutputTemp" value="callresponse.OutputTemp" action="set" />
</response>
</call>
</true>
</if>
</sequence>
...
</process>
}
}

This is only a portion of the complete BPL for the process to show how the rule is used. For details about the BPL Business Process editor, see the “Using the Business Process Designer” chapter of Developing BPL Processes.
Example Rule Classes

Ensemble contains various demonstration productions that contain examples of the different rule types. The following sections display an example rule class for each type:

- General Business Rule Example
- HL7 Message Routing Rule Example
- General Message Routing Rule Example
- Virtual Document Message Routing Rule Example

5.1 General Business Rule Example


The following code shows the class definition:

```csharp
/// Business rule responsible for mapping an input location
/// to "F" for Fahrenheit or "C" for Celsius temperature scale.
{
    Parameter RuleAssistClass = "Ens.Rule.GeneralBusinessRuleAssist";
    XData RuleDefinition [ XMLNamespace = "http://www.intersystems.com/rule" ]
    {
        <ruleSet name="" effectiveBegin="" effectiveEnd="">
            <when condition="(Location=="US&quot;)||((Location=="USA&quot;")>
                <return>"F"</return>
            </when>
            <otherwise>
                <return>"C"</return>
            </otherwise>
        </ruleSet>
    </ruleDefinition>
}
```

The following figure shows how the rule definition looks in the Ensemble Rule Editor:
The Demo.HL7.MsgRouter.XYZRoutingRule class is an example of a HL7 message routing business rule. It is part of the Demo.HL7.MsgRouter.Production production in the ENSDEMO namespace.

The following code shows the class definition:

```csharp
/// Routing from the XYZ message source.
{
    Parameter RuleAssistClass = "EnsLib.HL7.MsgRouter.RuleAssistCompatible";
    XData RuleDefinition [ XMLNamespace = "http://www.intersystems.com/rule" ]
    {
        <ruleDefinition alias="" context="EnsLib.HL7.MsgRouter.RoutingEngine">
            <ruleSet name="" effectiveBegin="" effectiveEnd="">
                <rule name="">
                    <constraint name="msgClass" value="EnsLib.HL7.Message"></constraint>
                    <constraint name="docCategory" value="Demo.HL7.MsgRouter.Schema"></constraint>
                    <when condition="(HL7.ParentId!='')">
                        <send transform="" target="Batch_FileOperation"></send>
                        <return></return>
                    </when>
                </rule>
                <rule name="">
                    <constraint name="msgClass" value="EnsLib.HL7.Message"></constraint>
                    <constraint name="docCategory" value="2.3.1"></constraint>
                    <constraint name="docType" value="FHS,BHS"></constraint>
                    <when condition="1">
                        <send transform="" target="Batch_FileOperation"></send>
                        <return></return>
                    </when>
                </rule>
                <rule name="">
                    <constraint name="msgClass" value="EnsLib.HL7.Message"></constraint>
                    <constraint name="docCategory" value="ADT_A08"></constraint>
                    <constraint name="docType" value="ADT_A01"></constraint>
                    <when condition="(Contains(HL7.{PID:PatientName(1).familylastname},""<S>"")) || (StartsWith(HL7.{PID:PatientName(1).familylastname},""Z""))">
                        <send transform="Demo.HL7.MsgRouter.ADTLastNameTransform" target="Other_FileOperation"></send>
                        <return></return>
                    </when>
                </rule>
            </ruleSet>
        </ruleDefinition>
    }
}
```

Using Business Rules with Ensemble
The following figure shows how the rule definition looks in the Ensemble Rule Editor:

![HL7 Message Routing Rule — Demo.HL7.MsgRouter.XYZRoutingRule](image-url)
5.3 General Message Routing Rule Example

The Demo.HL7.MsgRouter.AlertRule class is an example of a general message routing business rule. It is part of the Demo.HL7.MsgRouter.Production production in the ENSDEMO namespace.

The following code shows the class definition:

```csharp
{
    Parameter RuleAssistClass = "EnsLib.MsgRouter.RuleAssist";
    XData RuleDefinition [ XMLNamespace = "http://www.intersystems.com/rule" ]
    {
        <ruleDefinition alias="" context="EnsLib.MsgRouter.RoutingEngine">
            <ruleSet name="" effectiveBegin="" effectiveEnd="">
                <rule name="">
                    <constraint name="msgClass" value="Ens.AlertRequest"></constraint>
                    <when condition="1">
                        <send transform="Demo.HL7.MsgRouter.EmailAlertTransform" target="EMailAlertOperation"></send>
                    </when>
                    <return></return>
                </rule>
            </ruleSet>
        </ruleDefinition>
    }
}
```

The following figure shows how the rule definition looks in the Ensemble Rule Editor:

![Figure 5-3: General Message Routing Rule — Demo.HL7.MsgRouter.AlertRule](image)

5.4 Virtual Document Message Routing Rule Example

The Demo.HL7v3.Rule.RouteAndTransform class is an example of a virtual document message routing business rule. It is part of the Demo.HL7v3.Production.InterfaceEngine production in the ENSDEMO namespace.

The following code shows the class definition:

```csharp
{
    Parameter RuleAssistClass = "EnsLib.MsgRouter.VDocRuleAssistCompatible";
    XData RuleDefinition [ XMLNamespace = "http://www.intersystems.com/rule" ]
    {
```
Virtual Document Message Routing Rule Example

The following figure shows how the rule definition looks in the Ensemble Rule Editor:

![Figure 5-4: Virtual Document Routing Rule — Demo.HL7v3.Rule.RouteAndTransform](image-url)

```xml
<ruleDefinition alias="" context="Demo.HL7v3.Rule.Context">
  <ruleSet name="" effectiveBegin="" effectiveEnd="">
    <constraint name="msgClass" value="Demo.HL7v3.Message"></constraint>
    <constraint name="docName" value="MFMT_IN002101"></constraint>
    <when condition="1">
      <send transform="Demo.HL7v3.Transformation.MFMTIN002101" target="HL7v3 SOAP Out"></send>
      <send transform="Demo.HL7v3.Transformation.MFMTIN002101" target="HL7v3 File Out MFMT"></send>
      <return></return>
    </when>
  </ruleSet>
  <rule name="" disabled="false">
    <constraint name="msgClass" value="Demo.HL7v3.Message"></constraint>
    <constraint name="docName" value="QUPA_IN101103"></constraint>
    <when condition="1">
      <send transform="Demo.HL7v3.Transformation.QUPAIN101103" target="HL7v3 SOAP Out"></send>
      <send transform="Demo.HL7v3.Transformation.QUPAIN101103" target="HL7v3 File Out QUPA"></send>
      <return></return>
    </when>
  </rule>
</ruleDefinition>
```
Viewing the Business Rule Log

The Business Rule Log is a persistent record of business rules that have been executed, their respective results, and reasons for the result. The [Ensemble] > [Business Rule Log] page displays the contents of this log.

You can navigate to the Business Rule Log page from the Management Portal from the Business Rule Log item on the Ensemble View menu. You can also reach this list by clicking Rule Log in the ribbon bar of the [Ensemble] > [Business Process List] page.

The Business Rule Log page is divided into the following three panes where you can perform the indicated functions:

<table>
<thead>
<tr>
<th>Left</th>
<th>Middle</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter search and purge criteria</td>
<td>View the executed rule list</td>
<td>View rule execution details</td>
</tr>
</tbody>
</table>

You can expand and collapse the right and left panes as desired using the double arrow icons.

There are four commands in the ribbon bar of the Rule Log page:

- **Search** — Click to sort and filter the list of rule log entries using the criteria shown in the left pane. See the Search Rules By section for details.
- **Reset** — Click to reset the rule log search criteria to the default values of the quick search fields. See the Quick Search section for details.
- **Previous** — Click to show the previous page of results based on the Page Size.
- **Next** — Click to show the next page of results based on the Page Size.

### 6.1 Enter Search and Purge Criteria

The left pane permits you to enter search and purge criteria to filter the list of rules.

There are two types of search:

- **Quick Search**
- **Search Rules By**

You can also remove entries from the rule log as described in the following section:

- **Purge Rule Log**
6.1.1 Quick Search

Enter the following values to filter the executed rule list:

- **Sort Order** — Select to list either the oldest or the newest entries first. Default is *Newest First*.
- **Page Size** — The maximum number of rule log entries to display in the middle panel as a result of the search. If more entries exist, you can click Previous and Next to page through the results. Default is 500.
- **Auto-refresh** — Select a time interval to refresh the list or select not to have the list automatically refreshed. Default is *None* (no auto-refresh).
- **Page** — This is a read-only field showing what page of the list is being displayed.
- **Time Format** — Select to show the time only or the time with the date. Default is *Complete* (time with date).
- **Errors** — Select this check box if you only want to see rule executions that had errors. Default is to show all executions (check box is cleared).

As you enter values in these fields the middle pane display updates to reflect your entries.

6.1.2 Search Rules By

You can filter what entries display in the list by entering values in one or more of the following fields:

- **Start Time** — Enter the lower limit of a range of Time Executed values.
- **End Time** — Enter the upper limit of a range of Time Executed values.
- **Rule Name** — Choose a rule name as defined in the Ensemble Rule Editor. The filter finds all of the occasions when this rule has been invoked by business processes.
- **Session Id** — Find all the rule log entries associated with a particular session.

Once you enter new search criteria, click Search in the ribbon bar to refresh the list accordingly.

6.1.3 Purge Rule Log

You can purge the rule log by entering the number of days to keep the entries and then clicking Purge. The *Current Count* is a read-only field displaying the number of entries in the rule log. If you want to purge all entries in the log, enter 0 in the *Days* field.

6.2 View the Executed Rule List

Each time a business process executes or “fires” a rule, Ensemble writes an entry to the Business Rule Log stating the details of what happened. You can view this log on the [Ensemble] > [Business Rule Log] page of the Management Portal. The list displays the following information for each business rule log entry:

- **Session** — The unique identifier for the session that is (or was) associated with this rule. A session marks the beginning and end of all the activities prompted by a primary request message from outside Ensemble.
- **Time Executed** — The date and time when this rule was last invoked.
- **Rule Name** — The name assigned to the rule in the Ensemble Rule Editor.
• **Error** — An integer value of 1 means that **Error** is true; the rule encountered an error. A value of 0 means **Is Error** is false; the rule did not encounter any errors.

You can also perform the following actions on a selected rule:

• Click the **Session** to navigate to the **Visual Trace** display for the session that contained this particular execution of the business rule.

• Click the **Rule Name** to navigate to the **Ensemble Rule Editor** page for this business rule definition class.

### 6.3 View Rule Execution Details

You can select a log entry to view the details of that particular execution of the rule. Select a row in the middle pane and the expanded right pane displays the following informational fields:

**Execution ID**

Unique identifier for this rule execution.

**Session ID**

Unique identifier for the session that is (or was) associated with this execution of this rule. A session marks the beginning and end of all the activities prompted by a primary request message from outside Ensemble.

**Time Executed**

Date and time when this rule was executed.

**Rule Name**

Name of the rule definition class that was executed.

**Rule Set**

Name of the rule set that was executed.

**Reason**

Specific rule name that cause the rules engine to generate the result. If a business rule is empty or undefined, the reason is **Rule Missing**.

**Error?**

Displays **Yes** or **No** depending on whether the execution of the rule resulted in an error.

**Error Message**

If the rules engine returns an error, then this is the text associated with the error.

**Return Value**

Value returned by the rules engine for this rule.

**Activity Name**

Name assigned to the `<rule>` activity in the BPL code.
Effective Begin Date/Time

Effective begin date and time of the executed rule set.

Effective End Date/Time

Effective end date and time of the executed rule set.

These informational values correspond with the properties of the Ens.Rule.Log class, which you can view in the Class Reference.

From the right pane, you can perform the following actions on the selected rule:

- Click Trace to navigate to the Visual Trace display for the session that contained this particular execution of the business rule.
- Click Rule to navigate to the Ensemble Rule Editor page for this business rule definition class.
Index

A
- actions, 12
- adding
  - business rules, 7

B
- BPL
  - and business rules, 7, 23
- Business Processes page
  - Rule Log, 35
- Business Rule Editor page, 7
- Business Rule Log page, 35
  - Filter Rule Log, 35
  - purge criteria, 35
  - search criteria, 35
- business rules
  - about, 3
  - actions, 12
  - adding, 7
  - and BPL, 7, 23
  - and classes, 5
  - conditions, 16
  - expression examples, 20
  - functions, 20
  - history of rule activity, 35
  - importing and exporting, 7
  - list of rule definitions, 7
  - log entries, 35, 36
  - monitoring, 35
  - otherwise clauses, 12
  - routing rule sets, 11
  - rules, in rule sets, 12
  - rule sets, 11
  - types of, 4
  - values, in a rule definition, 18
  - when clauses, 12
- Business Rules page, 7
  - Log, 35

C
- classes
  - and business rules, 5
  - configuring
    - business rules, 7

E
- Ensemble Rule Editor page
  - Rule List, 7
  - Rule Log, 35
- errors
  - Is Error status, 37
  - expression examples, 20

F
- filtering lists
  - business rule log, 35
  - functions, 20

I
- Is Error status, 37

L
- log, business rule
  - see business rules

M
- monitoring
  - business rules, 35

O
- otherwise clauses, 12

R
- routing rule sets, 4
- rule, business
  - see business rule
  - rule sets, 4, 11

S
- sets, rule, 11
- Show Business Rule Log, 35

V
- values, 18
  - see also business rules; status
  - viewing
    - business rule log, 35
    - list of rule definitions, 7

W
- when clauses, 12