

Corticon Studio:

Rule Language Guide

Notices

Copyright agreement

© 2016 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

Business Making Progress, Corticon, DataDirect (and design), DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, Deliver More Than Expected, Icenium, Kendo UI, Making Software Work Together, NativeScript, OpenEdge, Powered by Progress, Progress, Progress Software Business Making Progress, Progress Software Developers Network, Rollbase, RulesCloud, RulesWorld, SequeLink, Sitefinity (and Design), SpeedScript, Stylus Studio, TeamPulse, Telerik, Telerik (and Design), Test Studio, and WebSpeed are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries. AccelEvent, AppsAlive, AppServer, BravePoint, BusinessEdge, DataDirect Spy, DataDirect SupportLink, Future Proof, High Performance Integration, OpenAccess, ProDataSet, Progress Arcade, Progress Profiles, Progress Results, Progress RFID, Progress Software, ProVision, PSE Pro, SectorAlliance, Sitefinity, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, Smart DataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, WebClient, and Who Makes Progress are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

Please refer to the Release Notes applicable to the particular Progress product release for any third-party acknowledgements required to be provided in the documentation associated with the Progress product.

Table of Contents

Chapter 1: Introduction to Corticon Rule Language.....	9
Rule structure.....	10
Basic data types.....	10
Truth values.....	11
Collection operators.....	11
Language operators.....	11
Vocabulary used in this Language Guide.....	12
 Chapter 2: Rule operators.....	 13
Icons.....	13
Tool tips.....	15
Usage restrictions.....	15
 Chapter 3: Categories of rule operators.....	 17
Organization.....	18
General terms.....	18
Attribute operators.....	19
Boolean.....	19
DateTime.....	20
Date.....	24
Time.....	28
Decimal.....	31
Integer.....	34
String.....	37
Entity/Association operators.....	41
Entity.....	41
Collection.....	42
Sequence.....	44
Extended Operators.....	46
 Chapter 4: Rule operator details and examples.....	 49
Absolute value.....	53
Add numbers.....	54
Add strings.....	55
Add days.....	56
Add hours.....	57
Add minutes.....	58

Contents

Add months.....	59
Add seconds.....	60
Add years.....	61
Associate element(s).....	62
At.....	63
Average.....	64
CellValue.....	65
Clone.....	68
Concatenate.....	71
Contains.....	72
Day.....	73
Days between.....	74
Day of week.....	75
Day of year.....	77
Decrement.....	78
Disassociate element(s).....	79
Divide.....	80
Div.....	81
Ends with.....	82
Equals when used as an assignment.....	83
Equals when used as a comparison.....	84
Equals ignoring case.....	85
Equals when using Strings.....	86
Exists.....	87
Exponent.....	89
False.....	90
First.....	91
First (number).....	92
Floor.....	94
For all.....	95
Greater than.....	97
Greater than or equal to.....	98
Hour.....	99
Hour between.....	100
In {List}.....	101
In (Range).....	104
Increment.....	106
Index of.....	107
Is empty.....	108
Iterate.....	109
Last.....	110
Last (number).....	112
Less than.....	113
Less than or equal to.....	115
Logarithm (Base 10).....	116

Logarithm (Base x).....	117
Lowercase.....	119
Maximum value.....	120
Maximum value (Collection).....	121
Minimum value.....	122
Minimum value (Collection).....	123
Minute.....	124
Minutes between.....	125
Mod.....	126
Month.....	127
Months between.....	128
Multiply.....	129
Natural logarithm.....	130
New.....	132
New unique.....	133
Not.....	135
Not empty.....	136
Not equal to.....	137
Now.....	139
Null.....	140
Other.....	141
Or.....	142
Remove element.....	143
Replace element(s).....	145
Round.....	147
Second.....	148
Seconds between.....	149
Size of string.....	150
Size of collection.....	151
Sorted by.....	152
Sorted by descending.....	154
Starts with.....	156
SubSequence.....	157
Substring.....	159
Subtract.....	160
Sum.....	161
Today.....	162
To date: Casting a dateTime to a date.....	163
To dateTime: Casting a string to a dateTime.....	164
To dateTime: Casting a date to a dateTime.....	165
To dateTime: Casting a time to a dateTime.....	166
To dateTime: Timezone offset.....	167
To decimal.....	168
To integer.....	169
To string.....	170

To time: Casting a dateTime to a time.....	171
Trend.....	172
True.....	174
Uppercase.....	175
Week of month.....	176
Week of year.....	177
Year.....	178
Years between.....	179

Appendix A: Character precedence: Unicode and Java Collator.....181

Appendix B: Precedence of rule operators.....185

Appendix C: DateTime data type.....189

Appendix D: Access to Corticon knowledge resources.....191

Introduction to Corticon Rule Language

Graphical modeling languages and tools (UML, ER, ORM, for example) are not sufficiently precise for specifications. Additional constraints on the objects in the model must also be defined. While natural languages are easily used by individuals without a programming background, they are often ambiguous. On the other hand, formal programming languages are precise, but not easily used by business analysts and other non-programmers.

The Corticon Rule Language has been developed to resolve this dilemma. Based on the Object Constraint Language (OCL, an extension of the Universal Modeling Language specification 1.1), the *Corticon Rule Language* (CRL) is designed to enable non-programmers to express rules clearly and precisely without the use of procedural programming languages. More information on OCL may be found at www.uml.org.

For details, see the following topics:

- [Rule structure](#)
- [Basic data types](#)
- [Truth values](#)
- [Collection operators](#)
- [Language operators](#)
- [Vocabulary used in this Language Guide](#)

Rule structure

In traditional programming languages (or logic systems), most rules are expressed via IF/THEN structures. The IF clause contains a conditional expression and the THEN clause contains actions the rule should perform if all conditions have been met. This IF/THEN structure is expressed as Conditions and Actions in the Rulesheet user interface of Corticon Studio. For more information on building and organizing rules in Corticon Studio, see the *Corticon Studio Tutorial: Basic Rule Modeling*.

Basic data types

The proper expression and execution of rules in Corticon Studio is dependent on the type of data involved. Each attribute in the Corticon Studio Business Vocabulary has a data type, meaning that it has restrictions on the type of data it may contain. Corticon standard data types are listed and described in the following table:

Data Type	Description
String	Any combination of alphanumeric characters, of any length
Integer	A whole number, including zero and negative numbers, of any length
Decimal	A number containing a decimal point, including zero and negative numbers, of any length
Boolean	Values are <code>true</code> and <code>false</code> . <code>T</code> and <code>F</code> may also be used.
DateTime	Values must be entered for both date and time.
Date	A value with only date information. No Time information is allowed.
Time	Value with only time information. No Date information is allowed.

In this guide, the data types Integer and Decimal are often referred to by the generic term `<Number>`. Wherever `<Number>` is used, either Integer or Decimal data types may be used.

Syntax such as `<DateTime>` indicates that data must conform to the data type shown in angle brackets (`< . . >`). For this example, you might enter `9/13/2013 2:00:00 PM EST`. Do not type the angle brackets themselves.

See [DateTime data type](#) on page 189 for further details on formatting DateTime, Date, and Time information.

Truth values

This guide uses the notation <Expression> to refer to some combination of terms from the Vocabulary that resolves or evaluates to a single “truth value”. A truth value is the Boolean value (true or false) assigned to an expression upon evaluation by the rule engine. For example, the expression `Patient.name='John'` has a truth value of true whenever the patient's name is John. If it is not John, then the truth value of this expression is false.

Collection operators

Many of the operators provided in the Corticon Rule Language deal exclusively with collections of entities. When using collection operators, the expression **must** use aliases to represent the collection(s) operated on by the collection operator(s). A complete discussion of aliases is included in the *Rule Modeling Guide*. Reminders are included throughout this manual wherever collection operators are referenced.

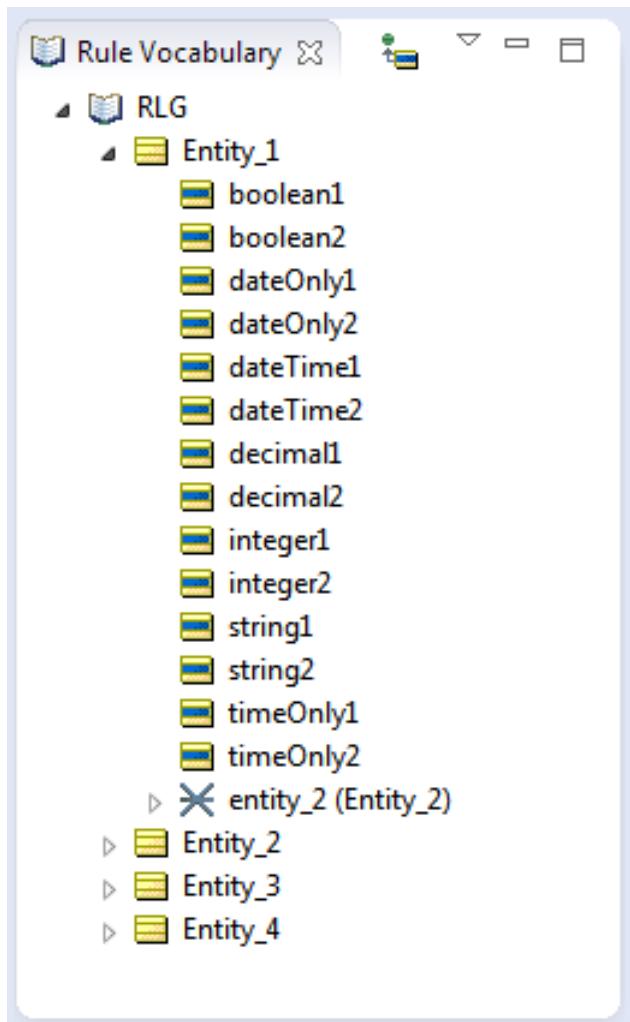
Language operators

The Corticon Rule Language operators can be grouped into various classifications as shown in the following tables. Each operator is subsequently described in detail in the [Language Reference](#) section of this document. This section includes a detailed description of the operator, its syntax, usage restrictions, and an example in a Corticon Rulesheet.

Vocabulary used in this Language Guide

This guide uses a generic Vocabulary in all its examples. The Vocabulary contains four entities, each of which contains the same attribute names and types. Attribute names reflect their data types. For example, `integer1` has a data type of Integer. This generic Vocabulary provides sufficient flexibility to create examples using all operators and functions in the Corticon Rule Language. `Entity1` is shown expanded in the following figure:

Figure 1: Vocabulary used in Corticon Language Guide examples



2

Rule operators

This chapter describes the toolset for accessing and using operators.

For details, see the following topics:

- [Icons](#)
- [Tool tips](#)
- [Usage restrictions](#)

Icons

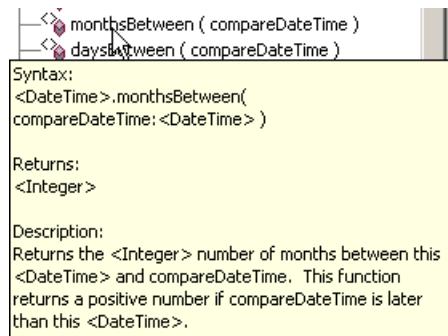
Rule Operators are assigned icons which provide the user with information about their usage. The following table describes these icons:

Icon	Where Found	Purpose	Examples
	General, Literals category	indicates special values or constants	null , true , other
	General, Functions category	indicates system values that are automatically retrieved upon rule execution.	now , today
	Operators, Boolean category	this special “unary” operator icon is used only with not	not
	Operators, all categories	indicates the operator uses a period “.” to attach to its operand. Most operators with this icon typically fell into the previous “function” category.	day , round , contains
	Operators, all categories	indicates the operator is used between two operands. Most operators with this icon typically fell into the previous “comparison” category.	equals , multiply
	Operators, Collection & Sequence categories	indicates the operator is used with collections or sequences. Also indicates an alias must be used to represent the collection operated on.	sum , size
	Extended Operators	indicates the operator has been added to the Vocabulary using the extension framework described in <i>Corticon Extensions Guide</i> .	-

Tool tips

In Corticon Studio, moving the mouse over a Vocabulary operator and pausing, or “hovering” for a moment, will cause a dynamic “tool tip” text box to display. This tool tip contains information about operator syntax, return data type, and description, all of which are supplied in more detail in this Guide. For questions not answered by the tool tip, refer to the detailed operator descriptions in this Guide. The following figure shows a typical tool tip for the date operator `.monthsBetween`:

Figure 2: Typical Rule Operator Tool Tip



Usage restrictions

The following illustrations show the general usage restrictions for the various types of Vocabulary terms depending on where they are used in a Rulesheet. This table indicates, for example, that entities (terms from the Vocabulary) may be used in any section of the Rulesheet. Rule Operators, however, are restricted to only three sections.

Note: Some operators have specific restrictions that vary from this general table – see each operator's usage restrictions for details of these exceptions.

Figure 3: Vocabulary usage restrictions in Rulesheet sections

Rulesheet Section Name	Scope	Filter Rows	Condition Rows	Condition Cells	Actions Rows	Action Cells	Rule Statements
Rulesheet Section #	1	2	3	4	5	6	7
Literals		<input checked="" type="checkbox"/>					
Functions		<input checked="" type="checkbox"/>					
Operators		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>		
Data	Values	<input checked="" type="checkbox"/>					
	Terms	<input checked="" type="checkbox"/>					

Figure 4: Sections of Rulesheet that correlate with usage restrictions

The screenshot shows the 'SectionsOfRulesheet.ers' editor interface. The top pane is titled 'Scope' and contains a list of conditions (a, b, c, d, e) and actions (Post Message(s), A, B, C). The bottom pane is titled 'Rule Statements' and contains a table with columns: Ref, ID, Post, Alias, and Text. The number 7 is highlighted in an orange circle at the bottom right corner of the 'Text' column.

Ref	ID	Post	Alias	Text
				7

3

Categories of rule operators

Corticon Studio presents its rule operators in logical groups.

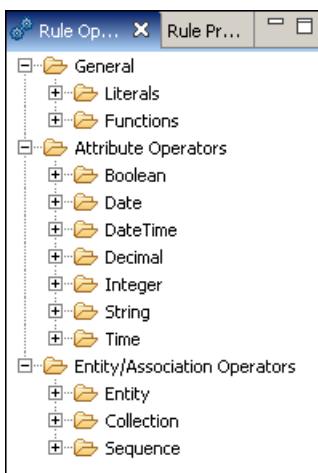
For details, see the following topics:

- [Organization](#)
- [General terms](#)
- [Attribute operators](#)
- [Entity/Association operators](#)
- [Extended Operators](#)

Organization

Rule Operators are classified based on the data type(s) of the terms *to which the operator may be applied* (known as the “operand”), as shown:

Figure 5: Rule Operator categories



General terms

Corticon's **General** operators are categorized as **Literals** and **Functions**.

Literals

Literal Terms can be used in any section of the Rulesheet, except **Scope** and **Rule Statements**. Exceptions to this general statement exist – see individual literals for detailed usage restrictions.

Corticon's **Literals** operators are as follows:

Name and Syntax	Returns	Description
Null		
null	none	<p>The null value corresponds to one of three different scenarios:</p> <ul style="list-style-type: none"> the absence of an attribute in a Ruletest scenario the absence of data for an attribute in a Ruletest scenario an object that has a value of null
True		
true or T	Boolean	Represents Boolean value true

Name and Syntax	Returns	Description
False		
false or F	Boolean	Represents the Boolean value false
Other		
other	any	When included in a condition's Values set, other represents any value not explicitly included in the set, including null .
CellValue		
cellValue	any	cellValue is a variable whose value is determined by the rule Column that executes

Functions

Corticon's Functions operators are as follows:

Name and Syntax	Returns	Description
Now		
now	Date	Returns the current system date and time when the rule is executed.
Today		
today	Date	Returns the current system date when the rule is executed.

Attribute operators

The Corticon Rule Language supports attribute operators categorized as Boolean, DateTime, Date, Time, Decimal, Integer, and String.

Boolean

Corticon's **Boolean** attribute operators are as follows:

Name and Syntax	Returns	Description
Equals (used as a comparison)		
<Expression1> = <Expression2>	Boolean	Returns a value of true if <Expression1> has the same value as <Expression2>.

Name and Syntax	Returns	Description
Equals (used as an assignment)		
<Boolean1> = <Expression1>	Boolean	Assigns the truth value of <Expression1> to <Boolean1>
Not Equal To		
<Expression1> <> <Expression2>	Boolean	Returns a value of true if <Expression1> does not have the same truth value as <Expression2>
Or		
<Expression1> or <Expression2> or ...	Boolean	Returns a value of true if either <Expression1> or <Expression2> evaluates to true.
And		
<<Boolean1> and <Boolean2>	Boolean	Returns a value of true if both <<Boolean1> and <Boolean2> are true. NOTE: This operator can be used only in Preconditions/Filters section of the Rulesheet.
Not		
not <Expression>	Boolean	Returns the negation of the truth value of <Expression>

Date Time

Note: A DateTime data type **must contain both** date information **and** time information. Applying a DateTime operator to a DateTime attribute should always produce a result. Be sure to use the data type that suits your needs.

Corticon's **DateTime** attribute operators are as follows:

Name and Syntax	Returns	Description
Equals (used as a comparison)		
<DateTime1> = <DateTime2>	Boolean	Returns a value of true if <DateTime1> is the same as <DateTime2>, including both the Date and the Time portions
Equals (used as an assignment)		
<DateTime1> = <DateTime2>	DateTime	Assigns the value of <DateTime2> to <DateTime1>

Name and Syntax	Returns	Description
Not Equal To		
<DateTime1> <> <DateTime2>	Boolean	Returns a value of true if <DateTime1> does not equal <DateTime2>
Less than		
<DateTime1> < <DateTime2>	Boolean	Returns a value of true if <Date1> is less than <Date2>
Greater than		
<DateTime1> > <DateTime2>	Boolean	Returns a value of true if <DateTime1> is greater than or equal to <DateTime2>
Less than or Equal to		
<DateTime1> <= <DateTime2>	Boolean	Returns a value of true if <DateTime1> is less than or equal to <DateTime2>
Greater than or Equal to		
<DateTime1> >= <DateTime2>	Boolean	Returns a value of true if <DateTime1> is greater than or equal to <DateTime2>
In (Range)		
attributeReference in [(rangeExpression)]	Boolean	Returns a value of true if attributeReference is in the range of DateTime values <i>from..to</i> , and where opening and closing parentheses () indicate exclusion of that limit and square brackets [] indicate inclusion of that limit.
In (List)		
attributeReference in {listExpression}	Boolean	Returns a value of true if attributeReference is in the comma-delimited list of literal values, defined enumeration values, or - if in use - enumeration labels.
Year		
<DateTime>.year	Integer	Returns the century/year portion of <DateTime> as a four digit Integer
Month		
<DateTime>.month	Integer	Returns the month in <DateTime> as an Integer between 1 and 12
Day		

Name and Syntax	Returns	Description
<DateTime>.day	Integer	Returns the day portion of <DateTime> as an Integer between 1 and 31
Hour		
<DateTime>.hour	Integer	Returns the hour portion of <DateTime>. The returned value is based on a 24-hour clock.
Minute		
<DateTime>.min	Integer	Returns the minute portion of <DateTime> as an Integer between 0 and 59
Second		
<DateTime>.sec	Integer	Returns the seconds portion of <DateTime> as an Integer between 0 and 59
Add years		
<DateTime>.addYears (<Integer>)	Date	Adds the number of years in <Integer> to the number of years in <DateTime>
Add months		
<DateTime>.addMonths (<Integer>)	Date	Adds the number of months in <Integer> to the number of months in <DateTime>
Add days		
<DateTime>.addDays (<Integer>)	Date	Adds the number of days in <Integer> to the number of days in <DateTime>
Add hours		
<DateTime>.addHours (<Integer>)	Date	Adds the number of hours in <Integer> to the number of hours in the Time portion of <DateTime>
Add minutes		
<DateTime>.addMinutes (<Integer>)	Date	Adds the number of minutes in <Integer> to the number of minutes in the Time portion of <DateTime>
Add seconds		
<DateTime>.addSeconds (<Integer>)	Date	Adds the number of seconds in <Integer> to the number of seconds in the Time portion of <DateTime>

Name and Syntax	Returns	Description
Years between		
<DateTime1>.yearsBetween(<DateTime2>)	Integer	Returns the Integer number of years between <DateTime1> and <DateTime2>. This function returns a positive number if <DateTime2> is later than <DateTime1>.
Months between		
<DateTime1>.monthsBetween(<DateTime2>)	Integer	Returns the Integer number of months between <DateTime1> and <DateTime2>. If the month and year portions of <DateTime1> and <DateTime2> are the same, the result is zero. This function returns a positive number if <DateTime2> is later than <DateTime1>.
Days between		
<DateTime1>.daysBetween(<DateTime2>)	Integer	Returns the Integer number of days between <DateTime1> and <DateTime2>. If the two dates differ by less than a full 24-hour period, the value is zero. This function returns a positive number if <DateTime2> is later than <DateTime1>.
Hours between		
<DateTime1>.hoursBetween(<DateTime2>)	Integer	Returns the Integer number of hours between <DateTime1> and <DateTime2>. If the two dates differ by less than a full hour, the value is zero. This function returns a positive number if <DateTime2> is later than <DateTime1>.
Minutes between		
<DateTime1>.minsBetween(<DateTime2>)	Integer	Returns the Integer number of minutes between <DateTime1> and <DateTime2>. This function returns a positive number if <DateTime2> is later than <DateTime1>.
Seconds between		
<DateTime1>.secsBetween(<DateTime2>)	Integer	Returns the Integer number of seconds between <DateTime1> and <DateTime2>. This function returns a positive number if <DateTime2> is later than <DateTime1>.
Day of Week		
<DateTime>.dayOfWeek	Integer	Returns an Integer corresponding to day of the week, with Sunday equal to 1, in <DateTime>.

Name and Syntax	Returns	Description
Week of Year		
<DateTime>.weekOfYear	Integer	Returns an Integer from 1 to 52, equal to the week number within the year in <DateTime>
Day of Year		
<DateTime>.dayOfYear	Integer	Returns an Integer from 1 to 366, equal to the day number within the year in <DateTime>
Week of Month		
<DateTime>.weekOfMonth	Integer	Returns an Integer from 1 to 6, equal to the week number within the month in <DateTime> or <Date>. A week begins on Sunday and ends on Saturday.
To Date		
<DateTime>.toDate	Date	Returns the date portion only of DateTime
To Time		
<DateTime>.toTime	Time	Returns the time portion only of DateTime
To String		
<DateTime>.toString	String	Converts DateTime to a String with date and time information
getMilliseconds		
<DateTime>.getMilliseconds	Integer	Returns the internal date/time, namely the number of milliseconds that have transpired since the epoch 1/1/1970 00:00:00 GMT.
toString		
<DateTime>.toString	Integer	Converts the value of <DateTime> to data type <String>.
toZulu		
<DateTime>.toZulu	String	Returns an ISO-8601-compliant date-time as a String.

Date

Corticon's **Date** attribute operators are as follows:

Name and Syntax	Returns	Description
Equals (used as a comparison)		
<Date1> = <Date2>	Boolean	Returns a value of true if <Date1> is the same as <Date2>.
Equals (used as an assignment)		
<Date1> = <Date2>	DateTime	Assigns the value of <Date2> to <Date1>
Not Equal To		
<Date1> <> <Date2>	Boolean	Returns a value of true if <Date1> does not equal <Date2>
Less than		
<Date1> < <Date2>	Boolean	Returns a value of true if <Date1> is less than <Date2>
Greater than		
<Date1> > <Date2>	Boolean	Returns a value of true if <Date1> is greater than or equal to <Date2>
Less than or Equal to		
<Date1> <= <Date2>	Boolean	Returns a value of true if <Date1> is less than or equal to <Date2>
Greater than or Equal to		
<Date1> >= <Date2>	Boolean	Returns a value of true if <Date1> is greater than or equal to <Date2>
In (Range)		
attributeReference in [(rangeExpression)]	Boolean	Returns a value of true if attributeReference is in the range of Date values <i>from..to</i> , and where opening and closing parentheses () indicate exclusion of that limit and square brackets [] indicate inclusion of that limit.
In (List)		
attributeReference in {listExpression}	Boolean	Returns a value of true if attributeReference is in the comma-delimited list of literal values, defined enumeration values, or - if in use - enumeration labels.
Year		

Name and Syntax	Returns	Description
<Date>.year	Integer	Returns the century/year portion of <Date> as a four digit Integer
Month		
<Date>.month	Integer	Returns the month in <Date> as an Integer between 1 and 12
Day		
<Date>.day	Integer	Returns the day portion of <Date> as an Integer between 1 and 31
Add years		
<Date>.addYears(<Integer>)	Date	Adds the number of years in <Integer> to the number of years in <Date>
Add months		
<Date>.addMonth(<Integer>)	Date	Adds the number of months in <Integer> to the number of months in <DateTime>
Add days		
<Date>.addDays(<Integer>)	Date	Adds the number of days in <Integer> to the number of days in <Date>
Years between		
<Date1>.yearsBetween(<Date2>)	Integer	Returns the Integer number of years between <Date1> and <Date2>. This function returns a positive number if <Date2> is later than <Date1>.
Months between		
<Date1>.monthsBetween(<Date2>)	Integer	Returns the Integer number of months between <Date1> and <Date2>. If the month and year portions of <Date1> and <Date2> are the same, the result is zero. This function returns a positive number if <Date2> is later than <Date1>.
Days between		
<Date1>.daysBetween(<Date2>)	Integer	Returns the Integer number of days between <Date1> and <Date2>. If the two dates differ by less than a full 24-hour period, the value is zero. This function returns a positive number if <Date2> is later than <Date1>.
Day of Week		

Name and Syntax	Returns	Description
<Date>.dayOfWeek	Integer	Returns an Integer corresponding to day of the week, with Sunday equal to 1, in <Date>.
Week of Year		
<Date>.weekOfYear	Integer	Returns an Integer from 1 to 52, equal to the week number within the year in <Date>
Day of Year		
<Date>.dayOfYear	Integer	Returns an Integer from 1 to 366, equal to the day number within the year in <Date>
Week of Month		
<Date>.weekOfMonth	Integer	Returns an Integer from 1 to 6, equal to the week number within the month in <DateTime> or <Date>. A week begins on Sunday and ends on Saturday.
To String		
<Date>.toString	String	Converts DateTime to a String with date and time information
To DateTime		
<Date>.toDateTime	DateTime	Returns a DateTime where the date portion is equal to the value of <Date> and the time portion is equal to 00:00:00 in the system's local timezone
To DateTime with Timezone Offset		
<Date>.toDateTime (<string>)	DateTime	Returns a DateTime where the date portion is equal to the value of <Date> and the time portion is equal to 00:00:00 in the timezone specified by the value of <string>
getMilliseconds		
toString		
<Date>.toString	Integer	Converts the value of <Date> to data type <String>.
<Date>.getMilliseconds	Integer	Returns the internal date/time, namely the number of milliseconds that have transpired since the epoch 1/1/1970 00:00:00 GMT.

Name and Syntax	Returns	Description
nextDay		
<Date>.nextDay	Date	Returns the Date that represents the date that follows this Date instance.

Time

Corticon's **Time** attribute operators are as follows:

Name and Syntax	Returns	Description
Equals (used as a comparison)		
<Time1> = <Time2>	Boolean	Returns a value of true if <Time1> is the same as <Time2>, including both the Date and the Time portions
Equals (used as an assignment)		
<Time1> = <Time2>	DateTime	Assigns the value of <Time2> to <Time1>
Not Equal To		
<Time1> <> <Time2>	Boolean	Returns a value of true if <Time1> does not equal <Time2>
Less than		
<Time1> < <Time2>	Boolean	Returns a value of true if <Time1> is less than <Time2>
Greater than		
<Time1> > <Time2>	Boolean	Returns a value of true if <Time1> is greater than <Time2>
Less than or Equal to		
<Time1> <= <Time2>	Boolean	Returns a value of true if <Time1> is less than or equal to <Time2>
Greater than or Equal to		
<Time1> >= <Time2>	Boolean	Returns a value of true if <Time1> is greater than or equal to <Time2>
In (Range)		

Name and Syntax	Returns	Description
attributeReference in [(rangeExpression)]	Boolean	Returns a value of true if attributeReference is in the range of Time values <i>from..to</i> , and where opening and closing parentheses () indicate exclusion of that limit and square brackets [] indicate inclusion of that limit.
In (List)		
attributeReference in {listExpression}	Boolean	Returns a value of true if attributeReference is in the comma-delimited list of literal values, defined enumeration values, or - if in use - enumeration labels.
Hour		
<Time>.hour	Integer	Returns the hour portion of <Time>. The returned value is based on a 24-hour clock.
Minute		
<Time>.min	Integer	Returns the minute portion of <Time> as an Integer between 0 and 59
Second		
<Time>.sec	Integer	Returns the seconds portion of <Time> as an Integer between 0 and 59
<Time>.addDays (<Integer>)	Date	Adds the number of days in <Integer> to the number of days in <Time>
Add hours		
<Time>.addHours (<Integer>)	Date	Adds the number of hours in <Integer> to the number of hours in the Time portion of <Time>
Add minutes		
<Time>.addMinutes (<Integer>)	Date	Adds the number of minutes in <Integer> to the number of minutes in the Time portion of <Time>
Add seconds		
<Time>.addSeconds (<Integer>)	Date	Adds the number of seconds in <Integer> to the number of seconds in the Time portion of <Time>
Hours between		

Name and Syntax	Returns	Description
<Time1>.hoursBetween (<Time2>)	Integer	Returns the Integer number of hours between <Time1> and <Time2>. If the two times differ by less than a full hour, the value is zero. This function returns a positive number if <Time2> is later than <Time1>.
Minutes between		
<Time1>.minsBetween (<Time2>)	Integer	Returns the Integer number of minutes between <Time1> and <Time2>. This function returns a positive number if <Time2> is later than <Time1>.
Seconds between		
<Time1>.secsBetween (<Time2>)	Integer	Returns the Integer number of seconds between <Time1> and <Time2>. This function returns a positive number if <Time2> is later than <Time1>.
To String		
<Time>.toString	String	Converts <Time> to a String with date and time information
To DateTime		
<Time>.toDateTime	DateTime	Returns a DateTime where the time portion is equal to the value of <Time> and the date portion is equal to the epoch.
toString		
<Time>.toString	Integer	Converts the value of <Time> to data type <String>.
getMilliseconds		
<Time>.getMilliseconds	Integer	Returns the internal date/time, namely the number of milliseconds that have transpired since the epoch 1/1/1970 00:00:00 GMT.
toString		
<Time>.toString	Integer	Converts the value of <Time> to data type <String>.
getTimeName		
<Time>.getTimeName	String	Returns a String that states whether the time is morning, afternoon, or evening.

Decimal

In this section, wherever the syntax includes <Number>, either Integer or Decimal data types may be used.

Corticon's **Decimal** attribute operators are as follows:

Name and Syntax	Returns	Description
Equals (used as a comparison)		
<Number1> = <Number2>	Boolean	Returns a value of true if <Number1> is the same as <Number2>.
Equals (used as an assignment)		
<Number1> = <Number2>	Number	Assigns the value of <Number2> to the value of <Number1>.
Not Equal To		
<Number1> <> <Number2>	Boolean	Returns a value of true if <Number1> is not equal to <Number2>.
Less than		
<Number1> < <Number2>	Boolean	Returns a value of true if <Number1> is less than <Number2>.
Greater than		
<Number1> > <Number2>	Boolean	Returns a value of true if <Number1> is greater than <Number2>.
Less than or Equal to		
<Number1> <= <Number2>	Boolean	Returns a value of true if <Number1> is less than or equal to <Number2>.
Greater than or Equal to		
<Number1> >= <Number2>	Boolean	Returns a value of true if <Number1> is greater than or equal to <Number2>.
In (Range)		
attributeReference in [(rangeExpression)]	Boolean	Returns a value of true if attributeReference is in the range of Decimal values <i>from..to</i> , and where opening and closing parentheses () indicate exclusion of that limit and square brackets [] indicate inclusion of that limit.
In (List)		

Name and Syntax	Returns	Description
attributeReference in {listExpression}	Boolean	Returns a value of true if attributeReference is in the comma-delimited list of literal values, defined enumeration values, or - if in use - enumeration labels.
Add		
<Number1> + <Number2>	Number	Returns the sum of <Number1> and <Number2>. The resulting data type is the more expansive of either <Number1> or <Number2>. For example, if an Integer value is added to a Decimal value, the resulting value will be a Decimal. See Precedence of rule operators on page 185.
Subtract		
<Number1> - <Number2>	Number	Subtracts <Number2> from <Number1>. The resulting data type is the more expansive of either <Number1> or <Number2>. See Precedence of rule operators on page 185.
Multiply		
<Number1> * <Number2>	Number	Returns the product of <Number1> and <Number2>. The resulting data type is the more expansive of either <Number1> or <Number2>. See Precedence of rule operators on page 185.
Divide		
<Number1> / <Number2>	Number	Divides <Number1> by <Number2>. The resulting data type is the more expansive of either <Number1> or <Number2>. See Precedence of rule operators on page 185.
Exponent		
<Number1> ** <Number2>	Number	Raises <Number1> to the power of <Number2>. The resulting data type is the more expansive of either <Number1> or <Number2>. See Precedence of rule operators on page 185.
Increment		
<Number1> += <Number2>	Number	Increments <Number1> by <Number2>. The data type of <Number1> must accommodate the addition of <Number2>. See Precedence of rule operators on page 185.
Decrement		

Name and Syntax	Returns	Description
<Number1> -= <Number2>	Number	Decrements <Number1> by the value of <Number2>. The data type of <Number1> must accommodate the addition of <Number2>. See Precedence of rule operators on page 185.
Absolute Value		
<Decimal>.absVal	Decimal	Returns the absolute value of <Number>. If the <Number> is positive, <Number> itself is returned; if <Number> is negative, the negation of <Number> is returned.
Floor		
<Decimal>.floor	Integer	Returns the largest (closest to positive infinity) Integer that is not greater than <Number>.
Round		
<Decimal>.round	Decimal	Rounds <Decimal> to the nearest Integer.
Round(n)		
<Decimal>.round(<Integer>)	Decimal	Rounds <Decimal> to the number of decimal places specified by <Integer>.
To Integer		
<Decimal>.toInteger	Integer	Converts an attribute of type Decimal to type Integer. Decimals will have the decimal point and fraction (those digits to the right of the decimal point) truncated.
To String		
<Decimal>.toString	String	Converts an attribute of type Decimal to type string
Maximum Value		
<Decimal>.max(<Number>)	Number	Returns the greater of <Decimal> and <Number>.
Minimum Value		
<Decimal>.min(<Number>)	Number	Returns the lesser of <Decimal> and <Number>.
Logarithm (base 10)		
<Decimal>.log	Decimal	Returns the logarithm (base 10) of <Decimal>. <Decimal> may not be zero.
Logarithm (base x)		

Name and Syntax	Returns	Description
<Decimal1>.log(<Decimal2>)	Decimal	Returns the logarithm (base <Decimal2>) of <Decimal1>. <Decimal1> may not be zero.
Natural Logarithm		
<Decimal>.ln	Decimal	Returns the logarithm (base e) of <Decimal>. <Decimal> may not be zero.
truncate		
<Decimal>.truncate	Integer	Truncates "this" Decimal value to an integer by removing the fractional portion.
toString		
<Decimal>.fraction	Decimal	Extracts the fraction portion of "this" Decimal.
movePoint(places)		
<Decimal>.movePoint (places: Integer)	Decimal	Moves the Decimal value's point moved n places where n can be a positive (moves right) or negative (moves left) value.

Integer

In this section, wherever the syntax includes <Number>, either Integer or Decimal data types may be used.

Corticon's **Integer** attribute operators are as follows:

Name and Syntax	Returns	Description
Equals (used as a comparison)		
<Number1> = <Number2>	Boolean	Returns a value of true if <Number1> is the same as <Number2>.
Equals (used as an assignment)		
<Number1> = <Number2>	Number	Assigns the value of <Number2> to the value of <Number1>. The data type of <Number1> must be expansive enough to accommodate <Number2>.
Not Equal To		
<Number1> <> <Number2>	Boolean	Returns a value of true if <Number1> is not equal to <Number2>.
Less than		

Name and Syntax	Returns	Description
<Number1> < <Number2>	Boolean	Returns a value of true if <Number1> is less than <Number2>.
Greater than		
<Number1> > <Number2>	Boolean	Returns a value of true if <Number1> is greater than <Number2>.
Less than or Equal to		
<Number1> <= <Number2>	Boolean	Returns a value of true if <Number1> is less than or equal to <Number2>.
Greater than or Equal to		
<Number1> >= <Number2>	Boolean	Returns a value of true if <Number1> is greater than or equal to <Number2>.
In (Range)		
attributeReference in [(rangeExpression)]	Boolean	Returns a value of true if attributeReference is in the range of Integer values <i>from..to</i> , and where opening and closing parentheses () indicate exclusion of that limit and square brackets [] indicate inclusion of that limit.
In (List)		
attributeReference in {listExpression}	Boolean	Returns a value of true if attributeReference is in the comma-delimited list of literal values, defined enumeration values, or - if in use - enumeration labels.
Add		
<Number1> + <Number2>	Number	Returns the sum of <Number1> and <Number2>. The resulting data type is the more expansive of either <Number1> or <Number2>. For example, if an Integer value is added to a Decimal value, the resulting value will be a Decimal. See Precedence of rule operators on page 185.
Subtract		
<Number1> - <Number2>	Number	Subtracts <Number2> from <Number1>. The resulting data type is the more expansive of either <Number1> or <Number2>. See Precedence of rule operators on page 185.
Multiply		

Name and Syntax	Returns	Description
<Number1> * <Number2>	Number	Returns the product of <Number1> and <Number2>. The resulting data type is the more expansive of either <Number1> or <Number2>. See Precedence of rule operators on page 185 .
Divide		
<Number1> / <Number2>	Number	Divides <Number1> by <Number2>. The resulting data type is the more expansive of either <Number1> or <Number2>. See Precedence of rule operators on page 185 .
Increment		
<Number1> += <Number2>	Number	Increments <Number1> by <Number2>. The data type of <Number1> must accommodate the addition of <Number2>. See Precedence of rule operators on page 185 .
Decrement		
<Number1> -= <Number2>	Number	Decrements <Number1> by the value of <Number2>. The data type of <Number1> must accommodate the subtraction of <Number2>. See Precedence of rule operators on page 185 .
Absolute value on page 53.		
<Integer>.absVal	Number	Returns the absolute value of <Integer>. If the <Integer> is positive, <Integer> itself is returned; if <Integer> is negative, the negation of <Integer> is returned.
To Decimal		
<Integer>.toDecimal	Decimal	Converts an attribute of type Integer to type Decimal.
To String		
<Integer>.toString	String	Converts an attribute of type Integer to type String.
Maximum Value		
<Integer1>.max(<Integer2>)	Integer	Returns the greater of <Integer1> and <Integer2>.
Minimum Value		

Name and Syntax	Returns	Description
<code><Integer1>.min(<Integer2>)</code>	Integer	Returns the lesser of <code><Integer1></code> and <code><Integer2></code> .
Div		
<code><Integer1>.div(<Integer2>)</code>	Integer	Returns the whole number of times that <code><Integer2></code> fits within <code><Integer1></code> - any remainder is discarded.
Mod		
<code><Integer1>.mod(<Integer2>)</code>	Integer	Returns the whole number remainder that results from dividing <code><Integer1></code> by <code><Integer2></code> . If the remainder is a fraction, then zero is returned.
Logarithm (base 10)		
<code><Integer>.log</code>	Decimal	Returns the logarithm (base 10) of <code><Integer></code> . <code><Integer></code> may not be zero.
Logarithm (base x)		
<code><Integer>.log(<Decimal>)</code>	Decimal	Returns the logarithm (base <code><Decimal></code>) of <code><Integer></code> . <code><Integer></code> may not be zero.
Natural Logarithm		
<code><Integer>.ln</code>	Decimal	Returns the natural logarithm (base e) of <code><Number></code> . <code><Integer></code> may not be zero.
isProbablePrime(certainty)		
<code><Integer>.isProbablePrime(certainty:Integer)</code>	Boolean	Returns true if this Integer is probably prime; false if definitely is not prime.
gcd(val)		
<code><Integer>.gcd(val:Integer)</code>	Integer	Returns the greatest common divisor of the absolute value of <code>this</code> and the absolute value of <code>val</code> .
negate		
<code><Integer>.negate</code>	Integer	Returns the negative value of this integer.

String

Corticon's **String** attribute operators are as follows:

	Name and Syntax	Returns	Description
Equals (used as a comparison)			
	<String1> = <String2>	Boolean	Returns a value of true if <String1> exactly matches <String2>. Both case and length are examined to determine equality. See Character precedence: Unicode and Java Collator on page 181 for character precedence.
Equals (used as an assignment)			
	<String1> = <String2>	String	Assigns the value of <String2> to the value of <String1>.
Not Equal to			
	<String1> <> <String2>	Boolean	Returns a value of true if <String1> is not equal to <String2>.
Less than			
	<String1> < <String2>	Boolean	Returns a value of true if <String1> is less than <String2>. See Character precedence: Unicode and Java Collator on page 181 for character precedence.
Greater than on page 97			
	<String1> > <String2>	Boolean	Returns a value of true if <String1> is greater than <String2>. See Character precedence: Unicode and Java Collator on page 181 for character precedence.
Less than or Equal to			
	<String1> <= <String2>	Boolean	Returns a value of true if <String1> is less than or equal to <String2>. See Character precedence: Unicode and Java Collator on page 181 for character precedence.
Greater than or Equal to			
	<String1> >= <String2>	Boolean	Returns a value of true if <String1> is greater than or equal to <String2>. See Character precedence: Unicode and Java Collator on page 181 for character precedence.
In (Range)			

	Name and Syntax	Returns	Description
	attributeReference in [(rangeExpression)]	Boolean	Returns a value of true if attributeReference is in the range of String values <i>from..to</i> , and where opening and closing parentheses () indicate exclusion of that limit and square brackets [] indicate inclusion of that limit.
In (List)			
	attributeReference in {listExpression}	Boolean	Returns a value of true if attributeReference is in the comma-delimited list of literal values, defined enumeration values, or - if in use - enumeration labels.
Adding Strings			
	<String1> + <String2>	String	Concatenates <String1> to <String2>. Alternative syntax.
Size			
	<String>.size	String	Returns the number of characters in <String>.
Concatenate			
	<String1>.concat(<String2>)	String	Concatenates <String1> to <String2>.
Uppercase			
	<String>.toUpperCase	String	Converts all characters <String> to uppercase.
Lowercase			
	<String>.toLowerCase	String	Converts all characters in <String> to lowercase.
To DateTime			
	<String>.toDateTime	DateTime	Converts the value in <String> to data type DateTime ONLY if all characters in <String> correspond to a valid DateTime mask (format)
To Decimal			
	<String>.toDecimal	Decimal	Converts an attribute of type String to data type Decimal ONLY if all characters in <String> are numeric and contain not more than one decimal point. If any non-numeric characters are present (other than a single decimal point or leading minus sign), no value is returned.
To Integer			

	Name and Syntax	Returns	Description
	<String>.toInteger	Integer	Converts an attribute of type String to type Integer ONLY if all characters in <String> are numeric. If any non-numeric characters are present, no value is returned.
Substring			
	<String>.substring (<Integer1>,<Integer2>)	String	Returns that portion of <String> between character positions <Integer1> and Integer2.
Equals Ignoring Case			
	<String1>.equalsIgnoreCase (<String2>)	Boolean	Returns a value of true if <String1> is the same as <String2>, irrespective of case.
Starts with			
	<String1>.startsWith (<String2>)	Boolean	Returns a value of true if the <String1> begins with the characters specified in <String2>.
Ends with			
	<String1>.endsWith (<String2>)	Boolean	Evaluates the contents of <String1> and returns a value of true if the String ends with the characters specified in <String2>.
Contains			
	<String1>.contains (<String2>)	Boolean	Evaluates the contents of <String1> and returns a value of true if it contains the exact characters defined by <String2>
Equals			
	<String1>.equals (<String2>)	Boolean	Returns a value of true if <String1> is the same as <String2>.
Index Of			
	<String1>.indexOf (<String2>)	Integer	Returns the beginning character position number of <String2> within <String1>, if <String1> contains <String2>. If it does not, the function returns a value of zero.
containsBlanks			
	<String>.containsBlanks	Boolean	Determines whether the specified String contains any blanks.
characterAt(index)			

	Name and Syntax	Returns	Description
	<String>.characterAt(index:Integer)	String	Returns the character at the specified position in the String.
isInteger			
	<String>.isInteger	Boolean	Determines whether "this" String contains only integer digits. Note: This operator examines each character in a string to determine whether it is in the range 0 to 9. Therefore, the operator returns <code>true</code> when the entire string evaluates as a positive integer, and <code>false</code> when a minus sign is the first character of a string that would evaluate as a negative integer. A new extended operator could be created if the string as a whole is to be evaluated as <code>true</code> whether positive or negative (for example, by allowing the first character to be a minus sign.)
trimSpaces			
	<String>.trimSpaces	String	Trims leading and trailing spaces from "this" String.
charsIn(validSet)			
	<String>.charsIn(validSet:String)	Boolean	Determines whether "this" String contains only characters specified in the validSet.

Entity/Association operators

The Corticon rule language supports Entity and Association operators categorized as Entity, Collection, and Sequence.

Entity

Corticon's **Entity** operators are as follows:

Name and Syntax	Returns	Description
New		

Name and Syntax	Returns	Description
<Entity>.new [<Expression1>,...]	Entity	Creates a new instance of <Entity>. Expressions (optional to assign attribute values) in square brackets [...] must be written in the form: <i>attribute = value</i> .
New Unique		
<Entity>.newUnique [<Expression1>,...]	Entity	Creates a new instance of <Entity> only if the instance created is unique as defined by optional <Expression1>,...
Clone		
<Entity>.clone [<Expression1>,...]	Entity	Creates a new instance of <Entity> with the same attributes and their respective values. Expressions (optional to override attribute values) in square brackets [...] must be written in the form: <i>attribute = value</i> .
Remove		
<Entity>.remove	Entity	Deletes the entity from memory and from the resultant XML document.

Collection

Corticon's **Collection** operators are as follows:

Name and Syntax	Returns	Description
Replace element(s)		
<Collection1> = <Collection2> <Collection1> = <Entity>	<i>modifies a collection</i>	replaces all elements in <Collection1> with elements of <Collection2> or with <Entity>, provided the new associations are allowed by the Business Vocabulary.
Associate element(s)		
<Collection1> += <Collection2> <Collection1> += <Entity>	<i>modifies a collection</i>	Associates all elements of <Collection2> or <Entity> with <Collection1>. Every <Collection> must be expressed as a unique alias.
Disassociate element(s)		
<Collection1> -= <Collection2>	<i>modifies a collection</i>	Disassociates all elements of <Collection2> from <Collection1>. Does not delete the disassociated elements. Every <Collection> must be expressed as a unique alias.

Name and Syntax	Returns	Description
Is empty		
<Collection> ->isEmpty	Boolean	Returns a value of true if <Collection> contains <i>no</i> elements
Not empty		
<Collection> ->notEmpty	Boolean	Returns a value of true if <Collection> contains <i>at least one</i> element.
Exists		
<Collection> ->exists (<Expression>)	Boolean	Returns a value of true if <Expression> holds true for <i>at least one</i> element of <Collection>
For all		
<Collection> ->forAll (<Expression>)	Boolean	Returns a value of true if <i>every</i> <Expression> holds true for <i>every</i> element of <Collection>
Sorted by		
<Collection> ->sortedBy (<Attribute>)	<i>converts a collection into a sequence</i>	Sequences the elements of <Collection> in <u>ascending</u> order, using the value of <Attribute> as the index. <Collection> must be expressed as a unique alias.
Sorted by descending		
<Collection> ->sortedByDesc (<Attribute>)	<i>converts a collection into a sequence</i>	Sequences the elements of <Collection> in <u>descending</u> order, using the value of <Attribute> as the index. <Collection> must be expressed as a unique alias.
Iterate		
<Collection> ->iterate(<Expression>)		Executes <Expression> for every element in <Collection>. <Collection> must be expressed as a unique alias.
Size of collection		
<Collection> ->size	Integer	Returns the number of elements in <Collection>. <Collection> must be expressed as a unique alias.
Sum		

Name and Syntax	Returns	Description
<Collection.attribute> ->sum	Number	Sums the values of the specified <attribute> for all elements in <Collection>. <attribute> must be a numeric data type.
Average		
<Collection.attribute> ->avg	Number	Averages all of the specified attributes in <Collection>. <Collection> must be expressed as a unique alias. <attribute> must be a numeric data type
Minimum		
<Collection.attribute> ->min	Number	Returns the lowest value of <attribute> for all elements in <Collection>. <attribute> must be a numeric data type
Maximum		
<Collection.attribute> ->max	Number	Returns the highest value of <attribute> for all elements in <Collection>. <attribute> must be a numeric data type
toSet		
Collection.toSet	String	Returns a single String that is the set of Strings in this collection.
allContain(lookFor)		
Collection.allContain (lookFor:String)	Boolean	Determines whether all the strings in this collection contain the lookFor String
uniqueCount		
Collection.uniqueCount	Integer	Returns the count of the unique Strings in this collection.

Sequence

Sequence operators act on collections that have *already* been ordered by a sorting operator (see [sortedBy](#) and [sortedByDesc](#)). In other words, sequence operators operate on collections that have been turned into sequences. The notation <Sequence> used below, is shorthand for a completed sorting operation. For example:

<Collection> -> sortedBy(<Attribute>)

produces a <Sequence>, in this case the elements of <Collection> arranged in ascending order using <Attribute> as the index. This <Sequence> can then be used with one of the sequence operators described below. The design of the Object Constraint Language (upon which the Corticon Rule Language is based), allows for the “chaining” of operators, so a collection operator and a sequence operator can be used in the same expression to produce a sequence and identify a particular element of that sequence in the same step. For example:

```
<Entity.attribute1> = <Collection> ->sortedBy(<Attribute3>) ->first.<Attribute2>
```

performs the following:

1. Sorts <Collection> in ascending order according to <Attribute3>, turning it into a <Sequence>
2. Locates the first element of <Sequence>
3. Reads the value of <Attribute2> of the first element
4. Assigns the value of <Attribute2> of the first element to <Entity.attribute1>

Corticon's **Sequence** operators are as follows:

	Name and Syntax	Returns	Description
At			
	<Sequence> ->at(<Integer>)	Entity	Returns the element at position <Integer>. <Sequence> must be expressed as a unique alias.
First			
	<Sequence> ->first	Entity	Returns the first element of <Sequence>. <Sequence> must be expressed as a unique alias.
Last			
	<Sequence> ->last	Entity	Returns the last element of <Sequence>. <Sequence> must be expressed as a unique alias.
SubSequence			
	<Sequence> ->subSequence(<integer1, integer2>)	Entity	Returns a Sequence containing all elements of <Sequence> between the positions <i>integer1</i> and <i>integer2</i> .
First(number)			
	<Sequence> ->first(<integer>)	Entity	Returns a Sequence containing elements of <Sequence> from the first element to <i>integer</i> ; in other words, ->first(x) is effectively >subSequence(1, x)
Last(number)			

	Name and Syntax	Returns	Description
	<Sequence> ->last(<i>integer</i>)	Entity	Returns a Sequence containing elements of <Sequence> between the end position of the collection and <i>integer</i> ; in other words, in a sequence of n elements, ->last(x) is effectively >subSequence(n-x+1, n)
Trend			
	<Attribute> -> <Sequence>.trend	String	Returns a 4-character string, INCR, DECR, CNST, or NONE depending on the trend of <Attribute> within <Sequence>.
mavg(elements)			
	<Sequence.decimal> .mavg (elements:Integer)	Decimal	Returns a single decimal value that is the average of the number of elements specified.
Sorted Alias: next			
	->next		Operates against a Sorted Alias (a special cached Sequence) inside a filter expression. The Rulesheet is set into a Ruleflow that iterates to bind the alias in each successive invocation to the next element in the sequence. For more information, see the topic "Sorted Alias" in the Collections chapter of the Corticon Studio: Rule Modeling Guide..

Extended Operators

Corticon's **Extended** operators are categorized as **RandomGenerator** and **SeMath**.

RandomGenerator

RandomGenerator does just what it says: it generates -- and then returns -- a random number.

Corticon's **RandomGenerator** operator is as follows:

Name and Syntax	Returns	Description
getRandomNumber		
RandomGenerator. getRandomNumber	Decimal	Returns the next random number out of the current static Random object.

SeMath

Corticon's **SeMath** operators are as follows:

Name and Syntax	Returns	Description
getCircumference		
SeMath.getCircumference (radius:<Decimal>)	Decimal	Converts the specified radius to circumference.
getFahrenheit		
SeMath.getFahrenheit (centigrade:<Decimal>)	Decimal	Converts the specified Centigrade temperature to Fahrenheit.
replaceString		
SeMath.replaceString (lookin:<String>, lookfor:<String> , replacewith:<String>)	String	Replaces all occurrences of a specified string with another string.

4

Rule operator details and examples

The following pages describe each operator in greater detail. Each Rule Operator has the following sections

- 1. Syntax** – Describes the standard syntax used with this operator. In this section, as in the previous summary tables, the angle bracket convention `< . . >` is used to indicate what types of terms and their data types can be used with the operator. When using the operator with real terms from the Vocabulary, do not include the angle brackets.
- 2. Description** – Provides a plain-language description of the operator's purpose and details of its use. Important reminders, tips, or cautions are included in this section.
- 3. Usage Restrictions** – Describes what limitations exist for this operator, and where an operator may not be used in a Rulesheet. Such limitations are rare, but important to a good understanding of Corticon Studio.
- 4. Example** – Shows an example of each operator in a Rulesheet. A screenshot of the example Rulesheet is provided, with portions of the Rulesheet not used by the example collapsed or truncated for clarity. The example also includes sample input and output data for Ruletest scenarios run against the Rulesheet.

The entire list of operators is presented in alphabetic order.

For details, see the following topics:

- [Absolute value](#)
- [Add numbers](#)
- [Add strings](#)
- [Add days](#)
- [Add hours](#)
- [Add minutes](#)

- Add months
- Add seconds
- Add years
- Associate element(s)
- At
- Average
- CellValue
- Clone
- Concatenate
- Contains
- Day
- Days between
- Day of week
- Day of year
- Decrement
- Disassociate element(s)
- Divide
- Div
- Ends with
- Equals when used as an assignment
- Equals when used as a comparison
- Equals ignoring case
- Equals when using Strings
- Exists
- Exponent
- False
- First
- First (number)
- Floor
- For all
- Greater than
- Greater than or equal to
- Hour

-
- Hour between
 - In {List}
 - In (Range)
 - Increment
 - Index of
 - Is empty
 - Iterate
 - Last
 - Last (number)
 - Less than
 - Less than or equal to
 - Logarithm (Base 10)
 - Logarithm (Base x)
 - Lowercase
 - Maximum value
 - Maximum value (Collection)
 - Minimum value
 - Minimum value (Collection)
 - Minute
 - Minutes between
 - Mod
 - Month
 - Months between
 - Multiply
 - Natural logarithm
 - New
 - New unique
 - Not
 - Not empty
 - Not equal to
 - Now
 - Null
 - Other

- Or
- Remove element
- Replace element(s)
- Round
- Second
- Seconds between
- Size of string
- Size of collection
- Sorted by
- Sorted by descending
- Starts with
- SubSequence
- Substring
- Subtract
- Sum
- Today
- To date: Casting a dateTime to a date
- To dateTime: Casting a string to a dateTime
- To dateTime: Casting a date to a dateTime
- To dateTime: Casting a time to a dateTime
- To dateTime: Timezone offset
- To decimal
- To integer
- To string
- To time: Casting a dateTime to a time
- Trend
- True
- Uppercase
- Week of month
- Week of year
- Year
- Years between

Absolute value

SYNTAX

<Number>.absVal

DESCRIPTION

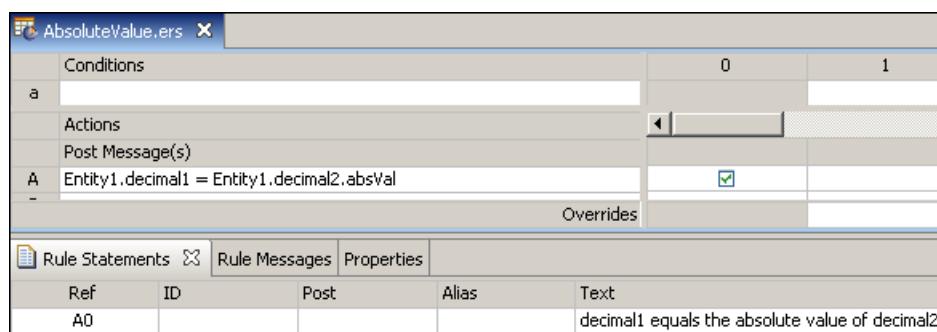
Returns the absolute value of <Number>. If the <Number> is positive, <Number> itself is returned; if <Number> is negative, the negation of <Number> is returned.

USAGE RESTRICTIONS

The Operators row in the table of [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

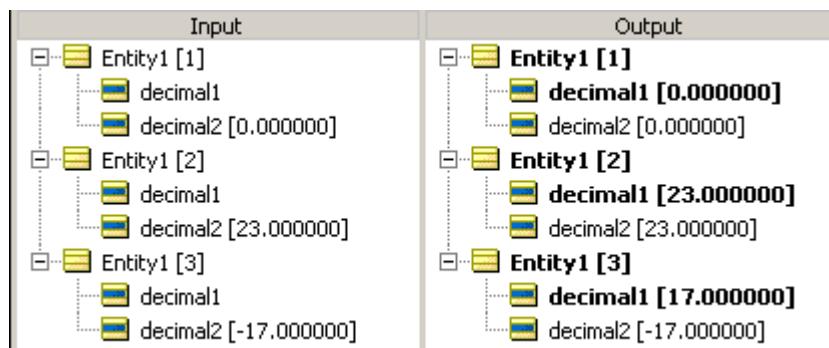
RULESHEET EXAMPLE

This sample Rulesheet uses **.absVal** to produce the absolute value of decimal2 and assign it to decimal1



SAMPLE RULETEST

A sample Ruletest provides decimal2 values for three different scenarios of Entity1. Input and Output panels are shown below.



Add numbers

SYNTAX

<Number1> + <Number2>

DESCRIPTION

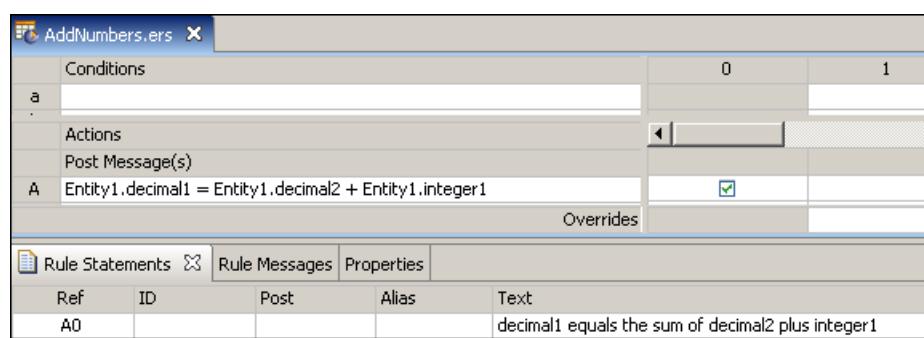
Adds <Number1> to <Number2>. The resulting data type is the more expansive of those of <Number1> and <Number2>. For example, if you are adding an Integer value and a Decimal value, the resulting value will be a Decimal. See [Precedence of rule operators](#) on page 185.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

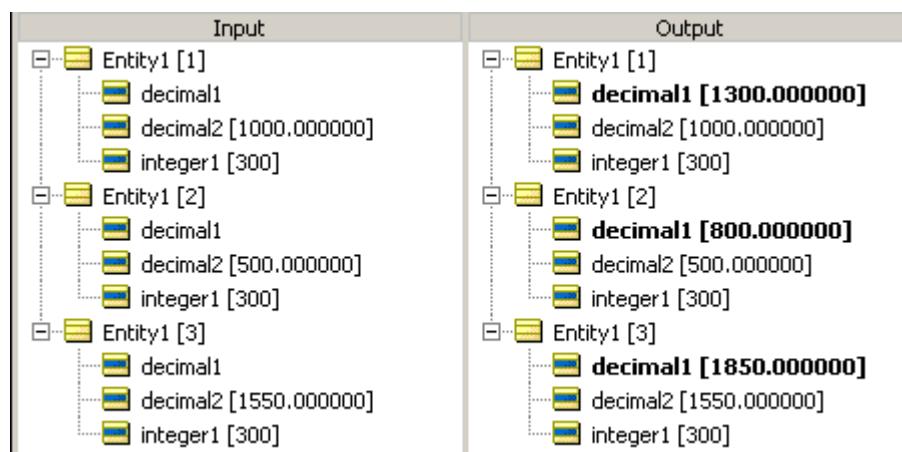
RULESHEET EXAMPLE

This sample Rulesheet uses the **add numbers** operation to add the value of `decimal2` to the value of `integer1` and assign the result to `decimal1`



SAMPLE RULETEST

A sample Ruletest provides an `integer1` value of 300 which is added to the value of `decimal2` and assigned to the value of `decimal1` for three instances of `Entity1`. Input and Output panels are shown below.



Add strings

SYNTAX

<String1> + <String2>

DESCRIPTION

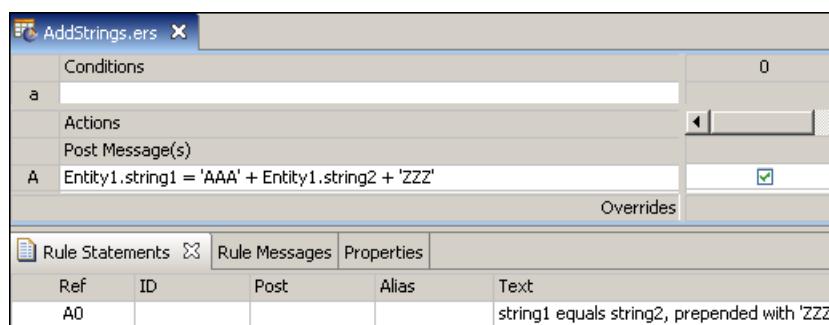
Adds <String1> to <String2>. This has the same effect as using the `.concat` operator. However, the “+” syntax permits concatenation of more than two String values without nesting, as shown in the example below.

USAGE RESTRICTIONS

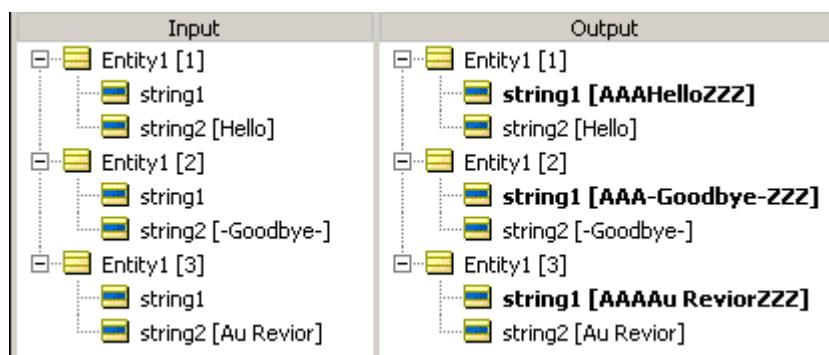
The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

This sample Rulesheet uses **add strings** operation to add the String AAA to string2 to ZZZ and assign the result to string1



SAMPLE RULETEST



Add days

SYNTAX

```
<DateTime>.addDays(<Integer>)
<Date>.addDays(<Integer>)
```

DESCRIPTION

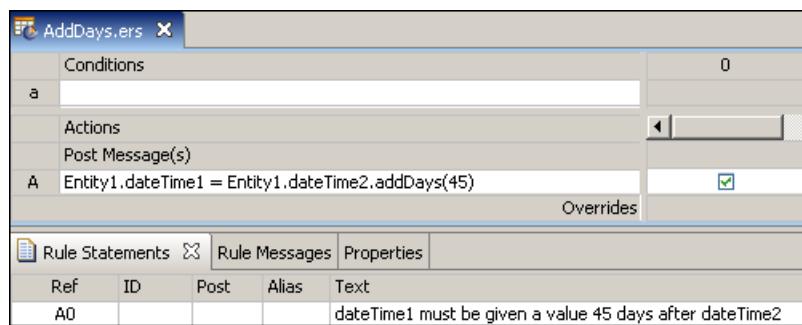
Adds the number of days in <Integer> to the number of days in <DateTime> or <Date>.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

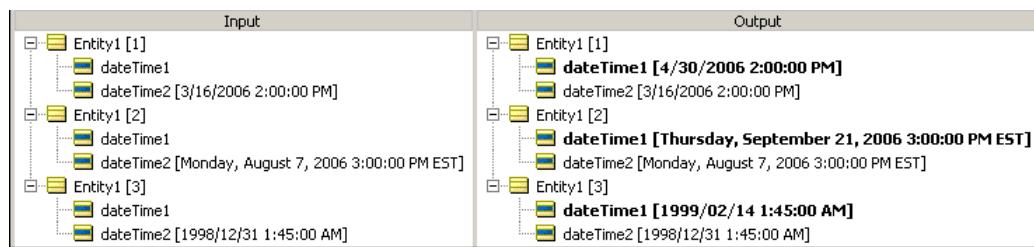
RULESHEET EXAMPLE

This sample Rulesheet uses **.addDays** to add 45 days to the value of `dateTime2` and assign the result to `dateTime1`.



SAMPLE RULETEST

A sample Ruletest provides values of `dateTime2` for three instances of `Entity1`. Input and Output panels are shown below. Notice the month portion of `dateTime1` also changes accordingly.



Add hours

SYNTAX

```
<DateTime>.addHours(<Integer>)
<Time>.addHours(<Integer>)
```

DESCRIPTION

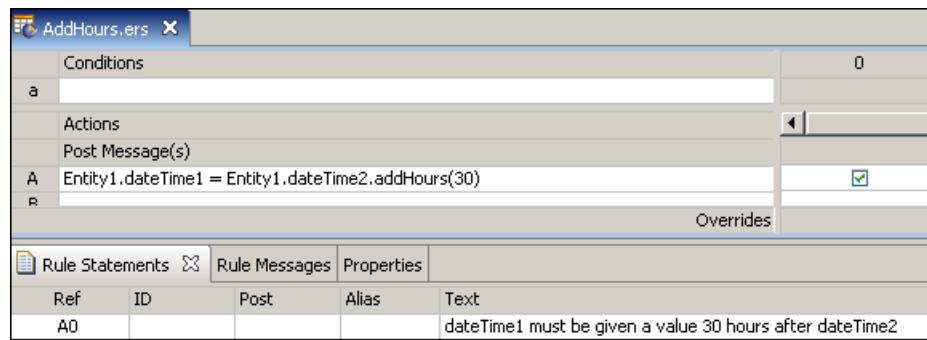
Adds the number of hours in <Integer> to the number of hours in the Time portion of <DateTime> or <Time>.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

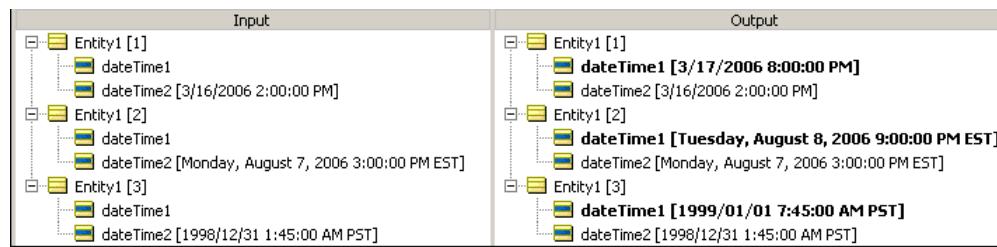
RULESHEET EXAMPLE

This sample Rulesheet uses the **. addHours** to add 30 hours to the value of `dateTime2` and assign the result to `dateTime1`.



SAMPLE RULETEST

A sample Ruletest provides values of `dateTime2` for three instances of `Entity1`. Input and Output panels are shown below.



Add minutes

SYNTAX

```
<DateTime>.addMinutes(<Integer>)
<Time>.addMinutes(<Integer>)
```

DESCRIPTION

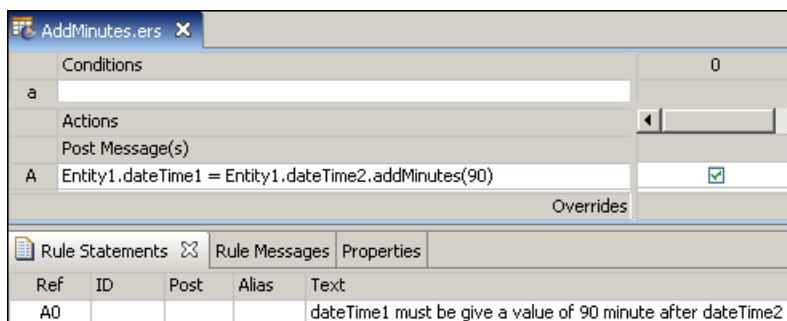
Adds the number of minutes in <Integer> to the number of minutes in the Time portion of <DateTime> or <Time>.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

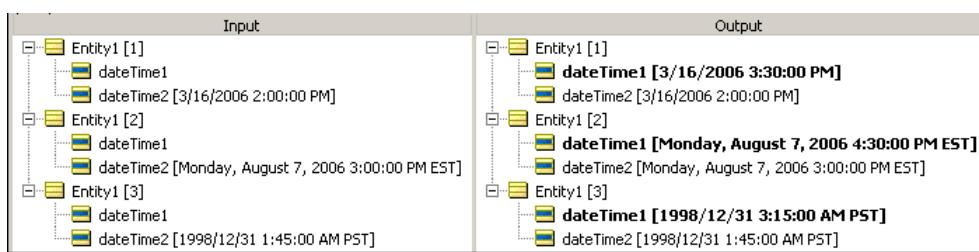
RULESHEET EXAMPLE

This sample Rulesheet uses the . **addMinutes** add 90 minutes to the value of `dateTime2` and assign the result to `dateTime1`.



SAMPLE RULETEST

A sample Ruletest provides values of `dateTime2` for three instances of `Entity1`. Input and Output panels are shown below.



Add months

SYNTAX

```
<DateTime>.addMonths(<Integer>)
<Date>.addMonths(<Integer>)
```

DESCRIPTION

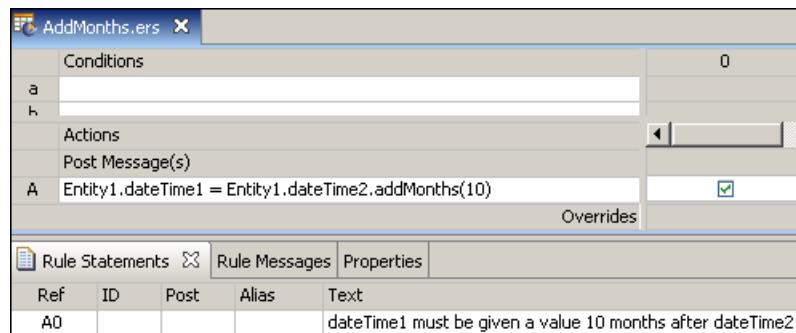
Adds the number of months in <Integer> to the number of months in <DateTime> or <Date>.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

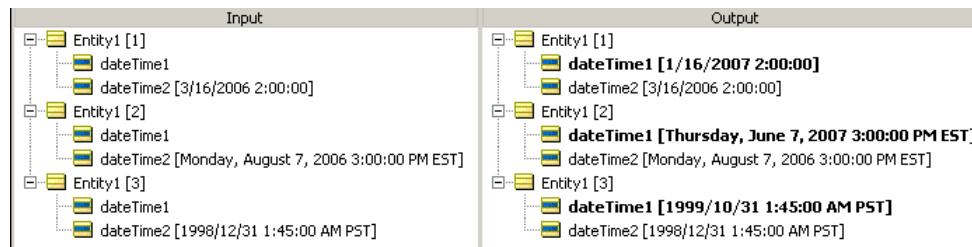
RULESHEET EXAMPLE

This sample Rulesheet uses . **addMonths** in a Nonconditional rule to add 10 months to the value of `dateTime2` and assign the result to `dateTime1`.



SAMPLE RULETEST

A sample Ruletest provides values of `dateTime2` for three instances of `Entity1`. Input and Output panels are shown below. Notice the year portion of `dateTime1` also changes accordingly.



Add seconds

SYNTAX

```
<DateTime>.addSeconds(<Integer>)
<Time>.addSeconds(<Integer>)
```

DESCRIPTION

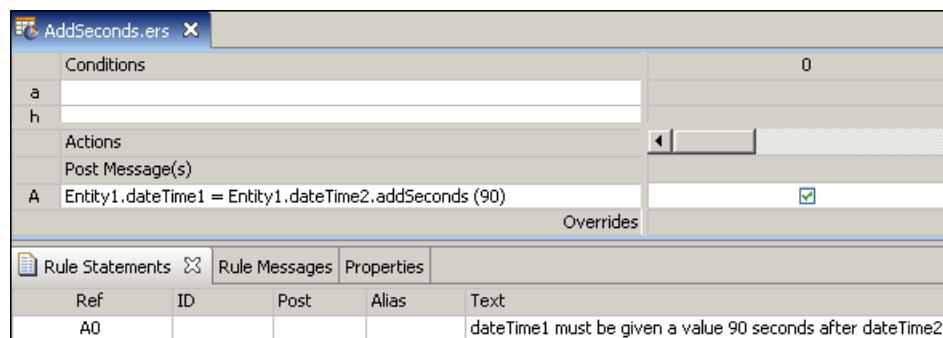
Adds the number of seconds in <Integer> to the number of seconds in the Time portion of <DateTime> or <Time>.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

This sample Rulesheet uses **.addSeconds** in a Nonconditional rule to add 90 seconds to the value of timeOnly2 and assign the result to timeOnly1.



SAMPLE RULETEST

A sample Ruletest provides values of timeOnly2 for three instances of Entity1. Input and Output panels are shown below. Notice how the time “wraps” around to the beginning of the day, even though Time data type does not include date information.

Input		Output	
Entity1 [1]	dateTime1 dateTime2 [3/16/2006 2:00:00 PM]	Entity1 [1]	dateTime1 [3/16/2006 2:01:30 PM] dateTime2 [3/16/2006 2:00:00 PM]
Entity1 [2]	dateTime1 dateTime2 [Monday, August 7, 2006 3:00:00 PM EST]	Entity1 [2]	dateTime1 [Monday, August 7, 2006 3:01:30 PM EST] dateTime2 [Monday, August 7, 2006 3:00:00 PM EST]
Entity1 [3]	dateTime1 dateTime2 [1998/12/31 1:45:00 AM PST]	Entity1 [3]	dateTime1 [1998/12/31 1:46:30 AM PST] dateTime2 [1998/12/31 1:45:00 AM PST]

Add years

SYNTAX

```
<DateTime>.addYears(<Integer>)
<Date>.addYears(<Integer>)
```

DESCRIPTION

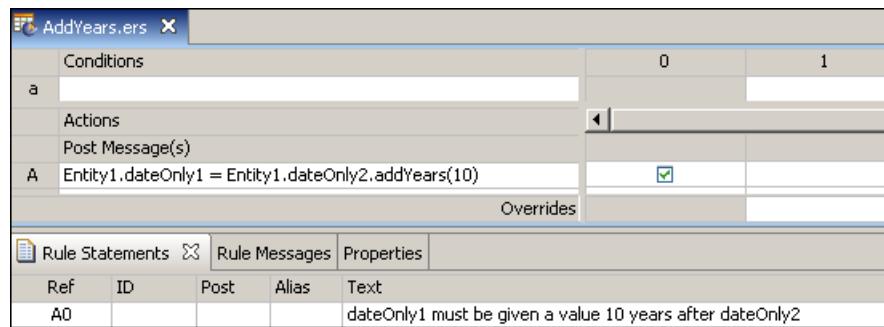
Adds the number of years in <Integer> to the number of years in the Date portion of <DateTime> or <Date>.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

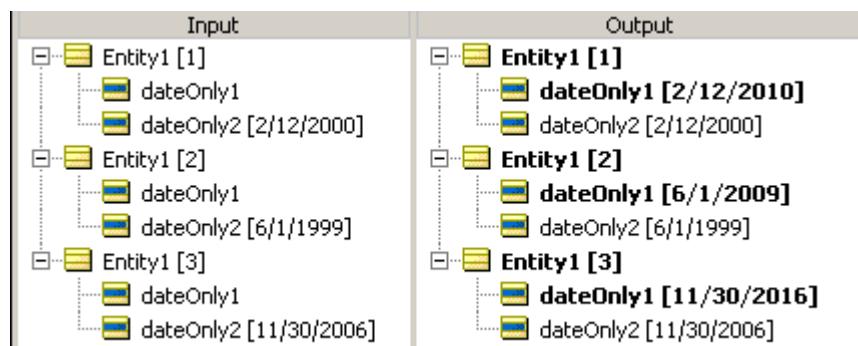
RULESHEET EXAMPLE

This sample Rulesheet uses **.addYears** in a Nonconditional rule to add 10 years to the value of `dateOnly2` and assign the result to `dateOnly1`.



SAMPLE RULETEST

A sample Ruletest provides values of `dateOnly2` for three instances of `Entity1`. Input and Output panels are shown below.



Associate element(s)

SYNTAX

```
<Collection1> += <Collection2>
<Collection1> += <Entity>
```

DESCRIPTION

Associates all elements of <Collection2> or a single element named <Entity> with <Collection1>, provided such an association is allowed by the Vocabulary. Every <Collection> must be expressed as a unique alias.

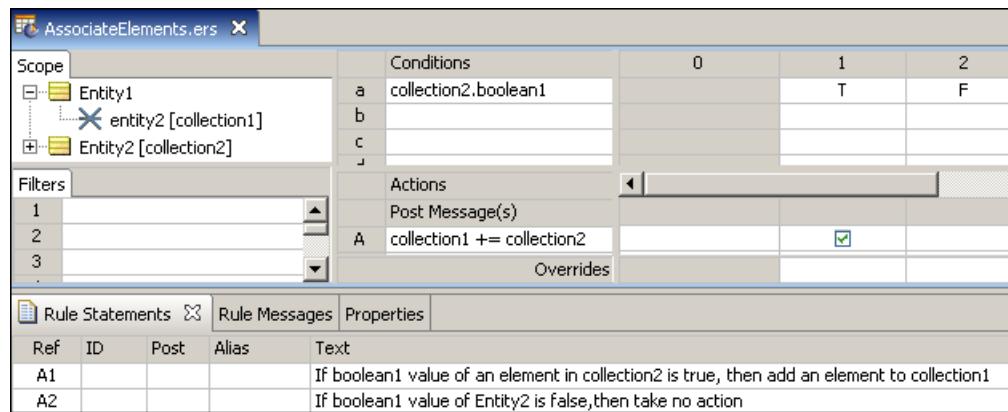
If the cardinality of the association between the parent entity of <Collection> and the <Entity> being added is “one-to-one” (a straight line icon beside the association in the Rule Vocabulary), then this **associate element** syntax is not used. Instead, **replace element** syntax is used, since the collection can contain only one element, and any element present will be replaced by the new element.

USAGE RESTRICTIONS

The Operators row of the table in [Vocabulary usage restrictions](#) does not apply. Special exceptions: **associate element** may only be used in Action Rows (section 5 in [Sections of Rulesheet that correlate with usage restrictions](#)).

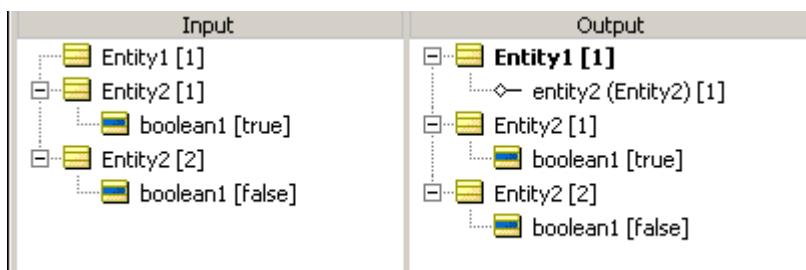
RULESHEET EXAMPLE

The following Rulesheet uses **associate element** to associate an element of collection2 to collection1 when boolean1 value of any element in collection2 is true. Note that the Action is not associating *all* elements in collection2 with collection1, *only* those elements within collection2 that satisfy the condition.



SAMPLE RULETEST

A sample Ruletest provides two examples of Entity2 with boolean1 values, and a single Entity1. Input and Output panels are described below:



At

SYNTAX

<Sequence> ->**at(<Integer>).<Attribute1>**

DESCRIPTION

Returns the value of <Attribute1> for the element at position <Integer> in <Sequence>. Another operator, such as [->sortedBy](#), must be used to transform a <Collection> into a <Sequence> before **->at** may be used. <Sequence> must be expressed as a unique alias. See [Advanced Collection Syntax](#) for more examples of usage.

<Attribute1> may be of any data type.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

This sample Rulesheet uses **->at(2)** to identify the second element of the sequence created by applying [sortedBy](#) to collection1. Once identified, the value of the string1 attribute belonging to this second element is evaluated. If the value of string1 is Joe, then boolean1 attribute of Entity1 is assigned the value of true.

Scope:

- Entity_1
 - boolean1
 - decimal1
- entity_2 (Entity_2) [collection1]

Conditions:

	Conditions	0	1	2
a	collection1->sortedBy(decimal1)->at(2).string1	'Joe'	not 'Joe'	
b				
c				
d				

Actions:

	Actions	T	F
A	Entity_1.boolean1		
B			
C			

Overrides:

Rule Statements:

Ref	ID	Post	Alias	Text	Rule Name	Run
1				If the string1 value of the 2nd element in collection1, in ascending order by decimal1, is equal to Joe, then boolean1 is true.		
2				If the string1 value of the 2nd element in collection1, in ascending order by decimal1, is NOT equal to Joe, then boolean1 is false.		

SAMPLE RULETEST

A sample Ruletest provides a collection of three elements, each with a decimal1 value. Input and Output panels are shown below.

/Examples/at.ers	
Input	Output
<ul style="list-style-type: none"> Entity_1 [1] <ul style="list-style-type: none"> boolean1 entity_2 (Entity_2) [1] <ul style="list-style-type: none"> decimal1 [2.500000] string1 [Sally] entity_2 (Entity_2) [2] <ul style="list-style-type: none"> decimal1 [5.800000] string1 [Moe] entity_2 (Entity_2) [3] <ul style="list-style-type: none"> decimal1 [3.300000] string1 [Joe] 	<ul style="list-style-type: none"> Entity_1 [1] <ul style="list-style-type: none"> boolean1 [true] entity_2 (Entity_2) [1] <ul style="list-style-type: none"> decimal1 [2.500000] string1 [Sally] entity_2 (Entity_2) [2] <ul style="list-style-type: none"> decimal1 [5.800000] string1 [Moe] entity_2 (Entity_2) [3] <ul style="list-style-type: none"> decimal1 [3.300000] string1 [Joe]

Average

SYNTAX

<Collection.attribute> ->avg

DESCRIPTION

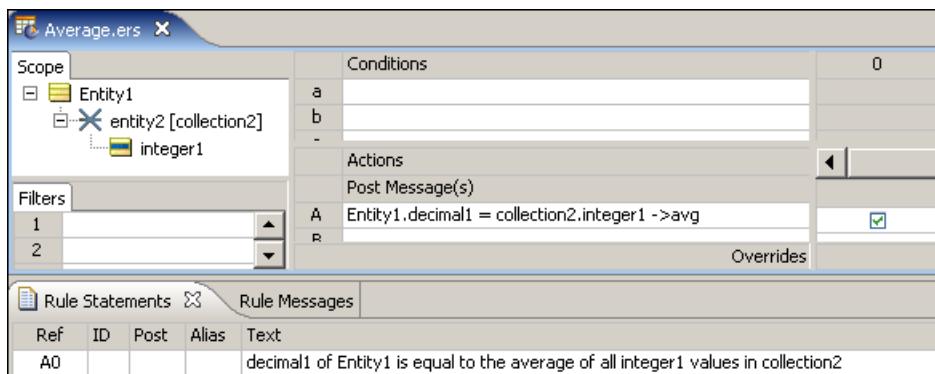
Averages the values of all of the specified attributes in <Collection>. <Collection> must be expressed as a unique alias. <attribute> must be a numeric data type.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

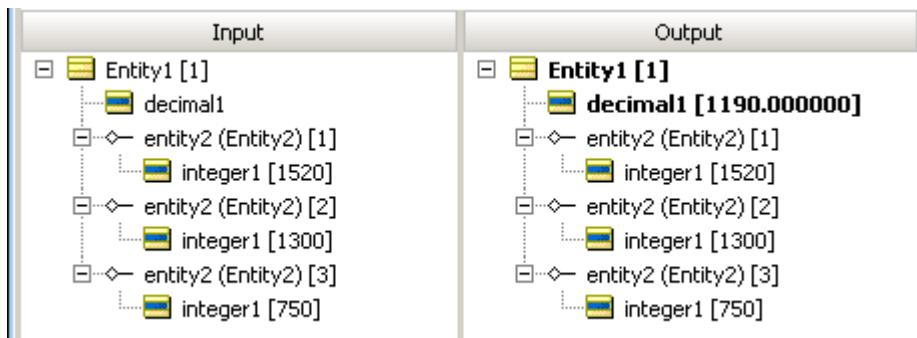
RULESHEET EXAMPLE

This sample Rulesheet uses **->avg** to average the integer1 values of all elements in collection2, then assigns the resulting value to decimal1 in Entity1. Note the use of the alias collection2 to represent the collection of Entity2 elements associated with Entity1.



SAMPLE RULETEST

A sample Ruletest provides `integer1` values for three elements in `collection2`. The following illustration shows Input and Output panels:



CellValue

SYNTAX

Various, see Examples below

DESCRIPTION

When used in an expression, **CellValue** is essentially a variable whose value is determined by the rule Column that executes. Using **CellValue** in a Condition or Action expression eliminates the need for multiple, separate Rows to express the same logic.

USAGE RESTRICTIONS

The Operators row of the table in [Vocabulary usage restrictions](#) does not apply. Special exceptions: **CellValue** may only be used in Condition and Action Rows (sections 3 and 5 in [Sections of Rulesheet that correlate with usage restrictions](#)).

RULESHEET EXAMPLE 1

This sample Rulesheet uses **CellValue** to increment `integer1` by the amount in the Action Cell of the rule Column that fires. An equivalent Rulesheet which does not use **CellValue** is also shown for comparison purposes.

The screenshot shows the CellValue1.ers Rulesheet editor. It has two main sections: Conditions and Actions.

Conditions:

	0	1	2
a Entity1.boolean1		T	F
b			

Actions:

	3	6
A Entity1.integer1 += cellValue		
B		

Overrides:

|--|--|--|

Rule Statements:

Ref	ID	Post	Alias	Text
1				If boolean1 is true, increment integer1 by 3
2				If boolean1 is false, increment integer1 by 6

Equivalent Rulesheet without using **cellValue**:

The screenshot shows the CellValue2.ers Rulesheet editor. It has two main sections: Conditions and Actions.

Conditions:

	0	1	2
a Entity1.boolean1		T	F
b			

Actions:

Post Message(s)		
A Entity1.integer1 +=3		<input checked="" type="checkbox"/>
B Entity1.integer1 +=6		<input checked="" type="checkbox"/>

Overrides:

|--|--|--|

Rule Statements:

Ref	ID	Post	Alias	Text
1				If boolean1 is true, increment integer1 by 3
2				If boolean1 is false, increment integer1 by 6

SAMPLE RULETEST 1

A sample Ruletest provides two examples of boolean1. The following table shows Input and Output panels.

The screenshot shows the Input and Output panels for Sample Ruletest 1. The Input panel shows two entities, Entity1[1] and Entity1[2]. Entity1[1] contains boolean1 [true] and integer1 [2]. Entity1[2] contains boolean1 [false] and integer1 [4]. The Output panel shows two entities, Entity1[1] and Entity1[2]. Entity1[1] contains boolean1 [true] and integer1 [5]. Entity1[2] contains boolean1 [false] and integer1 [10].

RULESHEET EXAMPLE 2

The following Rulesheet uses **cellValue** to evaluate whether collection1 includes at least one member with a string1 value of the entry in the Conditions Cell of the rule Column.

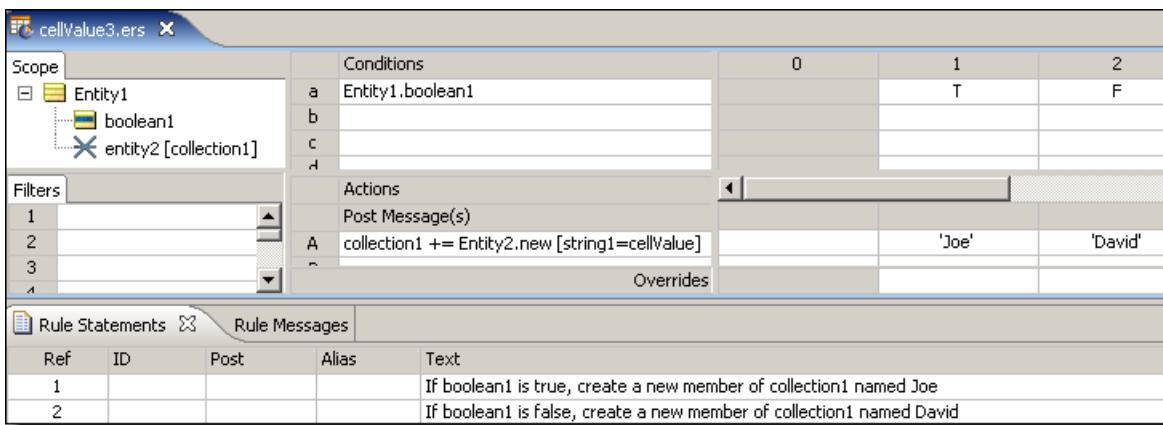
SAMPLE RULETEST 2

A sample Ruletest provides three examples of collection1 – each member has a string1 value. Input and Output panels are shown below.

Severity	Message	Entity
Info	We have a David	Entity1[1]
Info	We have a Jennifer	Entity1[1]

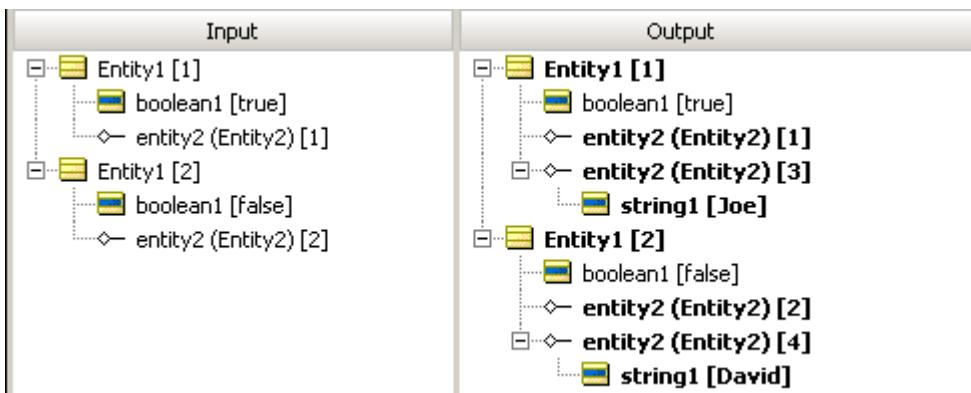
RULESHEET EXAMPLE 3

The following Rulesheet uses **CellValue** to create a new member of collection1 with string1 value equal to the Action Cell in the rule Column that fires.



SAMPLE RULETEST 3

A sample Ruletest provides `string1` values for three examples. The following illustration shows Ruletest Input and Output panels. Notice that each `collection1` already has one element prior to executing the test. This simply ensures the results will be displayed in hierarchical style.



Clone

SYNTAX

`<Entity>.clone[<Expression1>,<Expression2>...]`

DESCRIPTION

Copies the specified Entity and its attribute values to a new Entity where Expressions (in the form `attribute=value`) override the corresponding cloned attribute values. The new Entity has no associations. Where an Entity specifies an Entity Identity, that identity is not copied to its clone entity. For each Entity in Collection, the operator creates a duplicate of Entity. The implementation is a *shallow clone* -- associations are not duplicated.

Note: If the cloned entity is database-enabled and contains primary keys, the primary key values must be specified in the qualifier clause or an exception will occur. If an Entity uses a Datastore Identity as its Identity Strategy, a new identifier is created by the database for each clone.

USAGE RESTRICTIONS

The Operators row in the table of [Summary Table of Vocabulary Usage Restriction](#) does not apply. Special exceptions: **clone** may only be used in Action Rows (section 5 in [Sections of Rulesheet that correlate with usage restrictions](#)).

Nested clone calls are not supported, such as `E1.clone[assoc1 += E1.assoc1.clone[...]]`.

RULESHEET EXAMPLE

The following Rulesheet uses **.clone** to create a new Entity2 element when the value of `qtyOrdered` in Entity1 is greater than the `qtyShipped` value. An alias is not required by the **.clone** operator, because it is possible to create a new entity at the root level, without inserting it into a collection.

SAMPLE RULETEST

A sample Ruletest provides two collections of Entity1. Input, Output, and Expected panels are as follows:

Severity	Message	Entity
Warning	Incomplete shipment. Backorder was generated.	Entity_1[2]

RULESHEET EXAMPLE: COLLECTION

The following Rulesheet uses `.clone` to create a new Entity2 element in collection1 when Entity1 has a non-zero qtyOrdered value.

SAMPLE RULETEST: COLLECTION

A sample Ruletest provides three collections of Entity1. Input and Output panels are illustrated below:

Concatenate

SYNTAX

`<String1>.concat(<String2>)`

DESCRIPTION

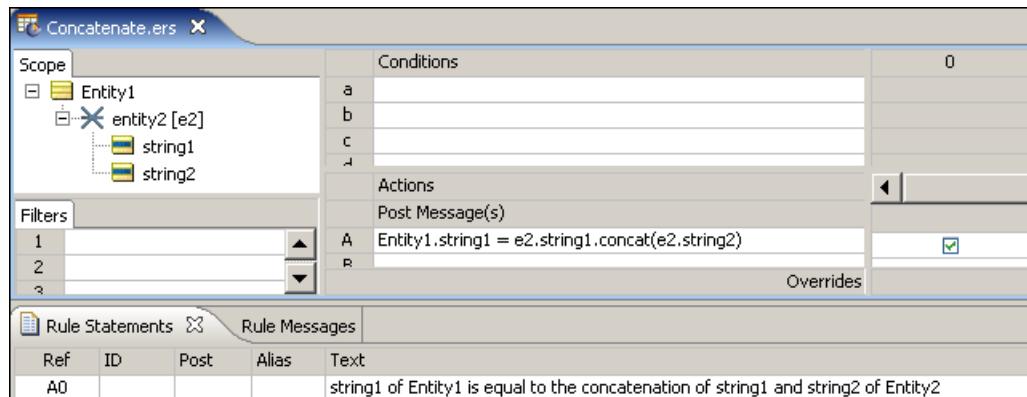
Concatenates `<String1>` to `<String2>`, placing `<String2>` at the end of `<String1>`

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions

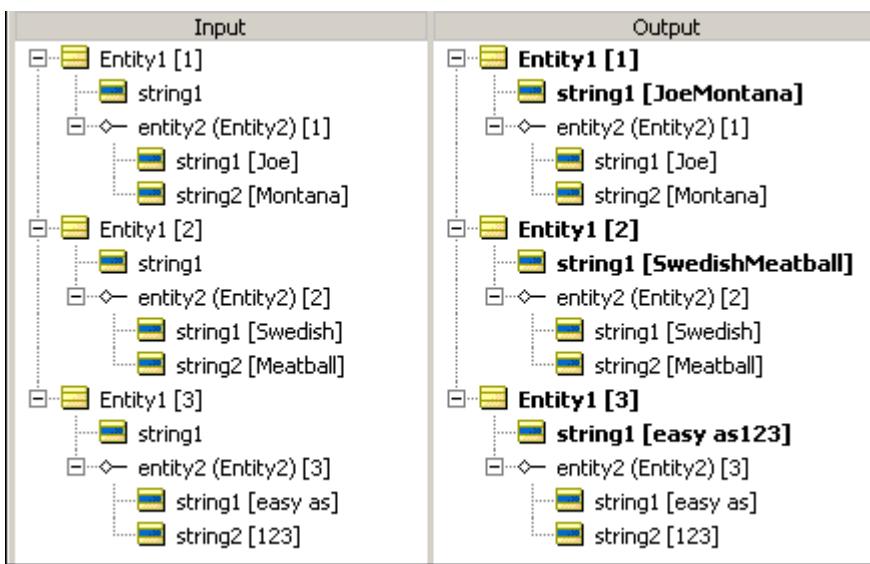
RULESHEET EXAMPLE

This sample Rulesheet uses `.concat` to create `string1` by combining `string1` and `string2` from `Entity1.entity2.`



SAMPLE RULETEST

A sample Ruletest provides three examples of `string1` and `string2`. Input and Output panels are shown below.



Contains

SYNTAX

```
<String1>.contains(<String2>)
```

DESCRIPTION

Evaluates <String1> and returns a value of true if it contains or includes the exact (case-sensitive) characters specified in <String2>.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions

RULESHEET EXAMPLE 1

The following uses .contains to evaluate whether string1 includes the characters `silver` and assigns a value to `boolean1` for each outcome.

Ref	ID	Post	Alias	Text
A1		Info	Entity1	String1 contains the word 'silver'
A2		Info	Entity1	String1 does not contain the word 'silver'

SAMPLE RULETEST 1

A sample Ruletest provides `string1` values for three examples. Input and Output panels are shown below. Note case sensitivity in these examples. Posted messages are not shown.

The screenshot shows the Progress Corticon Rule Language Editor interface. The top section is divided into two panels: 'Input' on the left and 'Output' on the right. Both panels show a hierarchical tree structure. In the Input panel, there are three entities labeled Entity1 [1], Entity1 [2], and Entity1 [3]. Each entity has a boolean1 node and a string1 node. The string1 values are 'Hi Ho Silver', 'hi ho silver', and 'silvery' respectively. In the Output panel, the same three entities are shown, but their boolean1 values are explicitly set to false, true, and true respectively, while the string1 values remain the same. Below these panels is a tabbed area with 'Rule Statements' selected. The 'Rule Statements' tab contains a table with three rows:

Severity	Message	Entity
Info	String1 contains the word 'silver'	Entity1[3]
Info	String1 contains the word 'silver'	Entity1[2]
Info	String1 does not contain the word 'silver'	Entity1[1]

Day

SYNTAX

`<DateTime>.day`

`<Date>.day`

`,`

DESCRIPTION

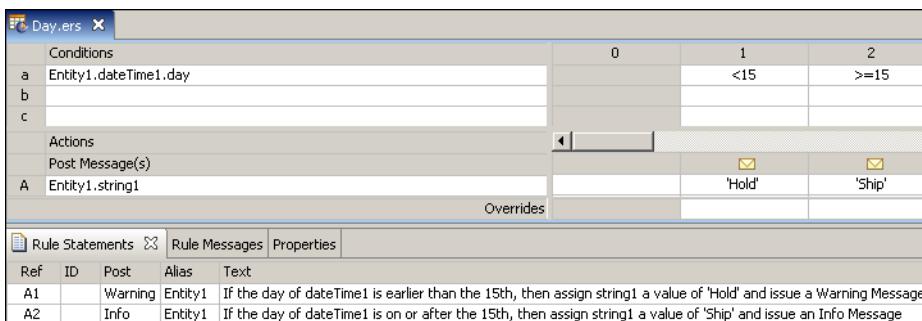
Returns the day portion of `<DateTime>` or `<Date>` as an Integer between 1 and 31.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

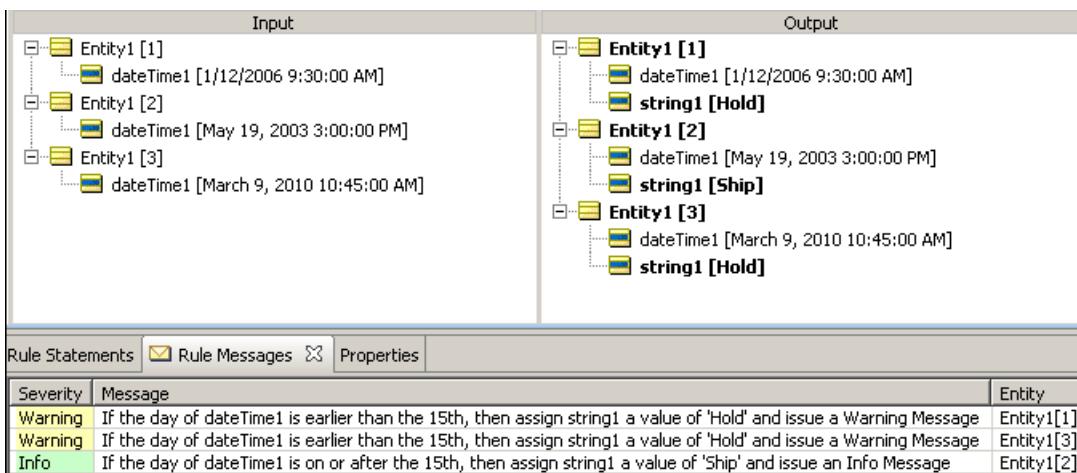
RULESHEET EXAMPLE

The following Rulesheet uses `.day` to assign a value to `string1` and post a message.



SAMPLE RULETEST

A sample Ruletest provides `dateTime1` values for three examples. Input and Output panels are shown below. Posted messages are not shown.



Days between

SYNTAX

```
<DateTime1>.daysBetween(<DateTime2>)
<Date1>.daysBetween(<Date2>)
```

DESCRIPTION

Returns the Integer number of days between DateTimes or Dates. This function calculates the number of milliseconds between the date values and divides that number by 86,400,000 (the number of milliseconds in a day). Any fraction is truncated, leaving an Integer result. If the two dates differ by less than a full 24-hour period, the value returned is zero. A positive Integer value is returned when `<DateTime2>` occurs after `<DateTime1>`.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions

RULESHEET EXAMPLE

The following Rulesheet uses `.daysBetween` to determine the number of days that have elapsed between `dateTime1` and `dateTime2`, compare it to the values in the Condition cells, and assign a value to `string1`.

Conditions	0	1	2
a Entity1.dateTime1.daysBetween(Entity1.dateTime2)		<=30	>30
b			

Actions		
Post Message(s)		
A Entity1.string1	'Not Overdue'	'Overdue'

Overrides

Rule Statements Rule Messages Properties

Ref	ID	Post	Alias	Text
A1				If 30 days or fewer have elapsed between dateTime1 and dateTime2, then Entity1 is not overdue
A2				If more than 30 days have elapsed between dateTime1 and dateTime2, then Entity 1 is overdue

SAMPLE RULETEST

A sample Ruletest provides `dateTime1` and `dateTime2` for two examples. Input and Output panels are shown below.

Day of week

SYNTAX

`<DateTime>.dayOfWeek`

`<Date>.dayOfWeek`

DESCRIPTION

Returns an Integer between 1 and 7, corresponding to the table below:

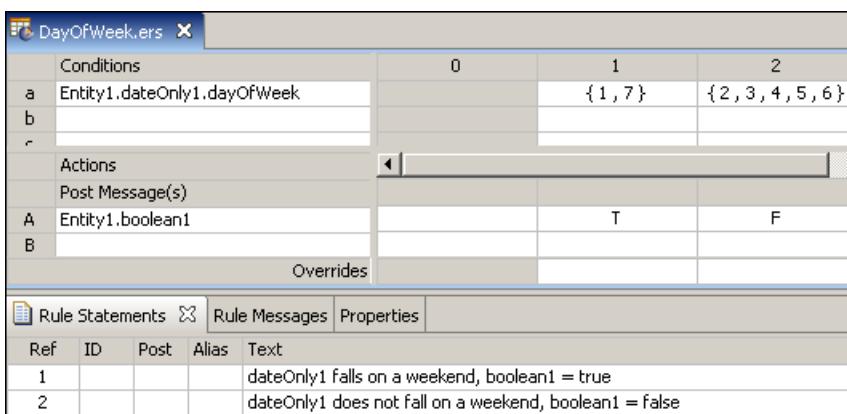
returned Integer	day of the week
1	Sunday
2	Monday
3	Tuesday
4	Wednesday
5	Thursday
6	Friday
7	Saturday

USAGE RESTRICTIONS

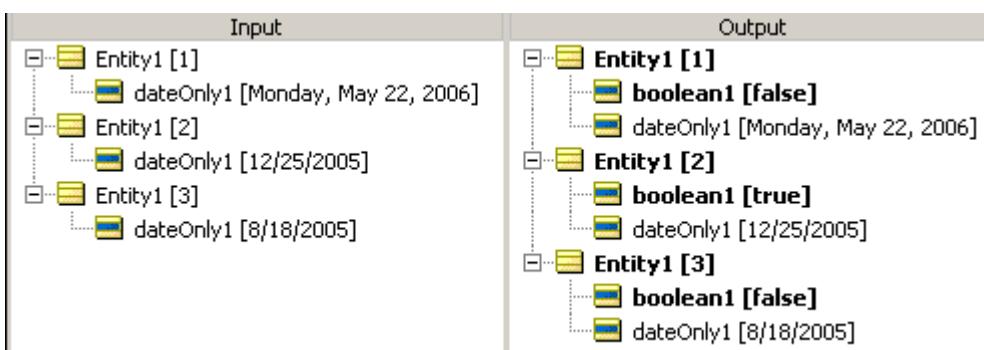
The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions

RULESHEET EXAMPLE

The following Rulesheet uses `.dayOfWeek` to assign a value to `boolean1`.



SAMPLE RULETEST



Day of year

SYNTAX

<DateTime>.dayOfYear

<Date>.dayOfYear

DESCRIPTION

Returns an Integer from 1 to 366, equal to the day number within the year.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions

RULESHEET EXAMPLE

The following Rulesheet uses .dayOfYear to assign a value to string1.

	Conditions	0	1	2
a	Entity1.dateOnly1.dayOfYear		<183	≥ 183
b				

	Actions	
	Post Message(s)	
A	Entity1.string1	'1st Half'
B		'2nd Half'

Overrides				
Rule Statements	Rule Messages	Properties		
Ref	ID	Post	Alias	Text
A1				dateOnly1 falls in the first half of the year
A2				dateOnly1 falls in the second half of the year

SAMPLE RULETEST

Input		Output	
Entity1 [1]	dateOnly1 [Monday, May 22, 2006]	Entity1 [1]	dateOnly1 [Monday, May 22, 2006] string1 [1st Half]
Entity1 [2]	dateOnly1 [12/25/2005]	Entity1 [2]	dateOnly1 [12/25/2005] string1 [2nd Half]
Entity1 [3]	dateOnly1 [8/18/2005]	Entity1 [3]	dateOnly1 [8/18/2005] string1 [2nd Half]

Decrement

SYNTAX

<Number1> -= <Number2>

DESCRIPTION

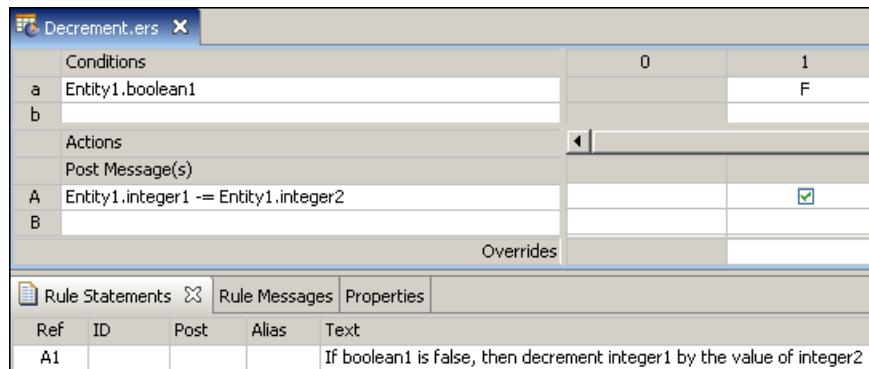
Decrements <Number1> by the value of <Number2>. The data type of <Number1> must accommodate the subtraction of <Number2>. In other words, an Integer may not be decremented by a Decimal without using another operator (such as [.toInteger](#) or [Floor](#) on page 94) to first convert the Decimal to an Integer.

USAGE RESTRICTIONS

The Operators row of the table in [Vocabulary usage restrictions](#) does not apply. Special exceptions: **decrement** may only be used in Action Rows (section 5 in [Sections of Rulesheet that correlate with usage restrictions](#)).

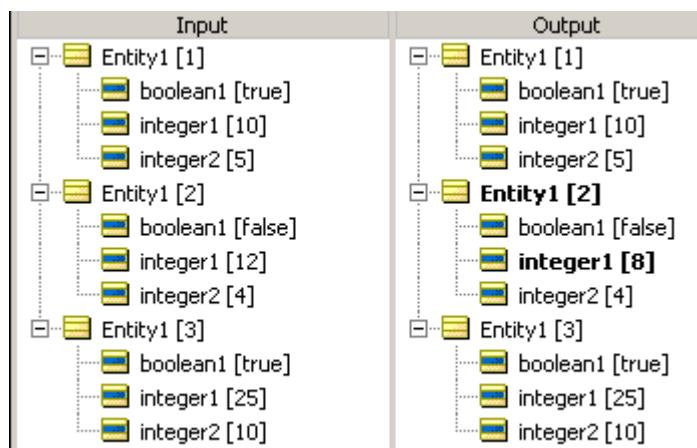
RULESHEET EXAMPLE

This sample Rulesheet uses **decrement** to reduce `integer1` by the value of `integer2` when `boolean1` is false.



SAMPLE RULETEST

A sample Ruletest provides three examples of `integer1`, `integer2`, and `boolean1`. Input and Output panels are shown below.



Disassociate element(s)

SYNTAX

<Collection1> -= <Collection2>

DESCRIPTION

Disassociates all elements of <Collection2> from <Collection1>. Elements are not deleted, but once disassociated from <Collection1>, they are moved to the root level of the data. <Collection1> must be expressed as a unique alias. Contrast this behavior with [remove](#), which deletes elements entirely.

USAGE RESTRICTIONS

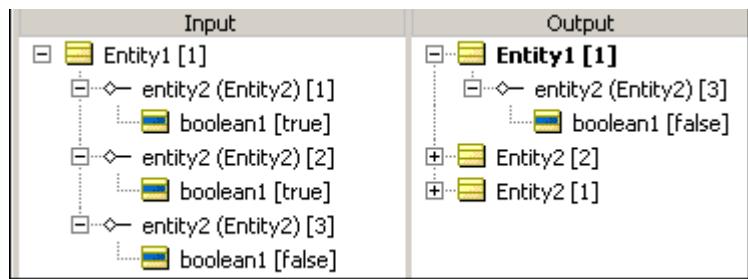
The Operators row of the table in [Vocabulary usage restrictions](#) does not apply. Special exceptions: **disassociate element** may only be used in Action Rows (section 5 in [Sections of Rulesheet that correlate with usage restrictions](#)).

RULESHEET EXAMPLE

This sample Rulesheet removes those elements from collection1 whose boolean1 value is true.

SAMPLE RULETEST

A sample Ruletest provides a collection with three elements. The illustration shows Input and Output panels:



Divide

SYNTAX

<Number1>/<Number2>

DESCRIPTION

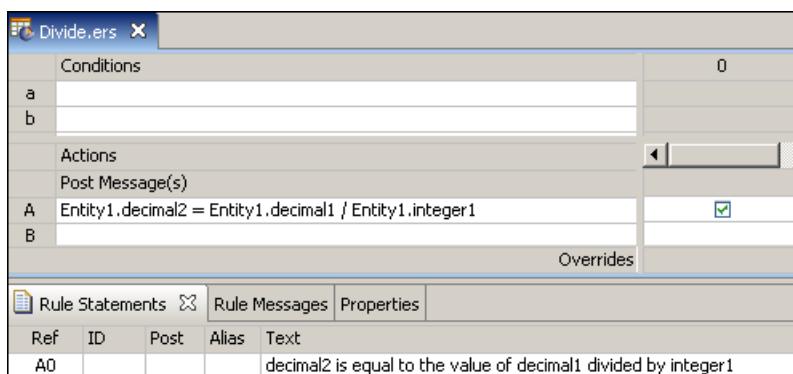
Divides <Number1> by <Number2>. The resulting data type is the more expansive of those of <Number1> and <Number2>.

USAGE RESTRICTIONS

The Operators row in the table of [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

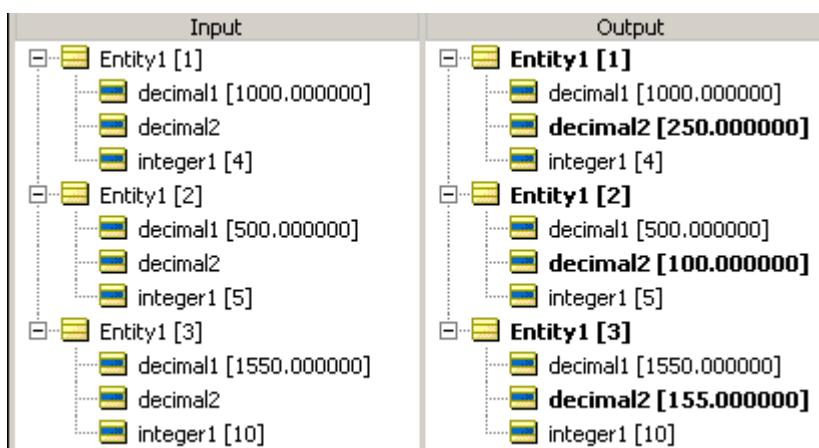
RULESHEET EXAMPLE

This sample Rulesheet uses **divide** to divide decimal1 by integer1 and assign the resulting value to decimal2



SAMPLE RULETEST

A sample Ruletest provides decimal1 and integer1 values for three examples. Input and Output panels are shown below.



Div

SYNTAX

<Integer1>.div(<Integer2>)

DESCRIPTION

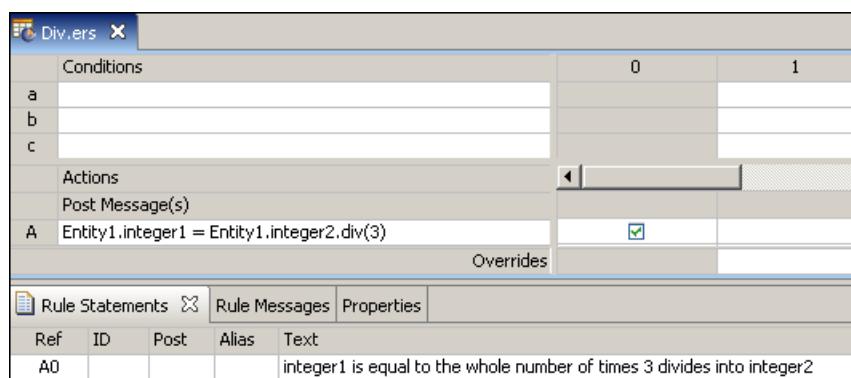
Returns an Integer equal to the whole number of times that <Integer2> divides into <Integer1>. Any remainder is discarded.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

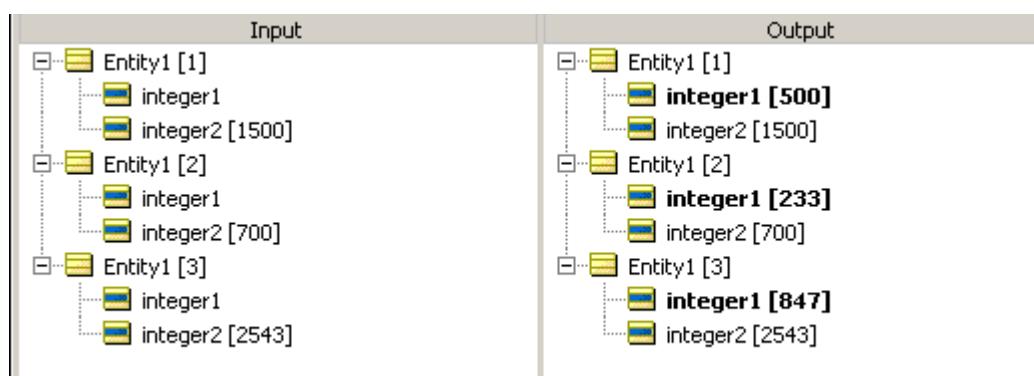
RULESHEET EXAMPLE

This sample Rulesheet uses .div to calculate the whole number of times 3 divides into integer2, and assigns the resulting value to integer1 .



SAMPLE RULETEST

A sample Ruletest provides integer2 values for three examples. Input and Output panels are shown below.



Ends with

SYNTAX

`<String1>.endsWith(<String2>)`

DESCRIPTION

Evaluates `<String1>` and returns a value of true if it ends with the characters specified in `<String2>`.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses `.endsWith` to evaluate whether `string1` ends with the characters `'ville'` and assigns a different value to `string2` for each outcome.

The screenshot shows the Progress Corticon Rulesheet editor interface. The main area displays a table with three rows (a, b, c) and four columns (Conditions, 0, 1, 2). Row 'a' contains the condition `Entity1.string1.endsWith('ville')`, with values T and F in columns 1 and 2 respectively. Row 'b' and 'c' have empty conditions and rows. Below the table is an 'Actions' section with a 'Post Message(s)' button. Under 'Actions', there is a row for 'A' with condition `Entity1.string2` and values "'Small'" and "'Big'" in columns 1 and 2. At the bottom, there is a 'Overrides' section and a 'Rule Statements' tab containing two entries:

Ref	ID	Post	Alias	Text
1				If string1 ends with 'ville' then Entity1 is a small town
2				If string1 does not end with 'ville' then Entity1 is a big town

SAMPLE RULETEST

A sample Ruletest provides `string1` values for three examples. Input and Output panels are shown below.

The screenshot shows the Progress Corticon Ruletest editor interface. The left panel, titled 'Input', shows three entities: Entity1 [1], Entity1 [2], and Entity1 [3]. Entity1 [1] has `string1 [Strongsville]` and `string2`. Entity1 [2] has `string1 [New York]` and `string2`. Entity1 [3] has `string1 [Amityville]` and `string2`. The right panel, titled 'Output', shows three corresponding entities: Entity1 [1], Entity1 [2], and Entity1 [3]. Entity1 [1] has `string1 [Strongsville]` and `string2 [Small]`. Entity1 [2] has `string1 [New York]` and `string2 [Big]`. Entity1 [3] has `string1 [Amityville]` and `string2 [Small]`.

Equals when used as an assignment

SYNTAX

Boolean	<code><Boolean1> = <Expression1></code>
DateTime*	<code><DateTime1> = <DateTime2></code>
Number	<code><Number1> = <Number2></code>
String	<code><String1> = <String2></code>

DESCRIPTION

Boolean	Assigns the truth value of <code><Expression1></code> to <code><Boolean1></code> .
DateTime*	Assigns the value of <code><DateTime2></code> to <code><DateTime1></code> .
Number	Assigns the value of <code><Number2></code> to <code><Number1></code> . Automatic <i>casting</i> (the process of changing a value's data type) -- which includes DateTime, Date, or Time data types -- will occur when assigning an Integer data type to a Decimal data type. To assign a Decimal value to an Integer value, use the <code>.toInteger</code> operator.
String	Assigns the value of <code><String2></code> to <code><String1></code> .

USAGE RESTRICTIONS

The Operators row of the table in [Vocabulary usage restrictions](#) does not apply. Special exceptions: **equals** used as an assignment may only be used in Action Rows (section 5 in [Sections of Rulesheet that correlate with usage restrictions](#)).

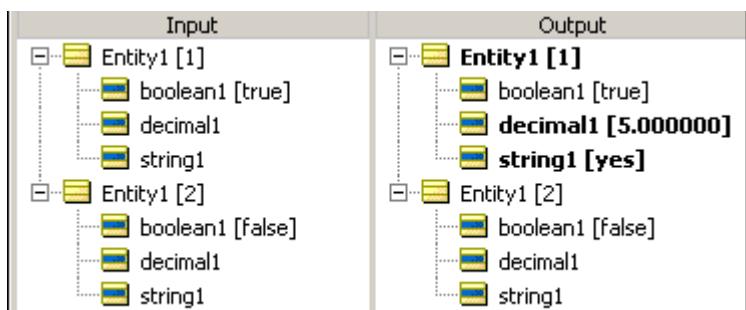
RULESHEET EXAMPLE

The following Rulesheet uses **equals** twice: in an Action row to assign a value to `decimal1`, and in an Action row to assign a value to `string1` based on the value of `boolean1`.

The screenshot shows the Progress Corticon Rulesheet editor interface. At the top, there is a toolbar with icons for file operations. Below the toolbar is a table with two rows. The first row has columns for 'Conditions' and 'Actions'. The second row has columns for 'Actions' and 'Overrides'. Under 'Conditions', row 'a' contains 'Entity1.boolean1'. Under 'Actions', row 'A' contains 'Entity1.decimal1 = 5.0' and row 'B' contains 'Entity1.string1 = 'yes''. In the 'Overrides' section, both rows have checkboxes checked. At the bottom of the editor, there are tabs for 'Rule Statements', 'Rule Messages', and 'Properties'. The 'Rule Statements' tab is selected, showing a table with three rows. Row A0 has a note: 'decimal1 is assigned a value of 5'. Row B0 has a note: 'If boolean1 is true, assign the value of [yes] to string1'. Row 2 has a note: 'If boolean1 is false, then take no action'.

SAMPLE RULETEST

A sample Ruletest provides two examples of boolean1. Input and Output panels are shown below:



Equals when used as a comparison

SYNTAX

Boolean	<Expression1> = <Expression2>
DateTime*	<DateTime1> = <DateTime2>
Number	<Number1> = <Number2>
String	<String1> = <String2>

DESCRIPTION

Boolean	Returns a value of true if <Expression1> is the same as <Expression2>.
DateTime*	Returns a value of true if <DateTime1> is the same as <DateTime2>, including both the Date and the Time portions
Number	Returns a value of true if <Number1> is the same as <Number2>. Different numeric data types may be compared in the same expression.
String	Returns a value of true if <String1> is the same as <String2>. Both case and length are examined to determine equality. Corticon Studio uses the ISO character precedence in comparing String values. See Character precedence: Unicode and Java Collator on page 181.

*includes DateTime, Date, and Time data types

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

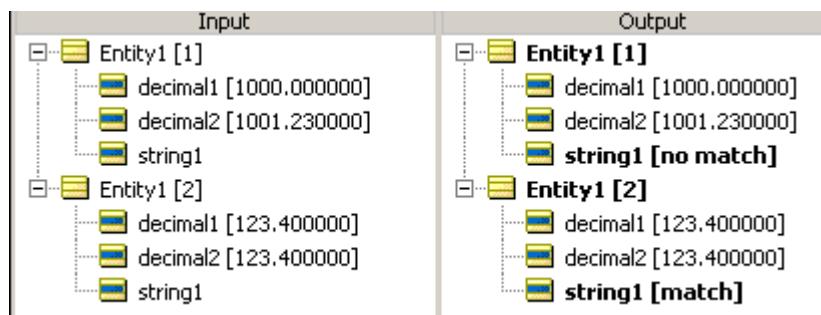
The following Rulesheet uses **equals** to Ruletest whether decimal1 equals decimal2, and assign a value to string1 based on the result of the comparison.

Conditions	0	1	2
a Entity1.decimal1 = Entity1.decimal2		T	F
Actions			
Post Message(s)			
A Entity1.string1		'match'	'no match'
Overrides			

Rule Statements				
Ref	ID	Post	Alias	Text
1				If decimal1 equals decimal2, then assign a value of [match] to string1
2				If decimal1 does not equal decimal2, then assign a value of [no match] to string1

SAMPLE RULETEST

A sample Ruletest provides two examples. Input and Output panels are shown below:



Equals ignoring case

SYNTAX

```
<String1>.equalsIgnoreCase(<String2>)
```

DESCRIPTION

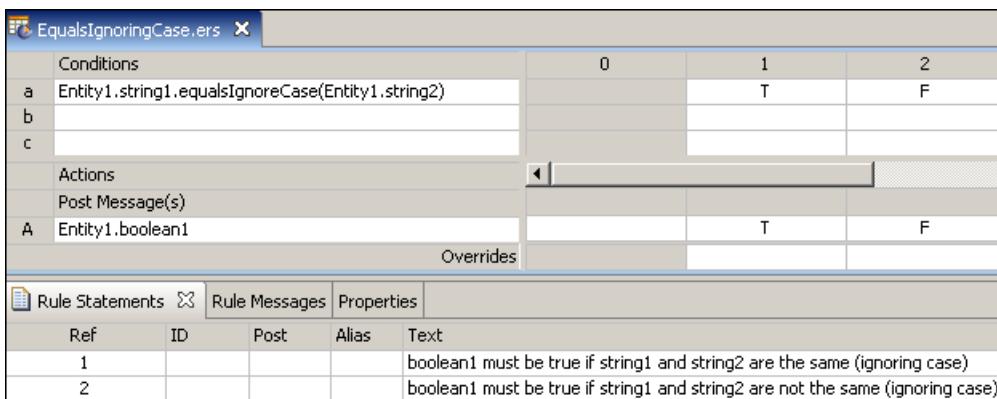
Returns a value of true if `<String1>` is the same as `<String2>`, irrespective of case.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

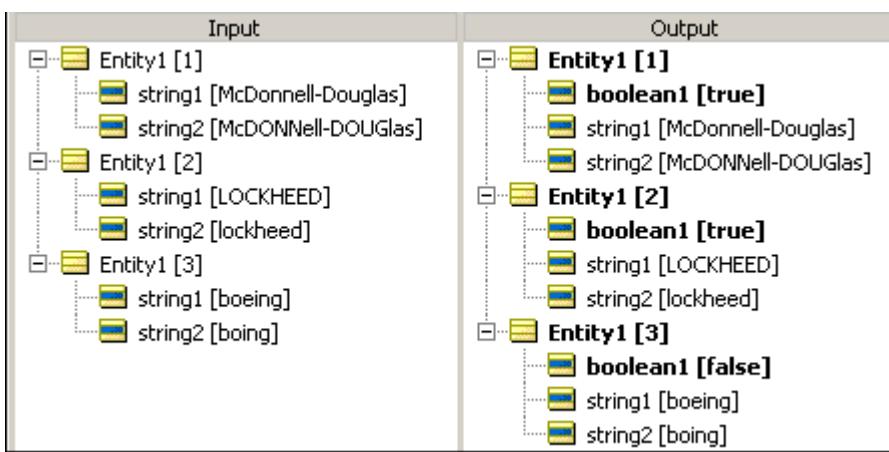
RULESHEET EXAMPLE

This sample Rulesheet uses `.equalsIgnoreCase` to compare the values of `string1` and `string2`, and assign a value to `boolean1` based on the results of the comparison.



SAMPLE RULETEST

A sample Ruletest provides the plane type for three sets of `string1` and `string2`. Input and Output panels are shown below. Notice how these results differ from those shown in the [equals](#) example.



Equals when using Strings

SYNTAX

`<String1>.equals(<String2>)`

DESCRIPTION

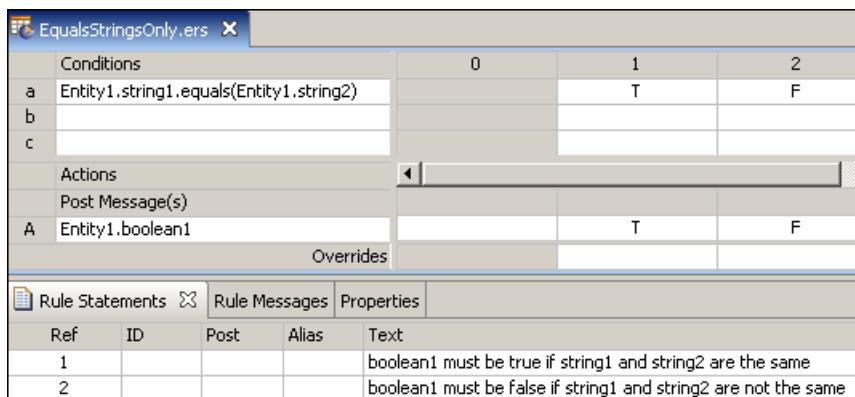
Returns a value of true if `<String1>` is exactly the same as `<String2>`, including character case. This is alternative syntax to [equals](#) (used as a comparison).

USAGE RESTRICTIONS

The Operators row in the table [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

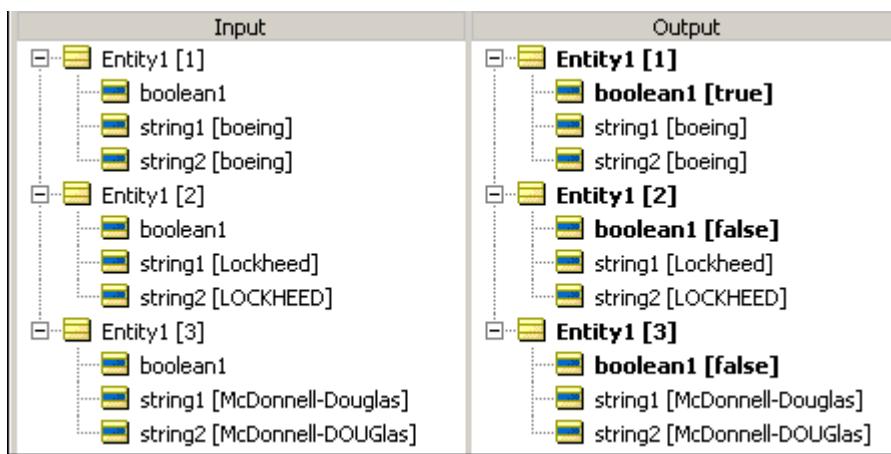
RULESHEET EXAMPLE

This sample Rulesheet uses `.equals` to compare the contents of `string1` and `string2`, and assign a value to `boolean1` as a result.



SAMPLE RULETEST

A sample Ruletest provides three sets of `string1` and `string2`. Input and Output panels are shown below. Notice how these results differ from those shown in the [.equalsIgnoreCase](#) example.



Exists

SYNTAX

```
<Collection> ->exists(<Expression1>,<Expression2>,...)
<Collection> ->exists(<Expression1> or <Expression2> or ...)
```

DESCRIPTION

Returns a value of true if `<Expression>` holds true for *at least one* element of `<Collection>`. `<Collection>` must be expressed as a unique alias. Multiple `<Expressions>` are optional, but at least one is required.

Both **AND** (indicated by commas between `<Expressions>`) and **OR** syntax (indicated by `or` between `<Expressions>`) are supported within the parentheses `(...)`. However, take care to ensure invariant expressions are not inadvertently created. For example:

```
<Collection> -> exists(integer1=5, integer1=8)
```

will always evaluate to `false` because no `integer1` value can be both 5 **AND** 8 simultaneously.

USAGE RESTRICTIONS

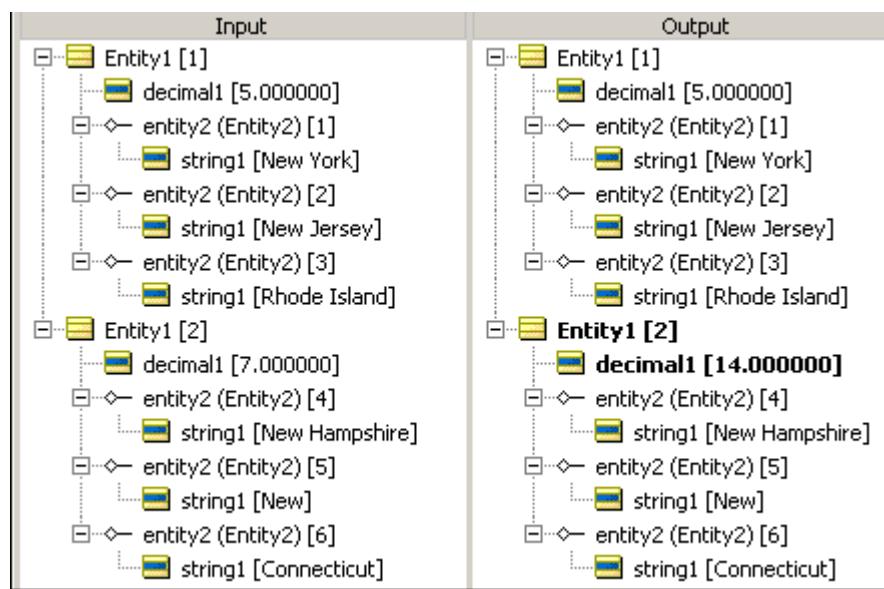
The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

This sample Rulesheet uses `->exists` to check for the existence of an element in `collection1` whose `string1` value equals `New`, and assigns a value to `decimal1` based on the results of the test. Note the use of unique alias `collection1` to represent the collection of `Entity2` associated with `Entity1`.

SAMPLE RULETEST

A sample Ruletest provides 2 separate collections of `Entity2` elements and `Entity1.decimal1` values. Input and Output panels are shown below.



Exponent

SYNTAX

`<Number1> ** <Number2>`

DESCRIPTION

Raises `<Number1>` by the power of `<Number2>`. The resulting data type is the more expansive of those of `<Number1>` and `<Number2>`. To find a root, `<Number2>` can be expressed as a decimal value, such as 0.5 for a square root, or -- for greater accuracy in larger roots -- in decimal format within parentheses, such as `** (1.0/3.0)` for a cube root.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

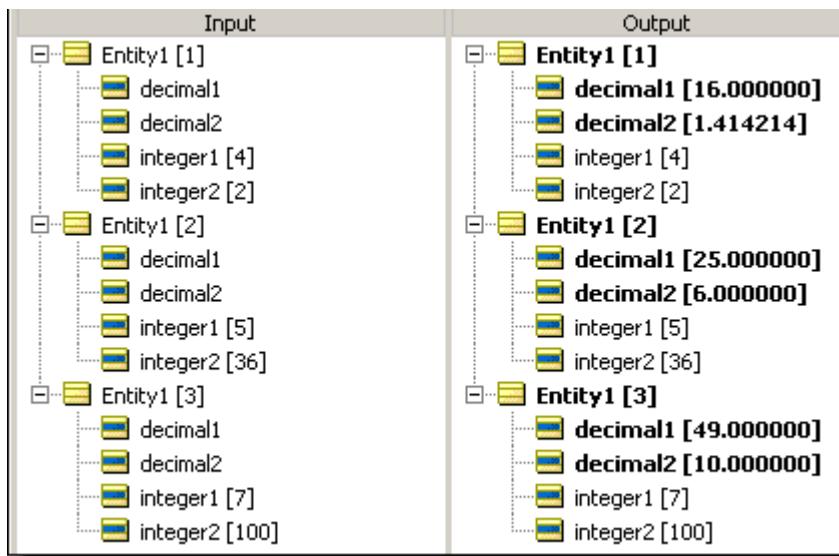
RULESHEET EXAMPLE

This sample Rulesheet uses **exponent** to raise `integer1` and `integer2` by the power of 2 and 0.5, respectively, and assign the resulting value to `decimal1` and `decimal2`, respectively.

Ref	ID	Post	Alias	Text
A0				decimal1 is equal to the square of integer1
B0				decimal2 is equal to the square root of integer2

SAMPLE RULETEST

A sample Ruletest provides `decimal1` and `integer1` values for three examples.



False

SYNTAX

false or F

DESCRIPTION

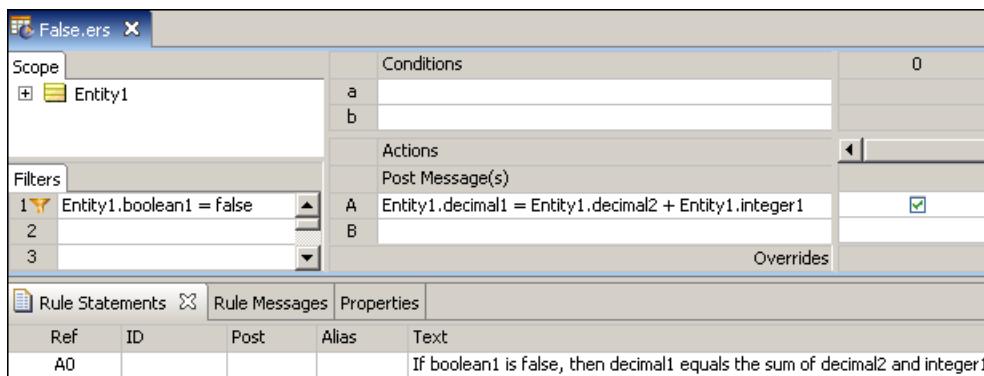
Represents the Boolean value false. Recall from discussion of [truth values](#) that an <expression> is evaluated for its truth value, so the expression Entity1.boolean1=false evaluates to true only when boolean1=false. But since boolean1 is Boolean and has a truth value all by itself without any additional syntax, we could simply state not Entity1.boolean1, with the same effect. Many examples in the documentation use explicit syntax like boolean1=true or boolean2=false for clarity and consistency, even though boolean1 or not boolean2 are equivalent, respectively, to the explicit syntax.

USAGE RESTRICTIONS

The Operators row of the table in [Vocabulary usage restrictions](#) applies. No special exceptions.

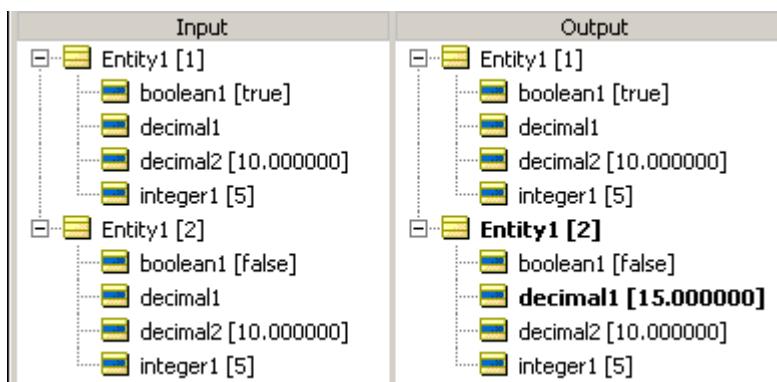
RULESHEET EXAMPLE

The following Rulesheet uses **false** in a Filter row to test whether boolean1 is false, and perform the Nonconditional computation if it is. As discussed above, the alternative expression not Entity1.boolean1 is logically equivalent.



SAMPLE RULETEST

A sample Ruletest provides three examples. Assume `decimal2=10.0` and `integer1=5` for all examples. Input and Output panels are shown below:



First

SYNTAX

`<Sequence> ->first.<attribute1>`

DESCRIPTION

Returns the value of `<attribute1>` of the first element in `<Sequence>`. Another operator, such as `->sortedBy`, must be used to transform a `<Collection>` into a `<Sequence>` before `->first` may be used. `<Sequence>` must be expressed as a unique alias. See [Advanced Collection Syntax](#) for more examples of usage.

`<attribute1>` may be of any data type.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

This sample Rulesheet uses `->first` to identify the first element of the sequence created by applying `->sortedBy` to `collection1`. Once identified, the value of the `string1` attribute belonging to this first element is evaluated. If the value of `string1` is `Joe`, then `boolean1` attribute of `Entity1` is assigned the value of `true`.

The screenshot shows the Progress Corticon Studio interface. The top half displays the Rule Editor panel, which includes a Scope tree on the left showing entities Entity_1 and Entity_2, and a Conditions table on the right. The Conditions table has columns for Conditions (a through e) and Actions (A through D). The bottom half shows the Rule Test panel with two tabs: Rule Statements and Rule Messages. The Rule Statements tab contains two entries, each with a Ref, ID, Post, and Text column.

Ref	ID	Post	A...	Text
1				If the string1 value of the first element in collection1, in ascending order by decimal1, is equal to Joe, then boolean1 is true.
2				If the string1 value of the 2nd element in collection1, in ascending order by decimal1, is NOT equal to Joe, then boolean1 is false.

SAMPLE RULETEST

A sample Ruletest provides a collection of three elements, each with a decimal1 value. Input and Output panels are shown below.

The screenshot shows the Rule Test panel for the rule /Examples/first.ers. It consists of two main sections: Input and Output. The Input section shows a hierarchical tree of entities Entity_1 [1] and Entity_2 [Entity_2] [1, 2, 3], each with attributes boolean1, decimal1, and string1. The Output section shows the resulting collection of Entity_1 [1] entities, each with boolean1 set to true and the same decimal1 and string1 values as their corresponding input entities.

First (number)

SYNTAX

<Sequence> ->first (integer)

DESCRIPTION

Returns a `->subSequence` of the first *integer* entities in the collection `<Sequence>`. Another operator, such as `->sortedBy` or `->sortedByDesc`, must be used to transform a `<Collection>` into a `<Sequence>` before `->first` can be used. `<Sequence>` must be expressed as a unique alias. If *integer* is larger than the number of entities in the collection, all the entities in the collection are returned. See [Advanced Collection Syntax](#) for more examples of usage.

USAGE RESTRICTIONS

The Operators row of the table in [Vocabulary usage restrictions](#) does not apply. Special exceptions: `last(x)` may only be used in Action Rows (section 5 in [Sections of Rulesheet that correlate with usage restrictions](#)).

RULESHEET EXAMPLE

This sample Rulesheet uses `->first(2)` to select the first two elements of the sequence created by applying `->sortedBy` to `collection2`. Once identified, the first 2 entities will be returned as the sequence `collection3`.

Scope	Conditions	
	a	0
	b	
	Actions	
Entity_1	Post Message(s)	
Entity_2 (Entity_2) [collection2]	A collection3=collection2->sortedBy(collection2.decimal1)->first(2)	<input checked="" type="checkbox"/>
Entity_3	B	
Entity_2 (Entity_2) [collection3]		

SAMPLE RULETEST

A sample Ruletest provides a collection of five elements, each with a `decimal1` value. Input and Output panels are shown below.

Input	Output
<ul style="list-style-type: none"> Entity_1 [1] <ul style="list-style-type: none"> entity_2 (Entity_2) [1] <ul style="list-style-type: none"> decimal1 [500.000000] entity_2 (Entity_2) [2] <ul style="list-style-type: none"> decimal1 [800.000000] entity_2 (Entity_2) [3] <ul style="list-style-type: none"> decimal1 [700.000000] entity_2 (Entity_2) [4] <ul style="list-style-type: none"> decimal1 [100.000000] entity_2 (Entity_2) [5] <ul style="list-style-type: none"> decimal1 [600.000000] Entity_3 [1] 	<ul style="list-style-type: none"> Entity_1 [1] <ul style="list-style-type: none"> entity_2 (Entity_2) [1] <ul style="list-style-type: none"> decimal1 [500.000000] entity_2 (Entity_2) [2] <ul style="list-style-type: none"> decimal1 [800.000000] entity_2 (Entity_2) [3] <ul style="list-style-type: none"> decimal1 [700.000000] entity_2 (Entity_2) [4] <ul style="list-style-type: none"> decimal1 [100.000000] entity_2 (Entity_2) [5] <ul style="list-style-type: none"> decimal1 [600.000000] Entity_3 [1] <ul style="list-style-type: none"> entity_2 (Entity_2) [1] entity_2 (Entity_2) [4]

Note: The selected entities and their values are highlighted to improve readability.

RULESHEET EXAMPLE: USING DESCENDING SORT

Sometimes it is easier to understand this type of action when you sort the data in descending order; when thinking of the "the top three sales figures", the first three largest values are what is intended. In this example, the action uses `->sortedByDesc` to order the collection largest-to-smallest and then moves the top 2 entities to the result sequence:

Scope	Conditions	Actions
a		0
b		
	Actions	<
	Post Message(s)	
A	collection3=collection2->sortedByDesc(collection2.decimal1)->first(2)	<input checked="" type="checkbox"/>
B		

SAMPLE RULETEST: USING DESCENDING SORT

The sample Ruletest shows the two entities with the highest values are copied to the results sequence:

Input	Output
<ul style="list-style-type: none"> Entity_1 [1] <ul style="list-style-type: none"> entity_2 (Entity_2) [1] <ul style="list-style-type: none"> decimal1 [500.000000] entity_2 (Entity_2) [2] <ul style="list-style-type: none"> decimal1 [800.000000] entity_2 (Entity_2) [3] <ul style="list-style-type: none"> decimal1 [700.000000] entity_2 (Entity_2) [4] <ul style="list-style-type: none"> decimal1 [100.000000] entity_2 (Entity_2) [5] <ul style="list-style-type: none"> decimal1 [600.000000] Entity_3 [1] 	<ul style="list-style-type: none"> Entity_1 [1] <ul style="list-style-type: none"> entity_2 (Entity_2) [1] <ul style="list-style-type: none"> decimal1 [500.000000] entity_2 (Entity_2) [2] <ul style="list-style-type: none"> decimal1 [800.000000] entity_2 (Entity_2) [3] <ul style="list-style-type: none"> decimal1 [700.000000] entity_2 (Entity_2) [4] <ul style="list-style-type: none"> decimal1 [100.000000] entity_2 (Entity_2) [5] <ul style="list-style-type: none"> decimal1 [600.000000] Entity_3 [1] <ul style="list-style-type: none"> entity_2 (Entity_2) [2] entity_2 (Entity_2) [3]

Note: The selected entities and their values are highlighted to improve readability.

Floor

SYNTAX

`<Decimal>.floor`

DESCRIPTION

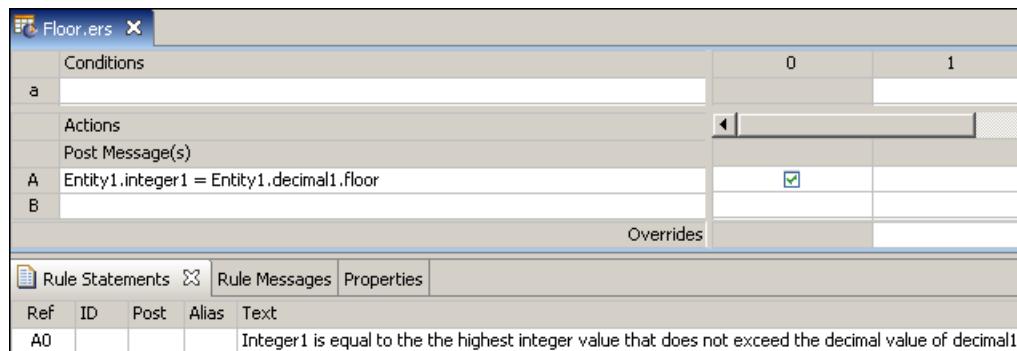
Returns the Integer closest to zero from `<Decimal>`. `.floor` may also be thought of as a truncation of `<Decimal>`.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

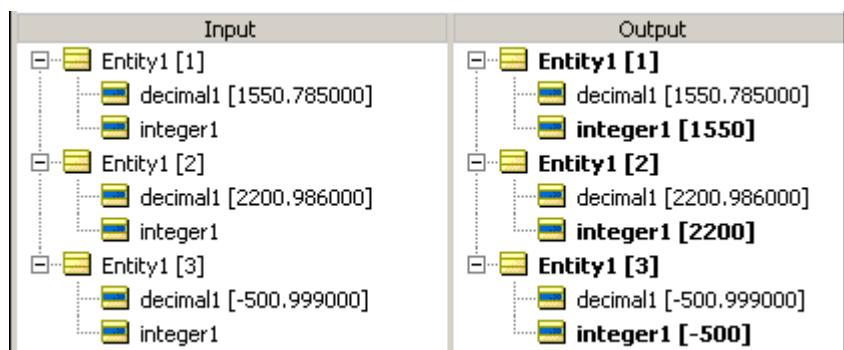
The Rulesheet uses **.floor** to assign Integer values to `integer1` that are closer to zero than the input `decimal1` values.



SAMPLE RULETEST

A sample Ruletest provides three `decimal1` values. Input and Output panels are shown below:

Note: Notice how these results differ from those shown in the [Round](#) example.



For all

SYNTAX

```
<Collection> ->forAll(<Expression1>, <Expression2>,...)
<Collection> ->forAll(<Expression1> or <Expression2> or ...)
```

DESCRIPTION

Returns a value of true if **every** `<Expression>` holds true for **every** element of `<Collection>`. `<Collection>` must be expressed as a unique alias. Multiple `<Expressions>` are optional, but at least one is required.

Both **AND** (indicated by commas between <Expressions>) and **OR** syntax (indicated by **or** between <Expressions>) is supported within the parentheses (...). However, take care to ensure invariant expressions are not inadvertently created. For example:

```
<Collection> -> forAll(integer1=5, integer1=8)
```

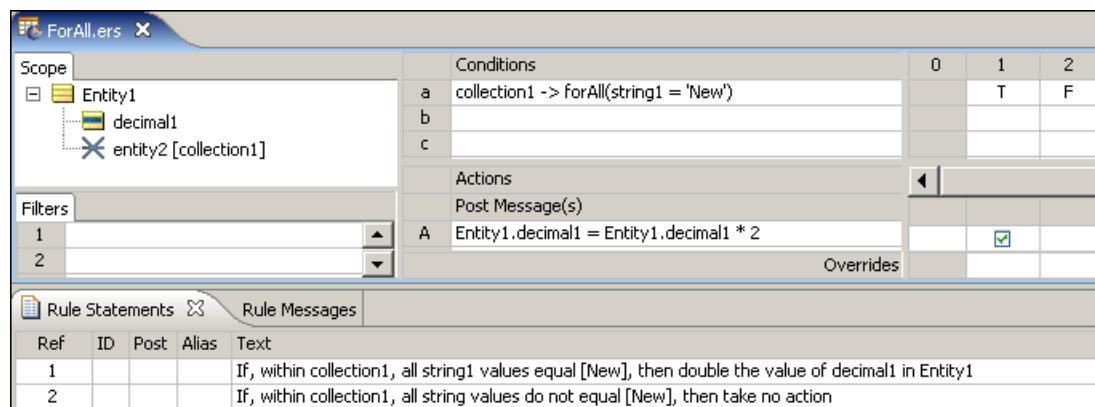
will always evaluate to `false` because no single `integer1` value can be both 5 **AND** 8 simultaneously, let alone all of them.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

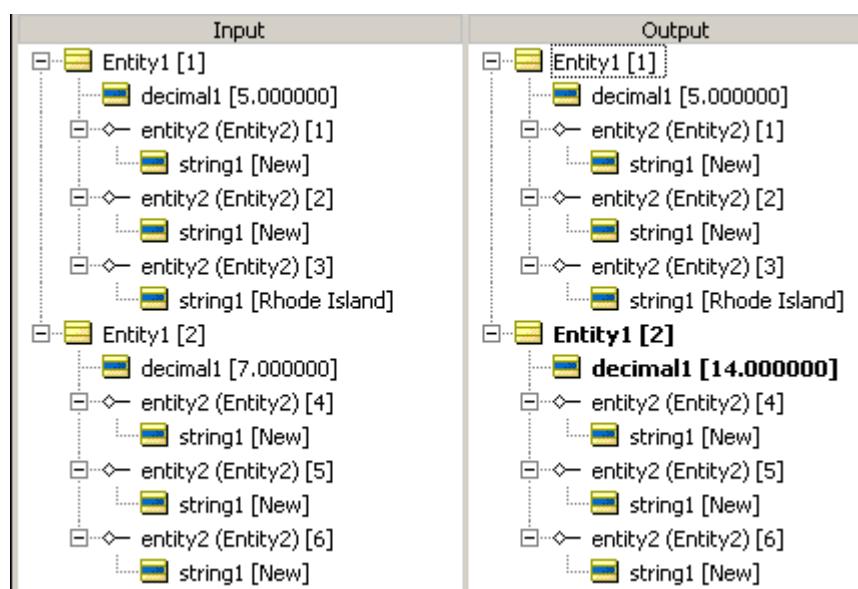
RULESHEET EXAMPLE

This sample Rulesheet uses `->forAll` to check for the existence of an element in `collection1` whose `string1` value equals `New`, and assigns a value to `decimal1` based on the results of the test. Note the use of unique alias `collection1` to represent the collection of `Entity2` associated with `Entity1`.



SAMPLE RULETEST

A sample Ruletest provides 2 separate collections of `Entity2` elements and `Entity1.decimal1` values. The following illustration shows Input and Output panel



Greater than

SYNTAX

DateTime*	<code><DateTime1> > <DateTime2></code>
Number	<code><Number1> > <Number2></code>
String	<code><String1> > <String2></code>

DESCRIPTION

DateTime*	Returns a value of true if <code><DateTime1></code> is greater than or equal to <code><DateTime2></code> . This is equivalent to <code><DateTime1></code> occurring “after” <code><DateTime2></code>
Number	Returns a value of true if <code><Number1></code> is greater than <code><Number2></code> . Different numeric data types may be compared in the same expression.
String	Returns a value of true if <code><String1></code> is greater than <code><String2></code> . Studio uses Character precedence: Unicode and Java Collator on page 181 to determine character precedence.

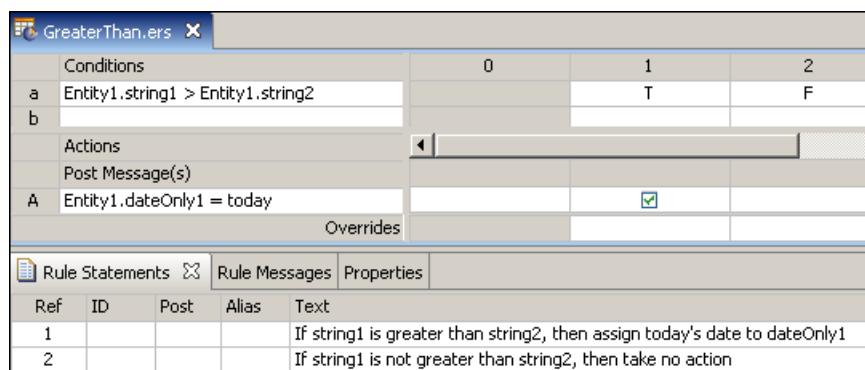
*includes DateTime, Date, and Time data types

USAGE RESTRICTIONS

The Operators row of the table in [Vocabulary usage restrictions](#) applies, with the following exception: **greater than** may also be used in Conditional Value Sets & Cells (section 5 in [Sections of Rulesheet that correlate with usage restrictions](#)).

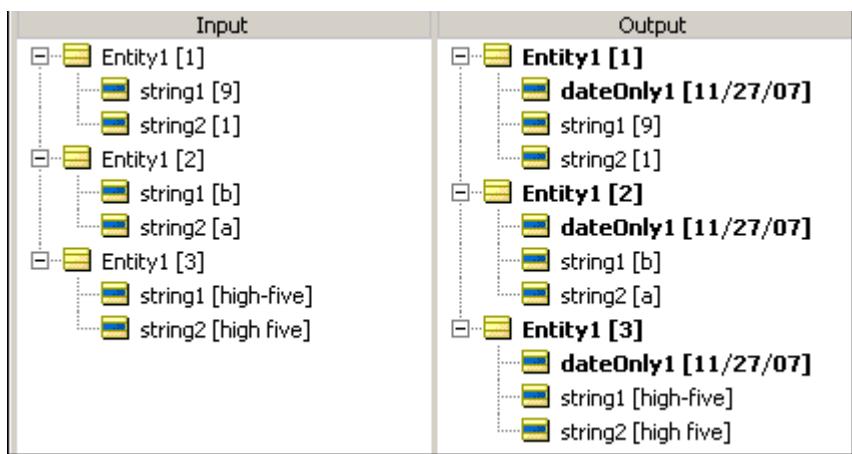
RULESHEET EXAMPLE

The following Rulesheet uses **greater than** to test whether string1 is greater than string2, and assign today's date to dateTime1 if it is. See [today](#) for an explanation of this literal term.



SAMPLE RULETEST

A sample Ruletest provides three examples. Input and Output panels are shown below:



Greater than or equal to

SYNTAX

DateTime*	<code><DateTime1> >= <DateTime2></code>
Number	<code><Number1> >= <Number2></code>
String	<code><String1> >= <String2></code>

DESCRIPTION

DateTime*	Returns a value of true if <code><DateTime1></code> is greater than or equal to <code><DateTime2></code> . This is equivalent to <code><DateTime1></code> occurring on or after <code><DateTime2></code> .
Number	Returns a value of true if <code><Number1></code> is greater than or equal to <code><Number2></code> . Different numeric data types may be compared in the same expression.
String	Returns a value of true if <code><String1></code> is greater than or equal to <code><String2></code> . Corticon Studio uses Character precedence: Unicode and Java Collator on page 181 to determine character precedence.

*includes DateTime, Date, and Time data types

USAGE RESTRICTIONS

The Operators row of the table in [Vocabulary usage restrictions](#) applies, with the following exception: **greater than or equal to** may also be used in Conditional Value Sets & Cells (section 5 in [Sections of Rulesheet that correlate with usage restrictions](#)).

RULESHEET EXAMPLE

The following Rulesheet uses **greater than or equal to** to test whether `string1` is greater than or equal to `string2`, and assign today's date to `dateTime1` if it is. See [today](#) for an explanation of this literal term.

SAMPLE RULETEST

A sample Ruletest provides two examples. Input and Output panels are shown below:

Hour

SYNTAX

`<DateTime>.hour`

`<Time>.hour`

DESCRIPTION

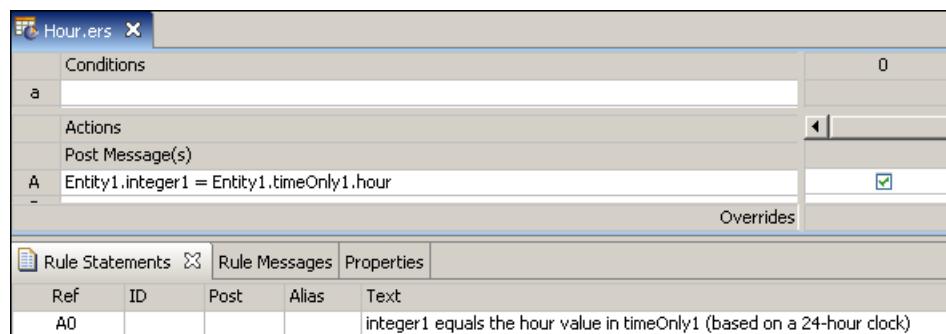
Returns the hour portion of `<DateTime>` or `<Time>`. The returned value is based on a 24-hour clock. For example, 10:00 PM (22:00 hours) is returned as 22.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

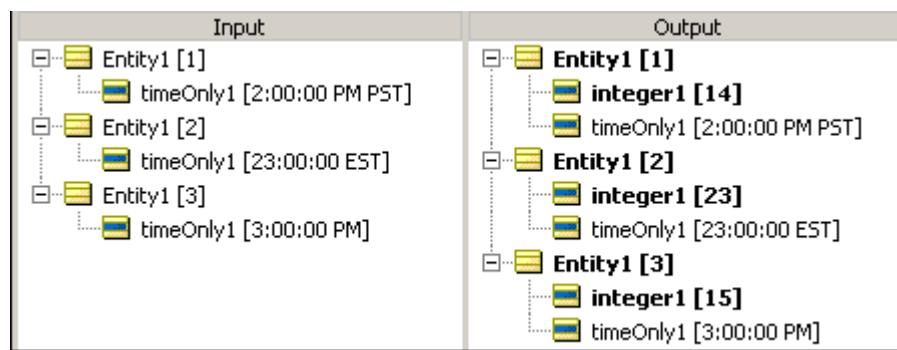
RULESHEET EXAMPLE

The following Rulesheet uses `.hour` to evaluate `dateTime1` and assign the hour value to `integer1`.



SAMPLE RULETEST

A sample Ruletest provides three examples of `dateTime1`. Input and Output panels are shown below. Notice that the hour returned is dependent upon the timezone of the machine executing the rule. The hour returned is independent of the machine running the Ruletest and only depends on the locale/timezone of the data itself.



Hour between

SYNTAX

`<DateTime1>.hoursBetween(<DateTime2>)`

DESCRIPTION

Returns the Integer number of hours between any two DateTimes or Times. The function calculates the number of milliseconds between the two values and divides that number by 3,600,000 (the number of milliseconds in an hour). The decimal portion is then truncated. If the two dates differ by less than a full hour, the value is zero. This function returns a positive number if `<DateTime2>` is later than `<DateTime1>`.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses `.hoursBetween` to determine the number of hours that have elapsed between `dateTime1` and `dateTime2`, compare it to the `Values` set, and assign a value to `string1`.

The screenshot shows the 'HoursBetween.ers' rule editor interface. The 'Conditions' section contains a table with three rows (a, b, c) and three columns (0, 1, 2). Row 'a' has condition 'Entity1.dateTime1.hoursBetween(Entity1.dateTime2)' with values 0, <=24, and >24. The 'Actions' section contains a table with one row 'A' and three columns. Row 'A' has action 'Entity1.string1' with values 'Not Overdue' and 'Overdue'. The 'Overrides' section is empty. Below these are tabs for 'Rule Statements' and 'Rule Messages'. The 'Rule Statements' tab shows two entries: Ref 1 with ID 1 and Text 'If 24 or fewer hours have elapsed between dateTime1 and dateTime2, then Entity1 is not overdue'; Ref 2 with ID 2 and Text 'If more than 24 hours have elapsed between dateTime1 and dateTime2, then Entity1 is not overdue'.

SAMPLE RULETEST

A sample Ruletest provides `dateTime1` and `dateTime2` for two examples. Input and Output panels are shown below.

The screenshot shows the Ruletest interface with two panels: 'Input' and 'Output'. The 'Input' panel displays two entities, Entity1 [1] and Entity1 [2]. Entity1 [1] has dateTime1 [3/10/2006 4:00:00 PM EST] and dateTime2 [3/15/2006 2:30:00 AM EST]. Entity1 [2] has dateTime1 [November 23, 2005 12:30:00 EST] and dateTime2 [November 23, 2005 12:45:00 EST]. The 'Output' panel shows the results of the rule execution. Entity1 [1] has dateTime1 [3/10/2006 4:00:00 PM EST], dateTime2 [3/15/2006 2:30:00 AM EST], and string1 [Overdue]. Entity1 [2] has dateTime1 [November 23, 2005 12:30:00 EST], dateTime2 [November 23, 2005 12:45:00 EST], and string1 [Not Overdue].

In {List}

SYNTAX

Date	<code><Date1> in {<Date2>, <Date3>, ...}</code>
DateTime	<code><DateTime1> in {<DateTime2>, <DateTime3>, ...}</code>
Decimal	<code><Decimal1> in {<Decimal2>, <Decimal3>, ...}</code>
Integer	<code><Integer1> in {<Integer2>, <Integer3>, ...}</code>
String	<code><String1> in {<String2>, <String3>, ...}</code>
Time	<code><Time1> in {<Time2>, <Time3>, ...}</code>

DESCRIPTION

Returns the value `true` if the attribute type is contained in the set of valid values for the attribute.

USAGE RESTRICTIONS

- The set of values is always enclosed in braces: {}
 - For integer and decimal data types, a list of literals or enumerated values without labels requires that the values are not in single quotes, such as {3,1,2}.
 - For date and String data types, a list of literals or enumerated values without labels requires that the values are in single quotes, such as {'B','A','C'}.
 - The list can be in any order.
 - Duplicate values or labels in a list are tolerated.

When enumerated datatypes with labels are used:

- The labels are listed without delimiters, such as {B, A, C}
 - Values and labels can be mixed, such as {A, B, 'C value'}.

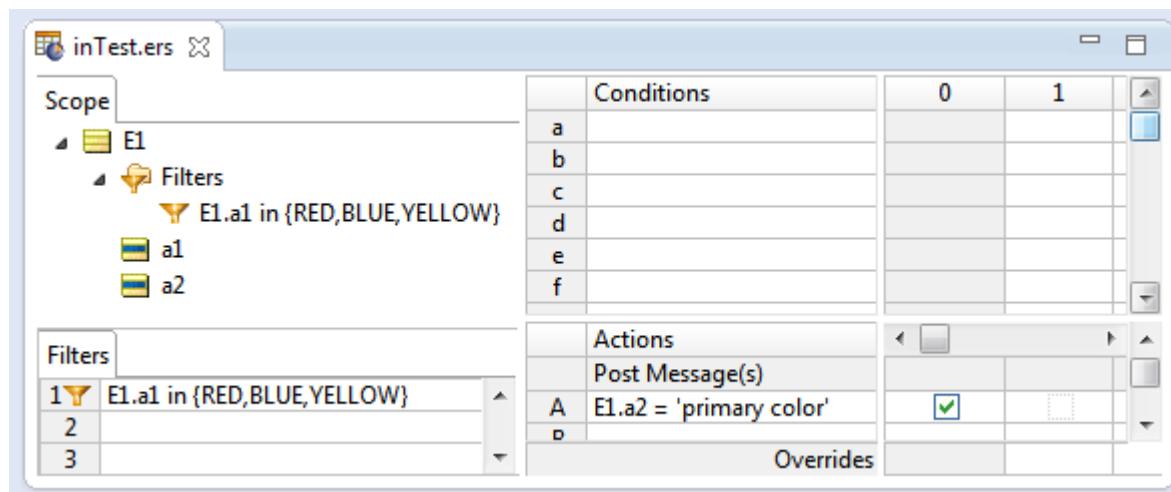
Note: While literal values in the enumeration table are accepted in a list, only existing label values will be exposed and accepted as valid.

The Operators row of the table in [Vocabulary Usage Restriction](#) does not apply. The `in` operator can be used in Conditions and Filters, but not in Actions.

RULESHEET EXAMPLE

The example's Vocabulary defined an enumerated list:

The following Rulesheet uses **in** to filter certain labels to be tested against request data:



SAMPLE TEST

A sample Ruletest provides examples. Input and Output panels are shown below.

Input	Output
E1 [1] a1 [RED]	E1 [1] a1 [RED] a2 [primary color]
E1 [2] a1 [PURPLE]	E1 [2] a1 [PURPLE]
E1 [3] a1 [BLUE]	E1 [3] a1 [BLUE] a2 [primary color]
E1 [4] a1 [ORANGE]	E1 [4] a1 [ORANGE]
E1 [5] a1 [GREEN]	E1 [5] a1 [GREEN]
E1 [6] a1 [YELLOW]	E1 [6] a1 [YELLOW] a2 [primary color]
E1 [7] a1 [WHITE]	E1 [7] a1 [WHITE]
E1 [8] a1 [BLACK]	E1 [8] a1 [BLACK]
E1 [9] a1	E1 [9] a1

In (Range)

SYNTAX

Date	<code><Date1> in (<earlierDate2>..<laterDate3>)</code>
DateTime	<code><DateTime1> in (<earlierDateTime2>..<laterDateTime3>)</code>
Decimal	<code><Decimal1> in (<smallerDecimal2>..<largerDecimal3>)</code>
Integer	<code><Integer1> in (<smallerInteger2>..<largerInteger3>)</code>
String	<code><String1> in (<startString2>..<endString3>)</code>
Time	<code><Time1> in (<earlierTime2>..<laterTime3>)</code>

A square bracket on either end of the expression indicates that the start or end value is to be included in the range.

DESCRIPTION

Returns the value `true` if the attribute type is contained in the range of valid values for the attribute.

USAGE RESTRICTIONS

- For integer and decimal data types, the range of values are not in single quotes. For example, `(1..3)`.
- For date and String data types, the range of values are in single quotes. For example, `('A'..'C')`.

The Operators row of the table in [Vocabulary Usage Restriction](#) does not apply. The `in` operator can be used in Conditions and Filters, but not in Actions.

RULESHEET EXAMPLE

The following Rulesheet uses `in` ranges for three data types OR'ed together in a filter to be tested against request data:

The screenshot shows the Ruletest interface with the rule file `*inRange.ers`. The left pane displays the rule structure:

- Scope:** E1
 - Filters:**
 - E1.date1 in ['1/1/62'..'12/31/83']
 - alert
 - date1
 - integer1
 - string1

The right pane contains two tables:

	Conditions	0	1
a			
b			
c			
d			
e			
f			
g			
h			
i			
j			
k			
.			

	Actions	
	Post Message(s)	
A	E1.alert='eligible'	<input checked="" type="checkbox"/>
B		
C		
D		
	Overrides	

SAMPLE TEST

A sample Ruletest provides examples. Input and Output panels are shown below.

Input	Output
<ul style="list-style-type: none"> E1 [1] <ul style="list-style-type: none"> date1 [2/4/52] integer1 [12] string1 [F] E1 [2] <ul style="list-style-type: none"> string1 [D] E1 [3] <ul style="list-style-type: none"> date1 [3/15/77] E1 [4] <ul style="list-style-type: none"> integer1 [32] E1 [5] <ul style="list-style-type: none"> integer1 [-40] string1 [A] E1 [6] <ul style="list-style-type: none"> date1 [1/1/62] string1 [C] 	<ul style="list-style-type: none"> E1 [1] <ul style="list-style-type: none"> alert [eligible] date1 [2/4/52] integer1 [12] string1 [F] E1 [2] <ul style="list-style-type: none"> string1 [D] E1 [3] <ul style="list-style-type: none"> alert [eligible] date1 [3/15/77] E1 [4] <ul style="list-style-type: none"> integer1 [32] E1 [5] <ul style="list-style-type: none"> integer1 [-40] string1 [A] E1 [6] <ul style="list-style-type: none"> alert [eligible] date1 [1/1/62] string1 [C]

Increment

SYNTAX

<Number1> += <Number2>

DESCRIPTION

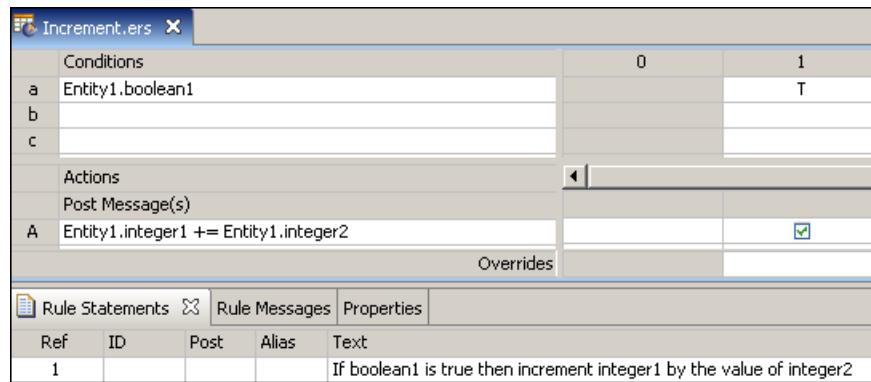
Increments <Number1> by the value of <Number2>. The data type of <Number1> must accommodate the addition of <Number2>. In other words, an Integer may not be incremented by a Decimal without using another operator (such as [.toInteger](#) or [Floor](#) on page 94.[floor](#)) to first convert the Decimal to an Integer.

USAGE RESTRICTIONS

The Operators row of the table in [Vocabulary usage restrictions](#) does not apply. Special exceptions: **increment** may only be used in Action Rows (section 5 in [Sections of Rulesheet that correlate with usage restrictions](#)).

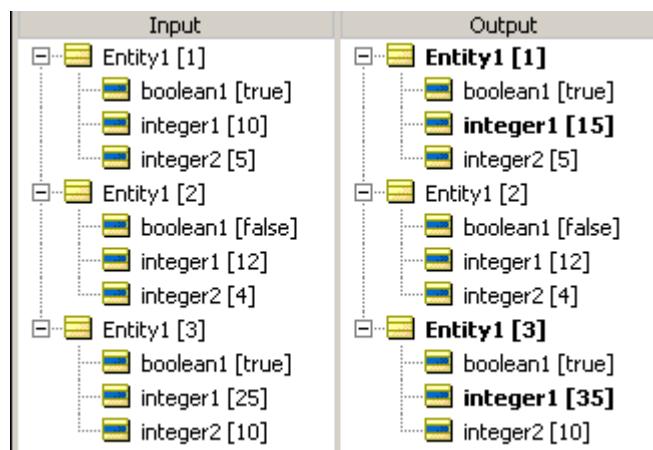
RULESHEET EXAMPLE

This sample Rulesheet uses **increment** to increment `integer1` by the value of `integer2` when `boolean1` is true.



SAMPLE RULETEST

A sample Ruletest provides three examples of `integer1`, `integer2`, and `boolean1`. Input and Output panels are shown below.



Index of

SYNTAX

`<String1>.indexOf(<String2>)`

DESCRIPTION

Determines if `<String2>` is contained within `<String1>` and returns an Integer value equal to the beginning character position of the first occurrence of `<String2>` within `<String1>`. If `<String1>` does not contain `<String2>`, then a value of 0 (zero) is returned. This operator is similar to [.contains](#) but returns different results. A 0 result from `.indexOf` is equivalent to a `false` value returned by the `.contains` operator.

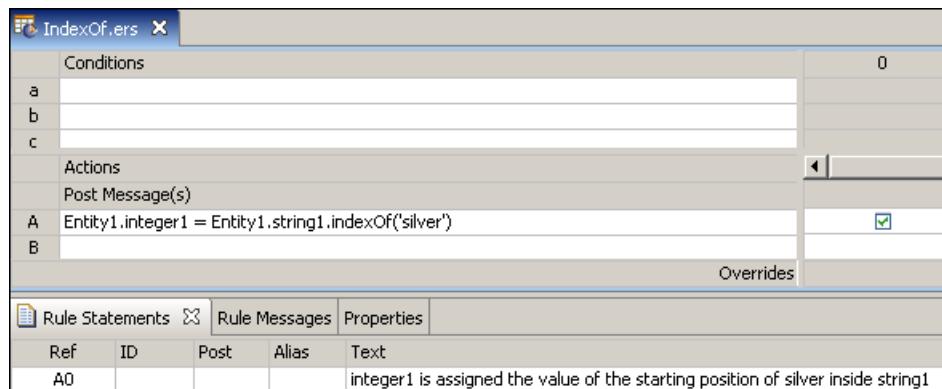
If `<String1>` contains more than one occurrence of `<String2>`, `.indexOf` returns the first character position of the first occurrence. For example: If `<String1>` holds the String value 'Mississippi' and `<String2>` holds the String value 'ss', then the `.indexOf` operator returns 3. The second occurrence of 'ss' beginning at position 6 is not identified.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

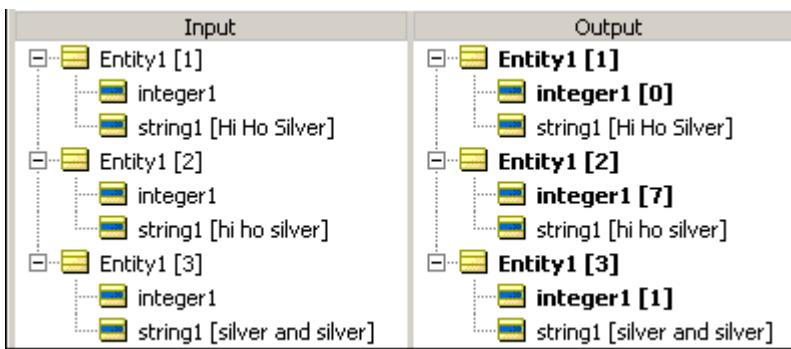
RULESHEET EXAMPLE

The following Rulesheet uses `.indexOf` to evaluate whether `string1` includes the characters `silver` and assigns a value to `integer1` corresponding to the beginning character position of the first occurrence.



SAMPLE RULETEST

A sample Ruletest provides `string1` values for three examples. Input and Output panels are shown below. Notice sensitivity to case in example 1.



IsEmpty

SYNTAX

`<Collection> ->isEmpty`

DESCRIPTION

Returns a value of true if `<Collection>` contains *no* elements (that is, has no children). `->isEmpty` does not check for an empty or null value of an attribute, but instead checks for *existence* of elements within the collection. As such, a unique alias must be used to represent the `<Collection>` being tested.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

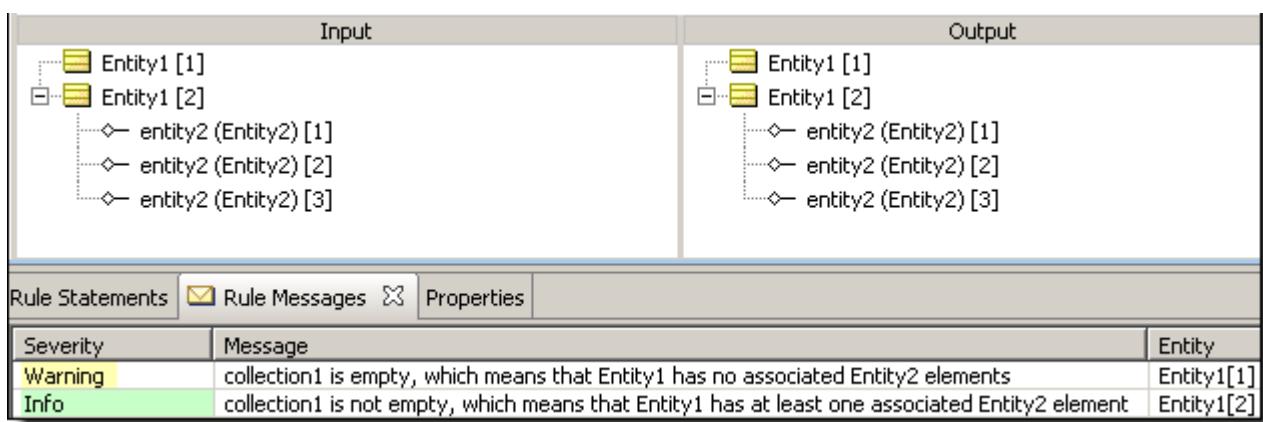
RULESHEET EXAMPLE

This sample Rulesheet uses `->isEmpty` to determine if `collection1` has any elements. Note the use of unique alias `collection1` to represent the collection of `Entity2` associated with `Entity1`.

The screenshot shows the Rulesheet editor interface with the file `IsEmpty.rules` open. The Scope tree on the left shows an `Entity1` node with a child `entity2 [collection1]`. The Conditions table in the center has four rows (a, b, c, d) and three columns (0, 1, 2). Row 'a' contains the condition `collection1 -> isEmpty`, with 'T' in column 1 and 'F' in column 2. The Actions table below it has two rows (A, B) and three columns. Row 'A' contains the action `Post Message(s)`. The Rule Statements table at the bottom lists two entries: a Warning for Entity1 stating "collection1 is empty, which means that Entity1 has no associated Entity2 elements" and an Info for Entity1 stating "collection1 is not empty, which means that Entity1 has at least one associated Entity2 element".

SAMPLE RULETEST

A sample Ruletest provides two example `collection1`. The following illustration shows Input and Output panels



Iterate

SYNTAX

```
<Collection> ->iterate(<Expression>)
```

DESCRIPTION

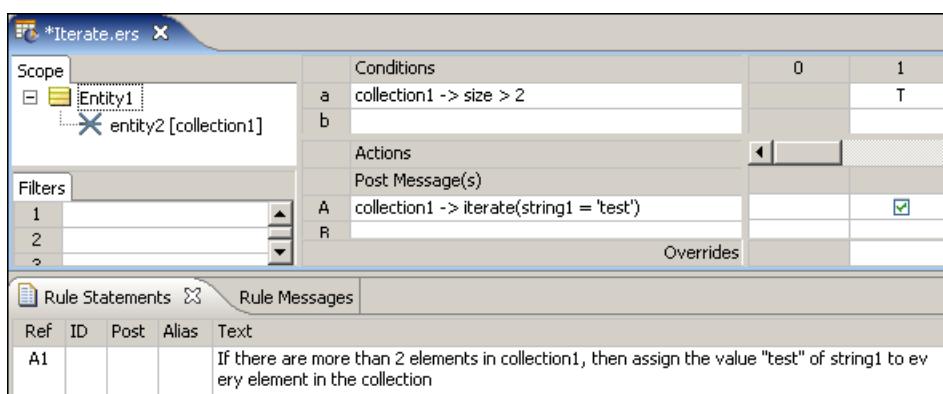
Executes `<Expression>` for every element in `<Collection>`. `<Collection>` must be expressed as a unique alias.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) does not apply. Special exceptions: `->iterate` may only be used in Action Rows (section 5 in [Sections of Rulesheet that correlate with usage restrictions](#)).

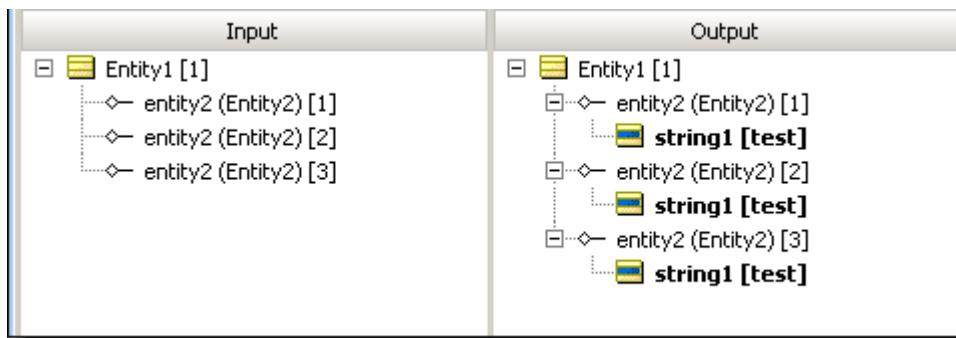
RULESHEET EXAMPLE

This sample Rulesheet uses `->iterate` to assign the value of `test` to `string1` in every element in `collection1`. See [->exists](#) for more information on this operator.



SAMPLE RULETEST

A sample Ruletest provides three elements in `collection1`. Input and Output panels are shown below.



Last

SYNTAX

<Sequence> ->**last**.<Attribute1>

DESCRIPTION

Returns the value of <Attribute1> of the last element in <Sequence>. Another operator, such as [->sortedBy](#), must be used to transform a <Collection> into a <Sequence> before **->last** may be used. <Sequence> must be expressed as a unique alias. <Attribute1> may be of any data type. See [Advanced Collection Syntax](#) for more examples of usage.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

This sample Rulesheet uses **->last** to identify the last element of the sequence created by applying [->sortedBy](#) to collection1. Once identified, the value of the string1 attribute belonging to this last element is evaluated. If the value of string1 is Joe, then boolean1 attribute of Entity1 is assigned the value of true.

The screenshot shows the Progress Corticon Studio interface with the Rule Test panel open. The panel is divided into several sections:

- Scope:** A tree view showing entities Entity_1, boolean1, entity_2 (Entity_2) [collection1], decimal1, and string1.
- Filters:** A list of filters numbered 1 through 5.
- Conditions:** A table with columns 0, 1, and 2. Row 0 contains 'Joe' under column 1 and 'not 'Joe'' under column 2. Rows 1 through 5 are empty.
- Actions:** A table with columns 0, 1, and 2. Row 0 contains a yellow envelope icon under column 1 and a blue envelope icon under column 2. Rows 1 through 5 are empty.
- Overrides:** A section at the bottom right of the conditions and actions tables.
- Rule Statements:** A table listing rule statements:

Ref	ID	Post	Alias	Text
1		Info	Entity_1	If the string1 value of the last element in collection1, in ascending order by decimal1, is equal to Joe, then boolean1 is true.
2		Warning	Entity_1	If the string1 value of the last element in collection1, in ascending order by decimal1, is equal to Joe, then boolean1 is true.
- Rule Messages:** A table listing rule messages:

Severity	Message	Entity
Info	If the string1 value of the last element in collection1, in ascending order by decimal1, is equal to Joe, then boolean1 is true.	Entity_1[1]

SAMPLE RULETEST

A sample Ruletest provides a collection of three elements, each with a decimal1 value. Input and Output panels are shown below.

The screenshot shows the Progress Corticon Studio interface with the Rule Test panel open, displaying a sample ruletest named /Examples/last.ers.

Input: Shows a collection of Entity_1 [1] elements. Each element has a boolean1 attribute and points to an entity_2 (Entity_2) [1] element. The entity_2 element has attributes decimal1 [2.500000] and string1 [Mary].

Output: Shows the same collection of Entity_1 [1] elements, but the boolean1 attribute is now set to [true]. Each element still points to an entity_2 (Entity_2) [1] element with attributes decimal1 [2.500000] and string1 [Mary].

Rule Statements: A table listing rule statements:

Severity	Message	Entity
Info	If the string1 value of the last element in collection1, in ascending order by decimal1, is equal to Joe, then boolean1 is true.	Entity_1[1]

A mouse cursor is hovering over the message in the Rule Statements table, which is highlighted with a yellow background.

Last (number)

SYNTAX

<Sequence> ->last(*integer*)

DESCRIPTION

Returns a [->subSequence](#) of the last *integer* entities in the collection <Sequence>. Another operator, such as [->sortedBy](#) or [->sortedByDesc](#), must be used to transform a <Collection> into a <Sequence> before [->last](#) can be used. <Sequence> must be expressed as a unique alias. If *integer* is larger than the number of entities in the collection, all the entities in the collection are returned. See [Advanced Collection Syntax](#) for more examples of usage.

USAGE RESTRICTIONS

The Operators row of the table in [Vocabulary usage restrictions](#) does not apply. Special exceptions: **last(x)** may only be used in Action Rows (section 5 in [Sections of Rulesheet that correlate with usage restrictions](#)).

RULESHEET EXAMPLE

This sample Rulesheet uses [->last\(2\)](#) to select the last two elements of the sequence created by applying [->sortedBy](#) to `collection2`. Once identified, the last 2 entities will be returned as the sequence `collection3`.

Scope	Conditions	
	a	
	b	
	Actions	0
Entity_1	Post Message(s)	
entity_2 (Entity_2) [collection2]	A collection3=collection2->sortedBy(collection2.decimal1)->last(2)	
decimal1		
Entity_3	B	
entity_2 (Entity_2) [collection3]		

SAMPLE RULETEST

A sample Ruletest provides a collection of five elements, each with a `decimal1` value. Input and Output panels are shown below.

Input	Output
<ul style="list-style-type: none"> Entity_1 [1] <ul style="list-style-type: none"> entity_2 (Entity_2) [1] <ul style="list-style-type: none"> decimal1 [500.000000] entity_2 (Entity_2) [2] <ul style="list-style-type: none"> decimal1 [800.000000] entity_2 (Entity_2) [3] <ul style="list-style-type: none"> decimal1 [700.000000] entity_2 (Entity_2) [4] <ul style="list-style-type: none"> decimal1 [100.000000] entity_2 (Entity_2) [5] <ul style="list-style-type: none"> decimal1 [600.000000] Entity_3 [1] 	<ul style="list-style-type: none"> Entity_1 [1] <ul style="list-style-type: none"> entity_2 (Entity_2) [1] <ul style="list-style-type: none"> decimal1 [500.000000] entity_2 (Entity_2) [2] <ul style="list-style-type: none"> decimal1 [800.000000] entity_2 (Entity_2) [3] <ul style="list-style-type: none"> decimal1 [700.000000] entity_2 (Entity_2) [4] <ul style="list-style-type: none"> decimal1 [100.000000] entity_2 (Entity_2) [5] <ul style="list-style-type: none"> decimal1 [600.000000] Entity_3 [1] <ul style="list-style-type: none"> entity_2 (Entity_2) [2] entity_2 (Entity_2) [3]

Note: The selected entities and their values are highlighted to improve readability.

RULESHEET EXAMPLE: SAME COLLECTION

In this example, the action uses the same collection for the source and the target:

Scope	Conditions	
a		0
b		
	Actions	
	Post Message(s)	
A	collection2=collection2->sortedBy(collection2.decimal1)->last(2)	<input checked="" type="checkbox"/>
B		
C		

SAMPLE RULETEST: SAME COLLECTION

The sample Ruletest shows the last 2 entities are retained in the collection, and the extraneous entities are moved out of the collection to root level:

Input	Output
<ul style="list-style-type: none"> Entity_1 [1] <ul style="list-style-type: none"> entity_2 (Entity_2) [1] <ul style="list-style-type: none"> decimal1 [500.000000] entity_2 (Entity_2) [2] <ul style="list-style-type: none"> decimal1 [800.000000] entity_2 (Entity_2) [3] <ul style="list-style-type: none"> decimal1 [700.000000] entity_2 (Entity_2) [4] <ul style="list-style-type: none"> decimal1 [100.000000] entity_2 (Entity_2) [5] <ul style="list-style-type: none"> decimal1 [600.000000] 	<ul style="list-style-type: none"> Entity_1 [1] <ul style="list-style-type: none"> entity_2 (Entity_2) [2] <ul style="list-style-type: none"> decimal1 [800.000000] entity_2 (Entity_2) [3] <ul style="list-style-type: none"> decimal1 [700.000000] Entity_2 [4] <ul style="list-style-type: none"> decimal1 [100.000000] Entity_2 [5] <ul style="list-style-type: none"> decimal1 [600.000000] Entity_2 [1] <ul style="list-style-type: none"> decimal1 [500.000000]

Note: Using the same collection as the source and the target is an important consideration because the original collection cannot be accessed again, and another iteration using this operator would likely produce a different result.

Less than

SYNTAX

DateTime*	<DateTime1> < <DateTime2>
Number*	<Number1> < <Number2>
String	<String1> < <String2>

DESCRIPTION

DateTime*	Returns a value of true if <DateTime1> is less than <DateTime2>. This is equivalent to <DateTime1> occurring “before” <DateTime2>
Number	Returns a value of true if <Number1> is less than <Number2>. Different numeric data types may be compared in the same expression.
String	Returns a value of true if <String1> is less than <String2>. Corticon Studio uses Character precedence: Unicode and Java Collator on page 181.

*includes DateTime, Date, or Time data types

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies, with the following exception: **less than** may also be used in Conditional Value Sets & Cells (section 5 in [Sections of Rulesheet: Numbers Correlate with Table Above](#)).

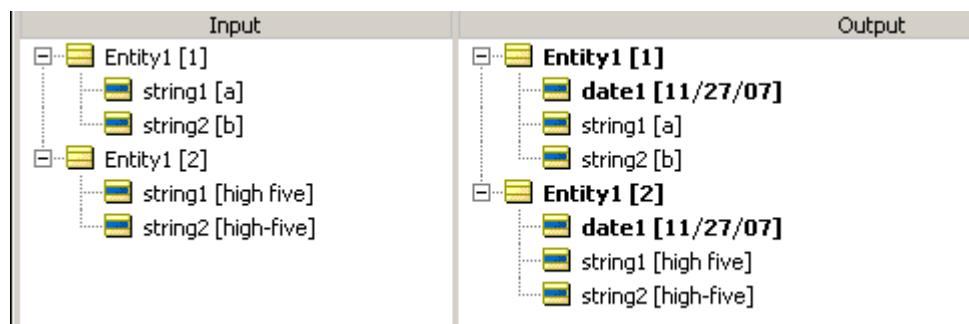
RULESHEET EXAMPLE

The following Rulesheet uses **less than** to test whether `string1` is less than `string2`, and assign today's date to `dateTime1` if it is. See [today](#) for an explanation of this literal term.

Ref	ID	Post	Alias	Text
1				If string1 is less than string2, then assign today's date to string1
2				If string1 is not less than string2, then take no action

SAMPLE RULETEST

A sample Ruletest provides two examples. Input and Output panels are shown below:



Less than or equal to

SYNTAX

DateTime*	<code><DateTime1> <= <DateTime2></code>
Number*	<code><Number1> <= <Number2></code>
String	<code><String1> <= <String2></code>

DESCRIPTION

DateTime*	Returns a value of true if <code><DateTime1></code> is less than or equal to <code><DateTime2></code> . This is equivalent to <code><DateTime1></code> occurring "on or before" <code><DateTime2></code>
Number	Returns a value of true if <code><Number1></code> is less than or equal to <code><Number2></code> . Different numeric data types may be compared in the same expression.
String	Returns a value of true if <code><String1></code> is less than or equal to <code><String2></code> . Corticon Studio uses Character precedence: Unicode and Java Collator on page 181.

*includes DateTime, Date, or Time data types

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies, with the following exception: **less than or equal to** may also be used in Conditional Value Sets & Cells (section 5 of [Sections of Rulesheet that correlate with usage restrictions](#)).

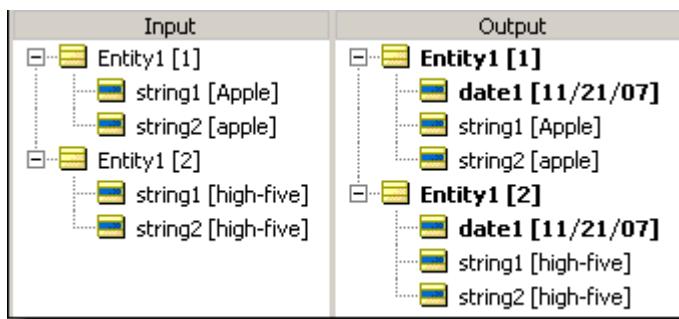
RULESHEET EXAMPLE

The following Rulesheet uses **less than or equal to** to test whether `string1` is less than or equal to `string2`, and assign today's date to `dateTime1` if it is. See [today](#) for an explanation of this literal term.

Ref	ID	Post	Alias	Text
1				If string1 is less than or equal to string2, then assign today's date to dateTime1
2				If string1 is not less than or equal to string2, then take no action

SAMPLE RULETEST

A sample Ruletest provides two examples. Input and Output panels are shown below:



Logarithm (Base 10)

SYNTAX

<Number>.log

DESCRIPTION

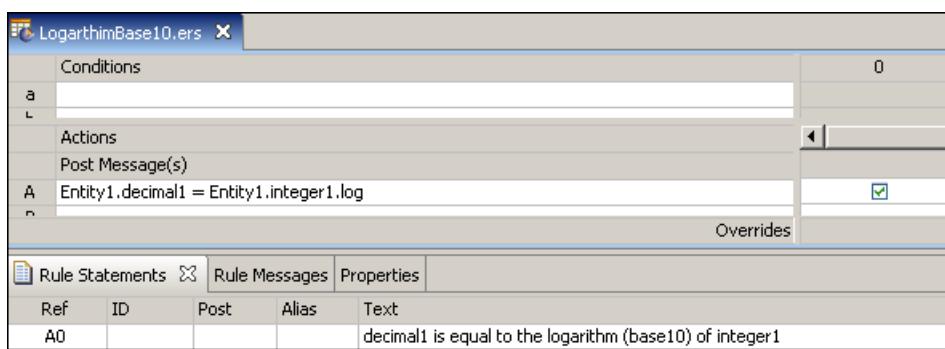
Returns a Decimal value equal to the logarithm (base 10) of <Number>. If <Number> is equal to 0 (zero) an error is returned when the rule is executed.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

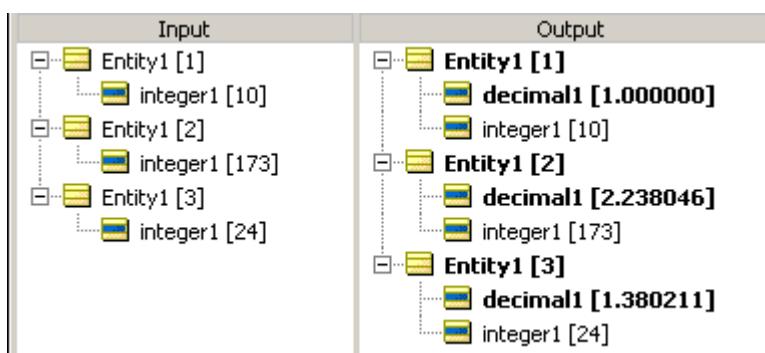
RULESHEET EXAMPLE

The following Rulesheet uses .log to calculate the logarithm (base 10) of integer1 and assign it to decimal1.



SAMPLE RULETEST 1

A sample Ruletest provides results for three examples of integer1. Input and Output panels are shown below:



SAMPLE RULETEST 2

Another sample Ruletest for three examples of `integer1` where one example is equal to zero (0). The resulting error is illustrated below. Notice that the error encountered in the second example causes all Rulesheet execution to halt, leaving the third example unprocessed.

Input	Output	Expected
Entity_1 [1] Integer1 [10]	Entity_1 [1] Decimal1 [1.000000] Integer1 [10]	
Entity_1 [2] Integer1 [0]	Entity_1 [2] Integer1 [0]	
Entity_1 [3] Integer1 [24]	Entity_1 [3] Decimal1 [1.380211] Integer1 [24]	

Severity	Message	Entity
Violation	The system has encountered a data value with an unexpected format: null	

Logarithm (Base x)

SYNTAX

`<Number>.log(<Decimal>)`

DESCRIPTION

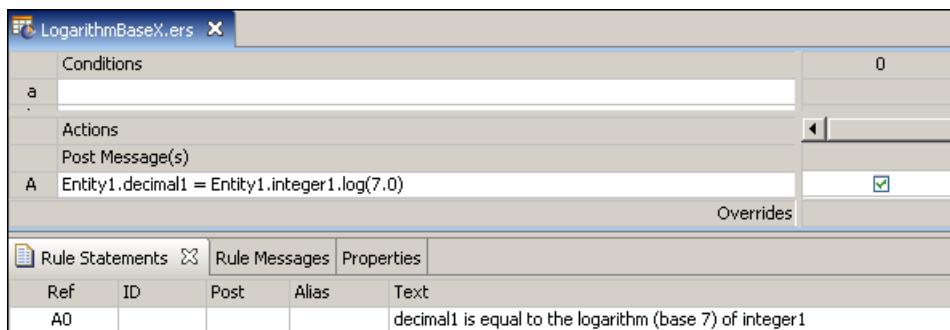
Returns a Decimal value equal to the logarithm (base `<Decimal>`) of `<Number>`. If `<Number>` is equal to 0 (zero) an error is returned when the rule is executed.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

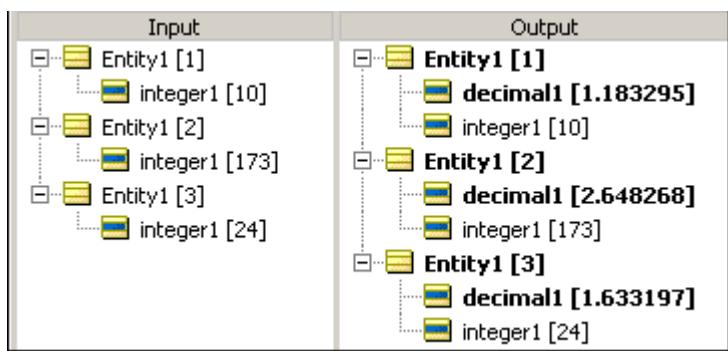
RULESHEET EXAMPLE

The following Rulesheet uses `.log` to calculate the logarithm (base 7.0) of `integer1` and assign it to `decimal1`.



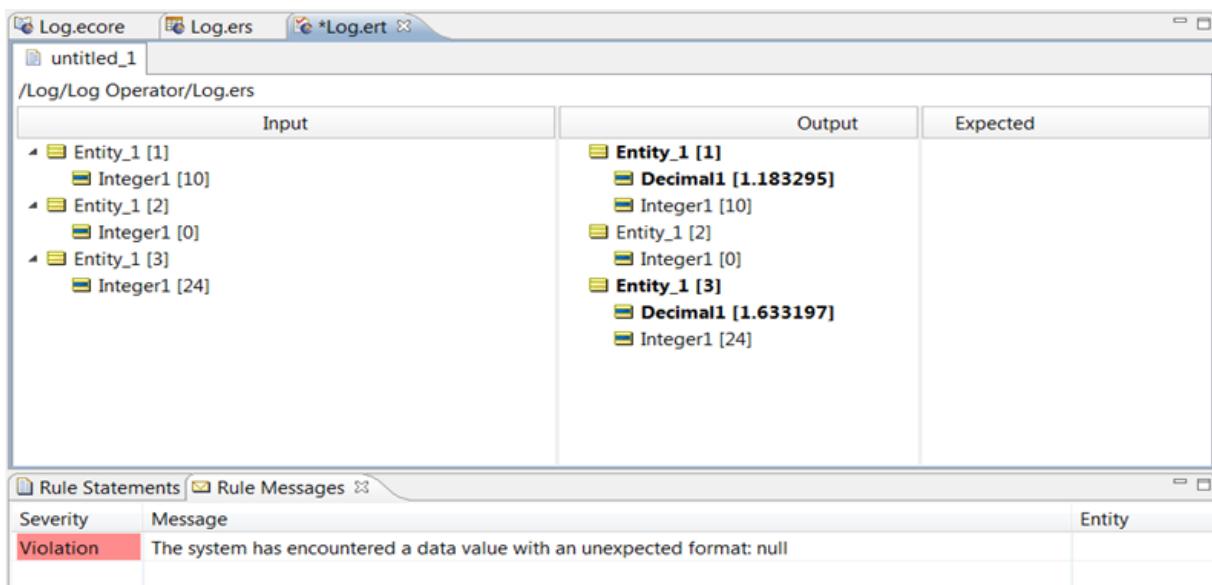
SAMPLE RULETEST 1

A sample Ruletest provides results for three examples of `integer1`. Input and Output panels are shown below:



SAMPLE RULETEST 2

Another sample Ruletest for three examples of `integer1` where one example is equal to zero (0). The resulting error is illustrated below.



Lowercase

SYNTAX

<String>.toLowerCase

DESCRIPTION

Converts all characters in <String> to lowercase characters.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses `.toLowerCase` to convert `string1` to lowercase, compare its value with `string2`, and assign a value to `boolean1` based on the results of the comparison.

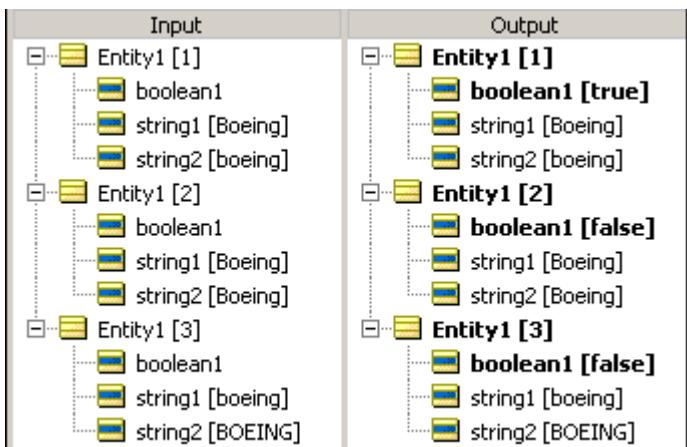
	Conditions	0	1	2
a	Entity1.string1.toLowerCase=Entity1.string2		T	F
b				
c				
d				
e				
	Actions			
	Post Message(s)			
A	Entity1.boolean1		T	F
B				
C				
	Overrides			

Rule Statements Rule Messages Properties

Ref	ID	Post	Alias	Text
1				If string1 converted to lowercase is equal to string2, then assign boolean1 a value of true
2				If string1 converted to lowercase is not equal to string2, then assign boolean1 a value of false

SAMPLE RULETEST

A sample Ruletest provides three examples of `string1` and `string2`. Input and Output panels are shown below:



Maximum value

SYNTAX

`<Number1>.max(<Number2>)`

DESCRIPTION

Returns either `<Number1>` or `<Number2>`, whichever is greater.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses `.max` to compare the values of `decimal1` and `decimal2`, and `integer1` and `integer2`, and posts a message based on their size relative to 5.0 and 8, respectively.

The screenshot shows the 'MaximumValue.ers' Rulesheet in the Rulesheet editor. The Conditions section contains three rows: 'a' (Entity1.decimal1.max(Entity1.decimal2) > 5.0), 'b' (Entity1.integer1.max(Entity1.integer2) > 8), and 'c'. The Actions section contains a 'Post Message(s)' action. The Rule Statements tab shows two entries:

Ref	ID	Post	Alias	Text
1		Info	Entity1	The larger of decimal1 and decimal2 is greater than 5
2		Info	Entity1	The larger of integer1 and integer2 is greater than 8

SAMPLE RULETEST

A sample Ruletest provides four examples, two using `decimal1` and `decimal2`, and two using `integer1` and `integer2` as input data.

The screenshot shows the Ruletest editor. The Input pane displays four entities (Entity1 [1] through Entity1 [4]) with their respective attribute values. The Output pane shows the same entities with their attribute values. The Rule Statements tab lists two messages:

Severity	Message	Entity
Info	The larger of decimal1 and decimal2 is greater than 5	Entity1[1]
Info	The larger of integer1 and integer2 is greater than 8	Entity1[3]

Maximum value (Collection)

SYNTAX

`<Collection.attribute> -> max`

DESCRIPTION

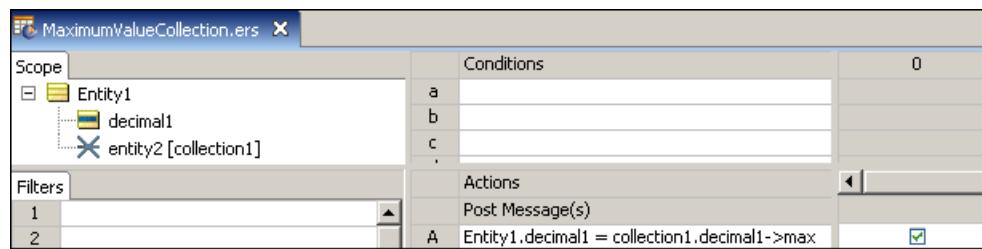
Returns the highest value of `<attribute>` for all elements in `<Collection>`. `<attribute>` must be a numeric data type. `<Collection>` must be expressed as a unique alias.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

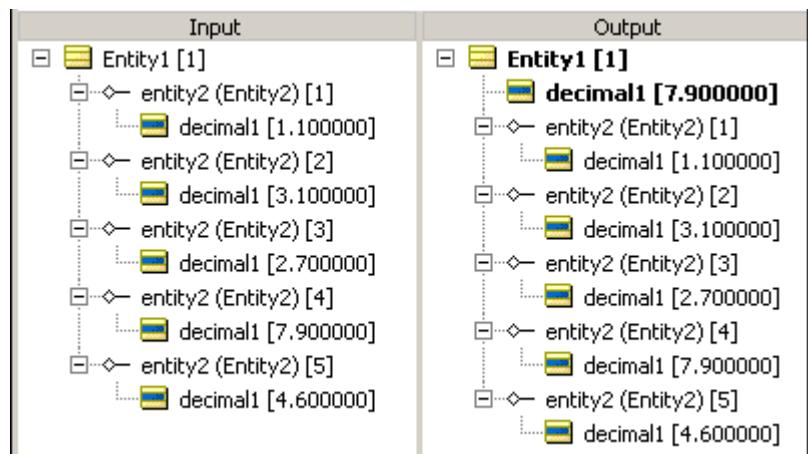
RULESHEET EXAMPLE

The following Rulesheet uses `->max` to identify the highest value of `decimal1` in all elements of `collection1`, then assign it to `Entity1.decimal1`.



SAMPLE RULETEST

A sample collection contains five elements, each with a value of `decimal1`.



Minimum value

SYNTAX

`<Number1>.min(<Number2>)`

DESCRIPTION

Returns either `<Number1>` or `<Number2>`, whichever is smaller.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses `.min` to compare the values of `decimal1` and `decimal2`, and `integer1` and `integer2`, and posts a message based on their size relative to 5.0 and 8, respectively.

Ref	ID	Post	Alias	Text
1		Info	Entity1	The smaller of decimal1 and decimal2 is greater than 5
2		Info	Entity1	The smaller of integer1 and integer2 is greater than 8

SAMPLE RULETEST

A sample Ruletest provides four examples, two using decimal inputs, and two using integers.

Severity	Message	Entity
Info	The smaller of decimal1 and decimal2 is greater than 5	Entity1[2]
Info	The smaller of integer1 and integer2 is greater than 8	Entity1[3]

Minimum value (Collection)

SYNTAX

`<Collection.attribute> -> min`

DESCRIPTION

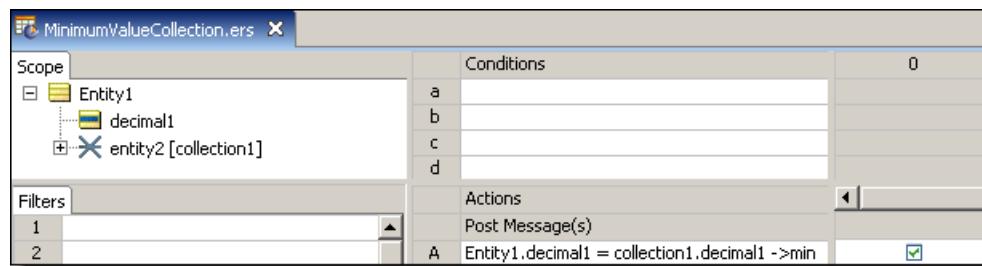
Returns the lowest value of `<attribute>` for all elements in `<Collection>`. `<attribute>` must be a numeric data type. `<Collection>` must be expressed as a unique alias.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

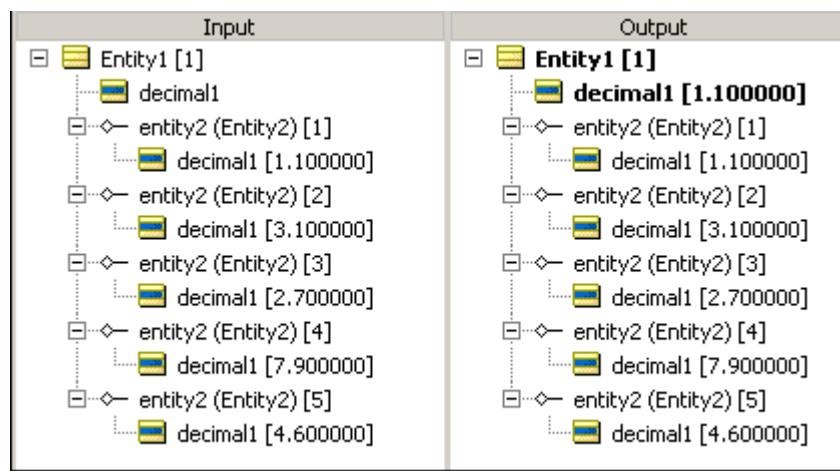
RULESHEET EXAMPLE

The following Rulesheet uses `->min` to identify the lowest value of `decimal1` in all elements of `collection1`, then assign it to `Entity1.decimal1`.



SAMPLE RULETEST

A sample collection contains five elements, each with a value of `decimal1`.



Minute

SYNTAX

`<DateTime>.min`

`<Time>.min`

DESCRIPTION

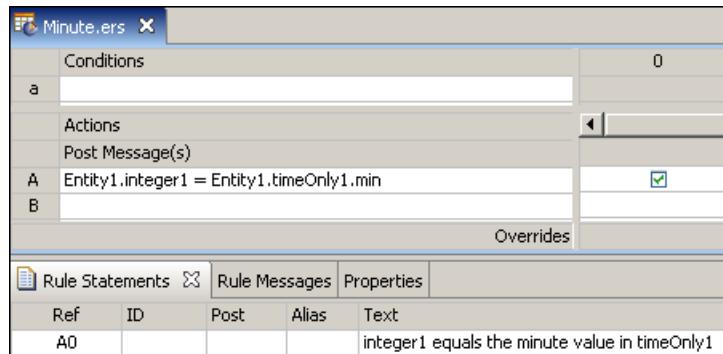
Returns the minute portion of `<DateTime>` or `<Time>` as an Integer between 0 and 59. This operator cannot be used with Date attributes because no time information is present.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

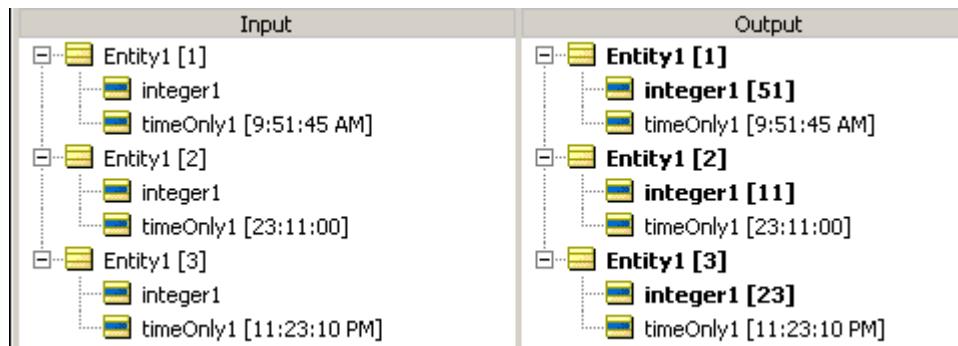
RULESHEET EXAMPLE

The following Rulesheet uses `.min` to evaluate `dateTime1` and assign the minute value to `integer1`.



SAMPLE RULETEST

A sample Ruletest provides three examples of `dateTime1`. Input and Output panels are shown below:



Minutes between

SYNTAX

```
<DateTime1>.minsBetween(<DateTime2>)
<Time1>.minsBetween(<Time2>)
```

DESCRIPTION

Returns the Integer number of minutes between DateTimes or between Times. The function calculates the number of milliseconds between the two dates and divides that number by 60,000 (the number of milliseconds in a minute). The decimal portion is then truncated. If the two dates differ by less than a full minute, the returned value is zero. This function returns a positive number if `<DateTime2>` is later than `<DateTime1>`.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses **.minsBetween** to determine the number of minutes that have elapsed between `dateTime1` and `dateTime2`, compare it to the Values set, and assign a value to `string1`.

Conditions	0	1	2
a Entity1.dateTime1.minsBetween(Entity1.dateTime2)	<=30	>30	>30
b			

Actions			
Post Message(s)			
A Entity1.string1	'Not Overdue'	'Overdue'	

Overrides				

Rule Statements

Ref	ID	Post	Alias	Text
1				If 30 or fewer minutes have elapsed between dateTime1 and dateTime2, then Entity1 is not overdue
2				If more than 30 minutes have elapsed between dateTime1 and dateTime2, then Entity2 is overdue

SAMPLE RULETEST

A sample Ruletest provides `dateTime1` and `dateTime2` for two examples. Input and Output panels are shown below. Notice the different masks (formats) used for the DateTime data.

Input		Output	
Entity1 [1]	dateTime1 [3/14/2006 4:00:00 PM EST]	Entity1 [1]	dateTime1 [3/14/2006 4:00:00 PM EST]
	dateTime2 [3/15/2006 2:30:00 AM EST]		dateTime2 [3/15/2006 2:30:00 AM EST]
Entity1 [2]	dateTime1 [November 23, 2005 12:30:00 EST]	Entity1 [2]	dateTime1 [November 23, 2005 12:30:00 EST]
	dateTime2 [November 23, 2005 12:10:00 EST]		dateTime2 [November 23, 2005 12:10:00 EST]

Mod

SYNTAX

`<Integer1>.mod(<Integer2>)`

DESCRIPTION

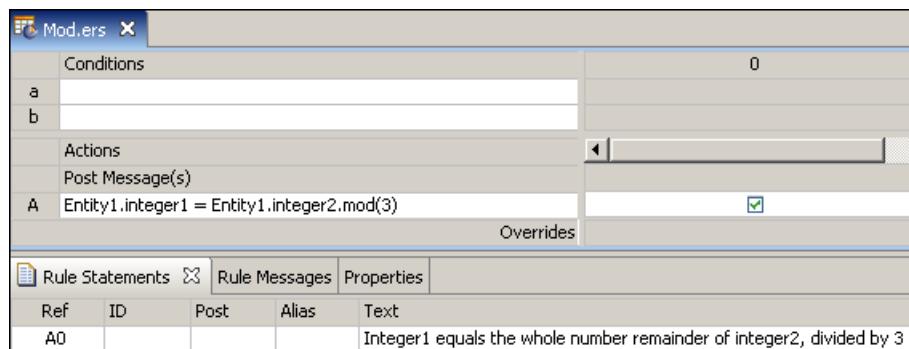
Returns the whole number remainder that results from dividing `<Integer1>` by `<Integer2>`. If the remainder is a fraction, then 0 (zero) is returned.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

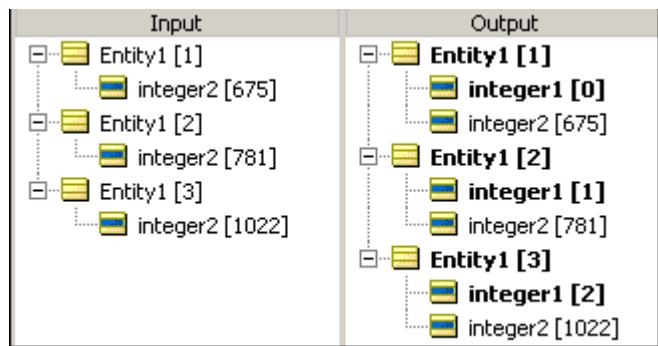
RULESHEET EXAMPLE

The following Rulesheet > uses **.mod** to calculate the whole number remainder resulting from the division of integer2 by 3. The result is assigned to integer1.



SAMPLE RULETEST

A sample Ruletest provides three examples of integer2. Input and Output panels are shown below.



Month

SYNTAX

<DateTime>.month

<Date>.month

DESCRIPTION

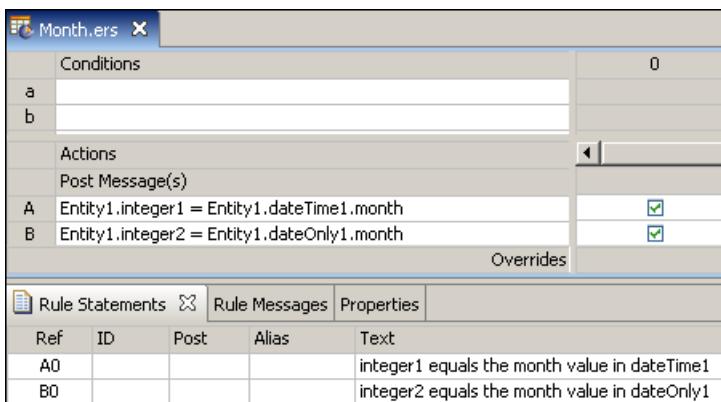
Returns the month in <DateTime> or <Date> as an Integer between 1 and 12.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

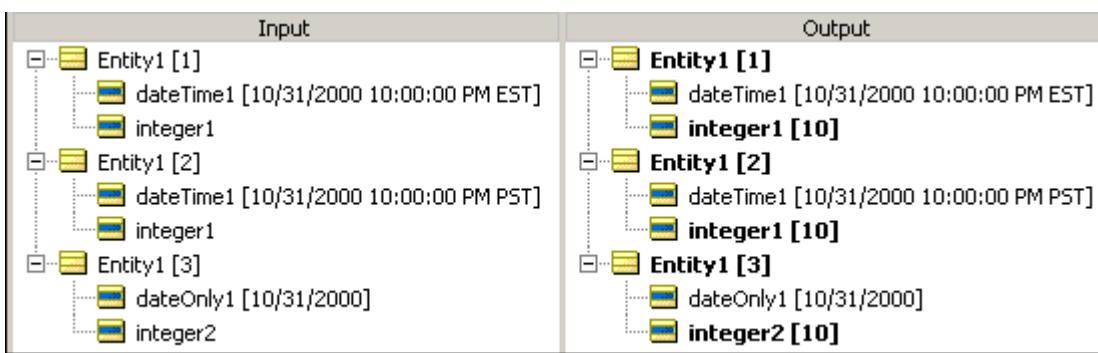
RULESHEET EXAMPLE

The following Rulesheet uses **.month** to evaluate dateTime1 and dateOnly1 and assign the month value to integer1 and integer2, respectively.



SAMPLE RULETEST

A sample Ruletest provides three examples of `dateTime1` or `dateOnly1`. Input and Output panels are shown below. The month returned is independent of the machine running the Ruletest and only depends on the locale/timezone of the data itself.



Months between

SYNTAX

`<DateTime1>.monthsBetween(<DateTime2>)`

`<Date1>.monthsBetween(<Date2>)`

DESCRIPTION

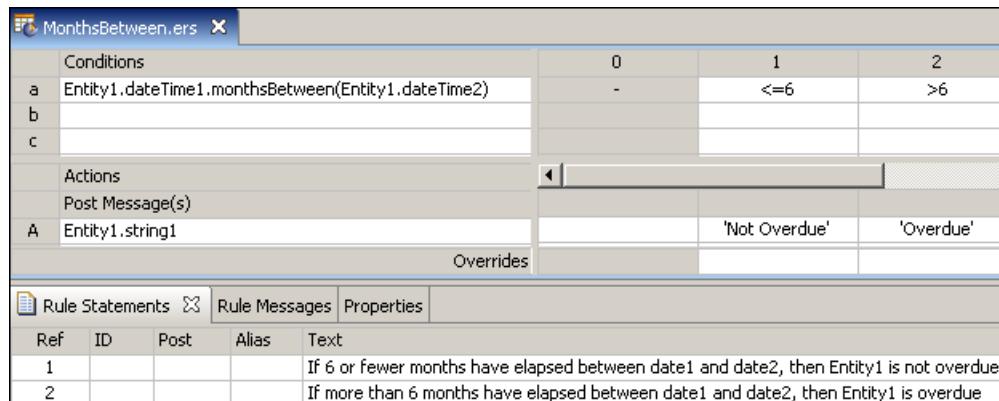
Returns the Integer number of months between DateTimes or between Dates. The month and year portions of the date data are subtracted to calculate the number of elapsed months. The day portions are ignored. If the month and year portions are the same, the result is zero. This function returns a positive number if `<DateTime2>` is later than `<DateTime1>`.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

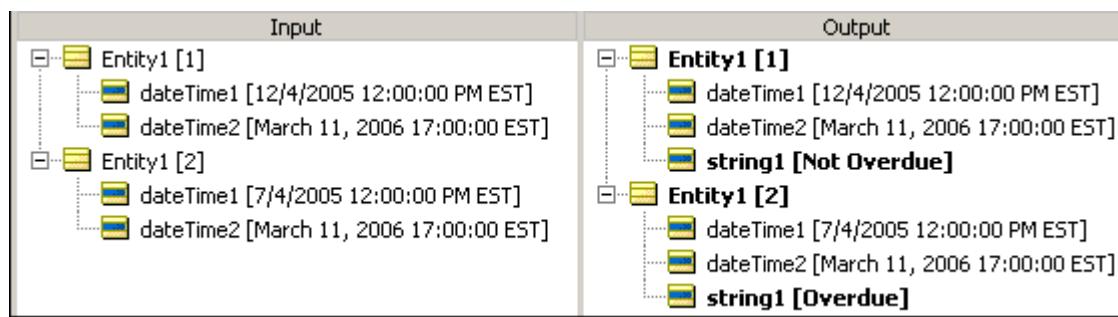
RULESHEET EXAMPLE

The following Rulesheet uses **.monthsBetween** to determine the number of months that have elapsed between `dateTime1` and `dateTime2`, compare it to the values in the Condition Cells, and assign a value to `string1`.



SAMPLE RULETEST

A sample Ruletest provides `dateTime1` and `dateTime2` for two examples. Input and Output panels are shown below. Notice the variations in date masks (formats).



Multiply

SYNTAX

`<Number1> * <Number2>`

DESCRIPTION

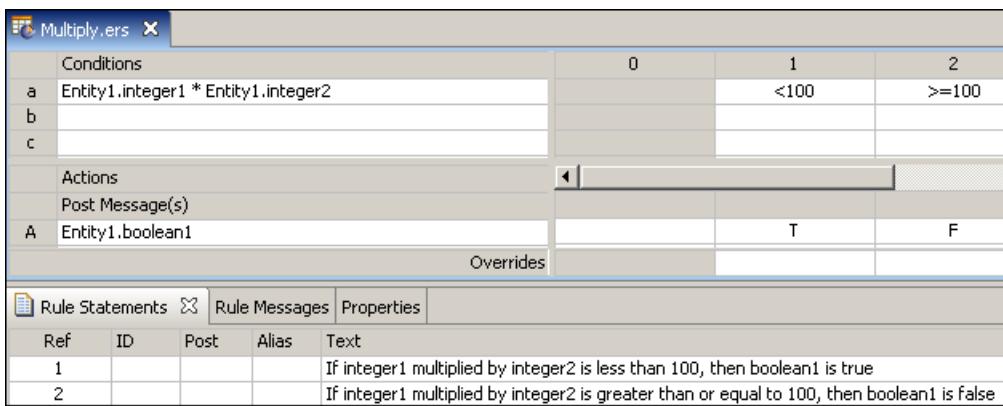
Multiplies `<Number1>` by `<Number2>`. The resulting data type is the more expansive of those of `<Number1>` and `<Number2>`.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

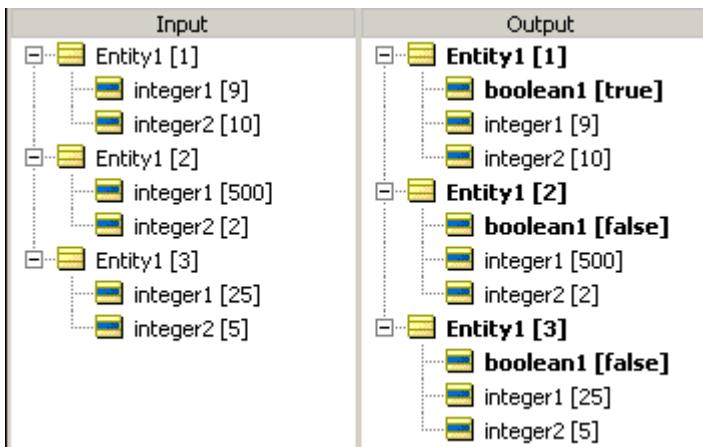
RULESHEET EXAMPLE

This sample Rulesheet uses **multiply** to multiply `integer1` and `integer2` and compare the result to 100



SAMPLE RULETEST

A sample Ruletest provides three examples of `integer1` and `integer2`. Input and Output panels are shown below.



Natural logarithm

SYNTAX

`<Number>.ln`

DESCRIPTION

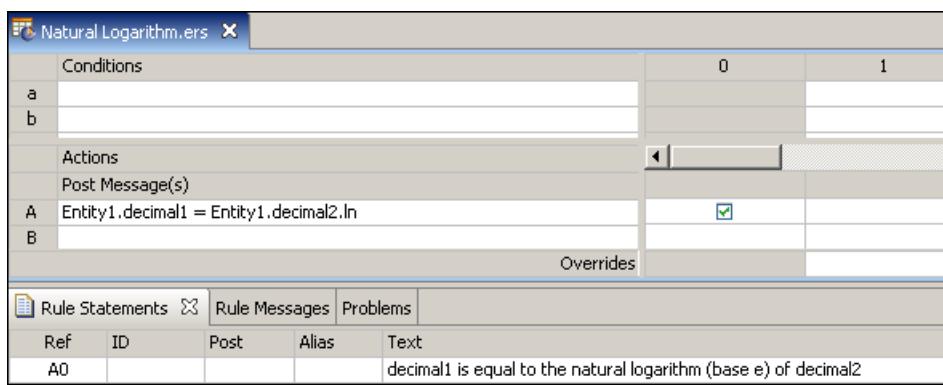
Returns a Decimal value equal to the natural logarithm (base e) of `<Number>`. If `<Number>` is equal to 0 (zero), an error is returned when the rule is executed. This error will halt execution for all data present.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

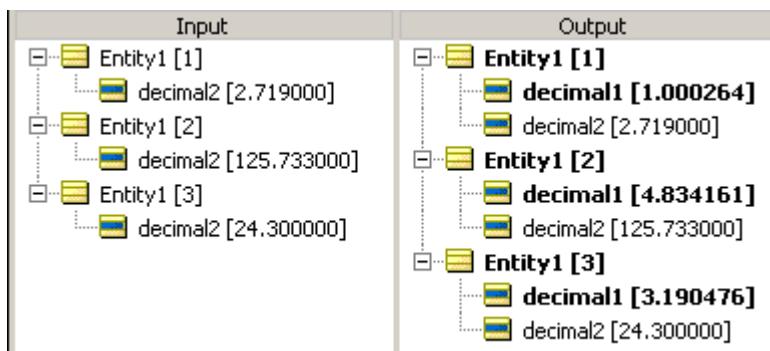
RULESHEET EXAMPLE

The following Rulesheet uses `.ln` to calculate the natural logarithm of `decimal2` and assign it to `decimal1`.



SAMPLE RULETEST 1

A sample Ruletest provides results for three examples of `decimal2`. Input and Output panels are shown below:



SAMPLE RULETEST 2

Another sample Ruletest for three examples of `decimal2` where one example is equal to zero (0). The resulting error is illustrated below:

Input	Output	Expected
Entity_1 [1] └─ Decimal2 [2.719000]	Entity_1 [1] └─ Decimal1 [1.000264] └─ Decimal2 [2.719000]	
Entity_1 [2] └─ Decimal2 [0.000000]	Entity_1 [2] └─ Decimal1 [0.000000] └─ Decimal2 [0.000000]	
Entity_1 [3] └─ Decimal2 [24.300000]	Entity_1 [3] └─ Decimal1 [3.190476] └─ Decimal2 [24.300000]	

Rule Messages

Severity	Message	Entity
Violation	The system has encountered a data value with an unexpected format: null	

New

SYNTAX

<Entity>.new[<Expression1>,<Expression2>...]

DESCRIPTION

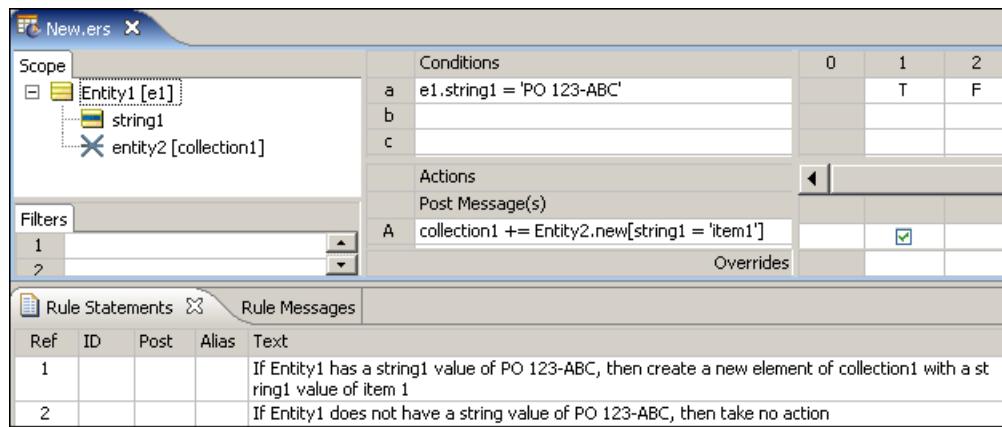
creates a new <Entity> with attribute values defined by optional <Expression>. Expressions (when present) should be written as assignments in the form: *attribute = value*. The attribute used in <Expression> (when present) must be an attribute of <Entity>.

USAGE RESTRICTIONS

The Operators row in the table of [Summary Table of Vocabulary Usage Restriction](#) does not apply. Special exceptions: **new** may only be used in Action Rows (section 5 in [Sections of Rulesheet that correlate with usage restrictions](#)).

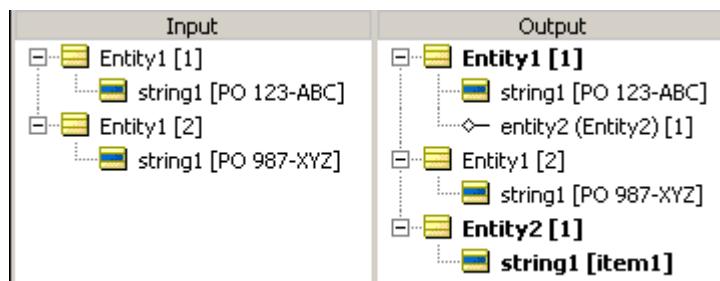
RULESHEET EXAMPLE

The following Rulesheet uses `.new` to create a new Entity2 element in collection1 when Entity1 has a string1 value equal to "PO 123-ABC". An alias is not required by the `.new` operator, because it is possible to create a new entity at the root level, without inserting it into a collection. The collection1 alias used here is required by the `+ =` ([Associate Element to collection](#)) operator.



SAMPLE RULETEST

A sample Ruletest provides 2 collections of Entity1. Input and Output panels are illustrated below:



Behavior of the `.new` operator

The `.new` operator does not consider implied conditions of non-mandatory attributes (from the initialize expressions) during execution (in other words, a `.new` operator always fires when explicit conditions are met).

Each initialize expression within a `.new...` expression will be executed (or not) depending upon implied conditions; that is, if any input to the expression is null, the target attribute remains null. Another case where an implied condition would prevent a `.new` operator for executing is where the new entity is a target to an association assignment and the parent of that association does not exist.

The following examples assume that all attributes are not mandatory.

- Rule 1:

```
IF entity1.attr1 > 10 THEN Entity2.new[attr1 = entity1.attr2]
```

Executes only if `entity1` exists, `entity1.attr1` is not null, and `entity1.attr1 > 10`. The `newEntity2.attr1` will be left as null if `entity1.attr2` is null.

- Rule 2:

```
Entity2.new[attr1 = entity1.attr1 + entity1.attr2]
```

Will always execute. `Entity2.attr1` will remain null if `entity1` does not exist, or `entity1.attr1` is null, or `entity1.attr2` is null.

- Rule 3:

```
entity1.assoc2 += Entity2.new[attr1 = entity1.attr1]
```

Will execute only if `entity1` exists. `Entity2.attr1` will remain null if `entity1.attr1` is null.

- Rule 4:

```
Entity2.new[attr1 = entity1.assoc1.attr1]
```

This action will always fire. `Entity2.attr1` will remain null if `entity1` does not exist, or `entity1.assoc1` does not exist, or `entity1.assoc1.attr1` is null. Note that this action will fire multiple times if `entity1.assoc1` contains multiple entities (once for each entity contained in the `entity1.assoc1` collection).

New unique

SYNTAX

`<Entity>.newUnique[<Expression1>,<Expression2>...]`

DESCRIPTION

newUnique is an unusual operator in that it contains both action *and* condition logic. When an Action containing this operator is executed, a new `<Entity>` will be created only if no other entity exists with the characteristics defined by `<Expression1>` **and** `<Expression2>`, etc. `<Expression1>` and `<Expression2>` are optional. If no expression is present within the square brackets [. . .], the **newUnique** operator will create a new entity only if none currently exists in memory.

USAGE RESTRICTIONS

The Operators row in the table of [Summary Table of Vocabulary Usage Restriction](#) does not apply. Special exceptions: **newUnique** may only be used in Action Rows (section 5 in [Sections of Rulesheet that correlate with usage restrictions](#)).

There is some restriction to using **newUnique** with associations. **newUnique** is valid for associations of multiplicity One to One or Many to One, but is invalid for associations One to Many or Many to Many, as illustrated:

The screenshot shows the Rulesheet editor interface. On the left, there's a tree view under 'Scope' with categories: ManyToOne, OneToOne, and Root. Under Root, it lists manyToMany (ManyToMany), manyToOne (ManyToOne), oneToMany (OneToMany), and oneToOne (OneToOne). On the right, there are three main sections: 'Conditions' (a through j), 'Actions' (Post Message(s) followed by four actions labeled A through D), and 'Filters' (1 through 4). Actions A and B are in green, while C and D are in red.

Conditions
a
b
c
d
e
f
g
h
i
j

Actions
Post Message(s)
A Root.newUnique[oneToOne = OneToOne]
B Root.newUnique[manyToOne = ManyToOne]
C Root.newUnique[oneToMany = OneToMany]
D Root.newUnique[manyToMany = ManyToMany]

Filters
1
2
3
4

RULESHEET EXAMPLE

The following Rulesheet uses **.newUnique** to create a new Entity2 element with `string1="item1"`, and add it to `collection1` only if no existing Entity2 already has `string1="item1"`. A collection alias is not required by the **.newUnique** operator because it is possible to create a new entity at the root level, without inserting it into a collection. The collection alias used here is required by the `+=(Associate Element to collection)` operator.

This screenshot shows a more detailed view of the Rulesheet editor for a rule named 'NewUnique.ers'. The 'Scope' panel shows a hierarchy with Entity1 [e1] containing string1 and entity2 [collection1]. The 'Conditions' table has rows for 'a' (e1.string1 = 'PO 123-ABC') and 'b'. The 'Actions' section contains a single action 'A' which performs `collection1 += Entity2.newUnique[string1 = 'item1']`. The 'Overrides' section has a checked checkbox. Below the editor is a table of 'Rule Statements' with two entries, each with a descriptive text.

Conditions	0	1	2
a e1.string1 = 'PO 123-ABC'		T	F
b			

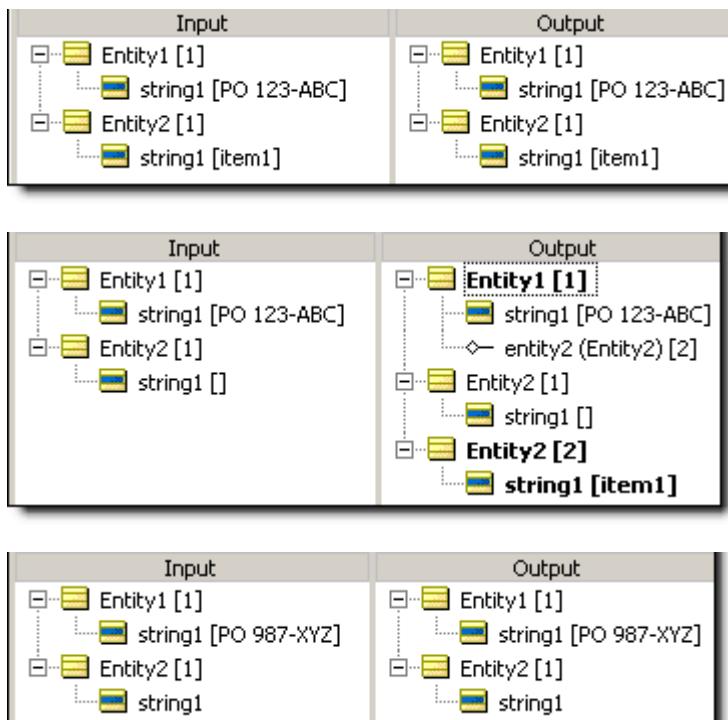
Actions
Post Message(s)
A collection1 += Entity2.newUnique[string1 = 'item1']

Overrides
<input checked="" type="checkbox"/>

Ref	ID	Post	Alias	Text
1				If the parent of collection1 has a string1 value of PO 123-ABC, then create a child entity with string1 = item1 only if none exists
2				If the parent of collection1 does not have a string1 value of PO 123-ABC, then take no action

SAMPLE RULETEST 1

Each of three sample tests provides different combinations of Entity1 and Entity2. Input and Output panels are illustrated below:



Not

SYNTAX

```
not <Expression>
```

DESCRIPTION

Returns the negation of the truth value of <Expression>.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies, with the following special exception: **not** may also be used in Conditional Cells.

RULESHEET EXAMPLE

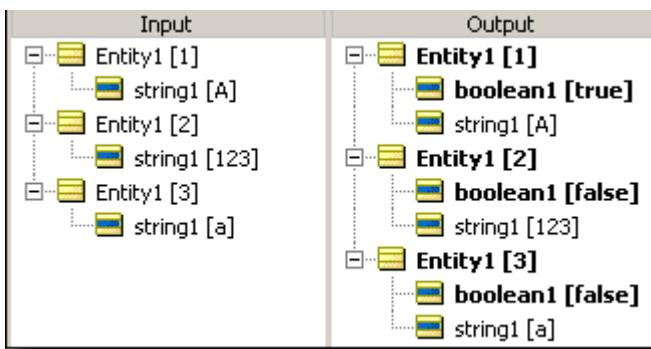
The following Rulesheet uses **not** to negate the value of **A** in the Condition Cell of rule 2. **Not** may only be used in this manner if there is at least one other value (including **other** or **null**) present in the Condition Cells values drop-down list (in other words, there must be at least one alternative to the value negated by **not**).

Conditions	0	1	2
a Entity1.string1		'A'	not 'A'
Actions			
Post Message(s)			
A Entity1.boolean1		T	F
Overrides			

Rule Statements		Rule Messages	Properties	
Ref	ID	Post	Alias	Text
1				If string1 is equal to A, then boolean1 is assigned the value of true
2				If string1 is not equal to A, then boolean1 is assigned the value of false

SAMPLE RULETEST

A sample Ruletest provides three examples of `string1`. Input and Output panels are shown below:



Not empty

SYNTAX

`<Collection> ->notEmpty`

DESCRIPTION

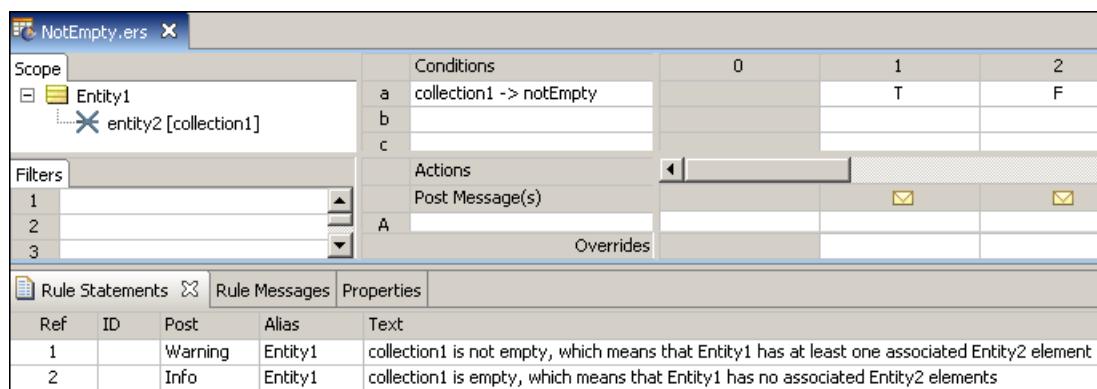
Returns a value of true if `<Collection>` contains *at least one* element. `->notEmpty` does not check for attribute values, but instead checks for the *existence of elements within a collection*. As such, it requires the use of a unique alias to represent the collection being tested.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

This sample Rulesheet uses the `->notEmpty` function to determine if `collection1` has elements. Note the use of unique alias `collection1` to represent the collection of `Entity2` associated with `Entity1`.



SAMPLE RULETEST

A sample Ruletest provides two collections. The following illustration shows Input and Output panels

Input		Output										
<ul style="list-style-type: none"> Entity1 [1] Entity1 [2] <ul style="list-style-type: none"> entity2 (Entity2) [1] entity2 (Entity2) [2] 	<ul style="list-style-type: none"> Entity1 [1] Entity1 [2] <ul style="list-style-type: none"> entity2 (Entity2) [1] entity2 (Entity2) [2] 											
<input type="button" value="Rule Statements"/> <input type="button" value="Rule Messages"/> <input type="button" value="Properties"/>		<table border="1"> <thead> <tr> <th>Severity</th> <th>Message</th> <th>Entity</th> </tr> </thead> <tbody> <tr> <td>Warning</td> <td>collection1 is not empty, which means that Entity1 has at least one associated Entity2 element</td> <td>Entity1[2]</td> </tr> <tr> <td>Info</td> <td>collection1 is empty, which means that Entity1 has no associated Entity2 elements</td> <td>Entity1[1]</td> </tr> </tbody> </table>		Severity	Message	Entity	Warning	collection1 is not empty, which means that Entity1 has at least one associated Entity2 element	Entity1[2]	Info	collection1 is empty, which means that Entity1 has no associated Entity2 elements	Entity1[1]
Severity	Message	Entity										
Warning	collection1 is not empty, which means that Entity1 has at least one associated Entity2 element	Entity1[2]										
Info	collection1 is empty, which means that Entity1 has no associated Entity2 elements	Entity1[1]										

Not equal to

SYNTAX

Boolean	<code><Expression1> <> <Expression2></code>
DateTime*	<code><DateTime1> <> <DateTime2></code>
Number	<code><Number1> <> <Number2></code>
String	<code><String1> <> <String2></code>

DESCRIPTION

Boolean	Returns a value of true if <Expression1> does not have the same truth value as <Expression2>.
DateTime*	Returns a value of true if <DateTime1> does not equal <DateTime2>. This is equivalent to <DateTime1> not occurring “on” <DateTime2>
Number	Returns a value of true if <Number1> is not equal to <Number2>. Different numeric data types may be compared in the same expression.
String	Returns a value of true if <String1> is not equal to <String2>. Studio uses Character precedence: Unicode and Java Collator on page 181 to determine character precedence.

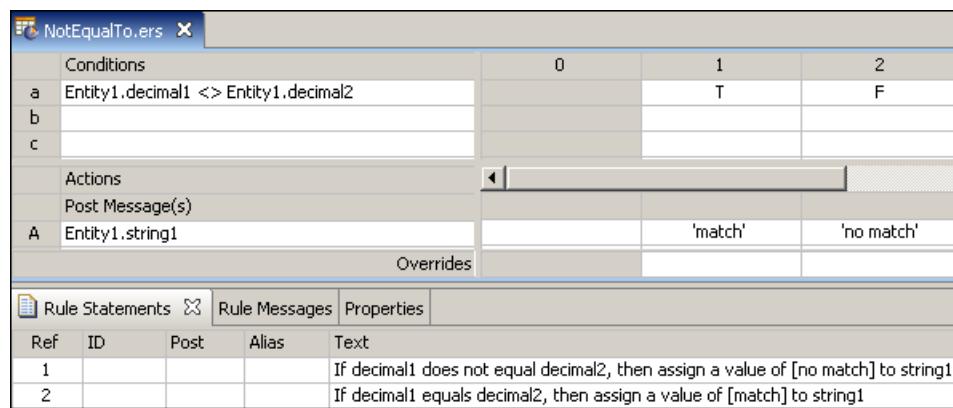
*includes DateTime, Date, and Time data types

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

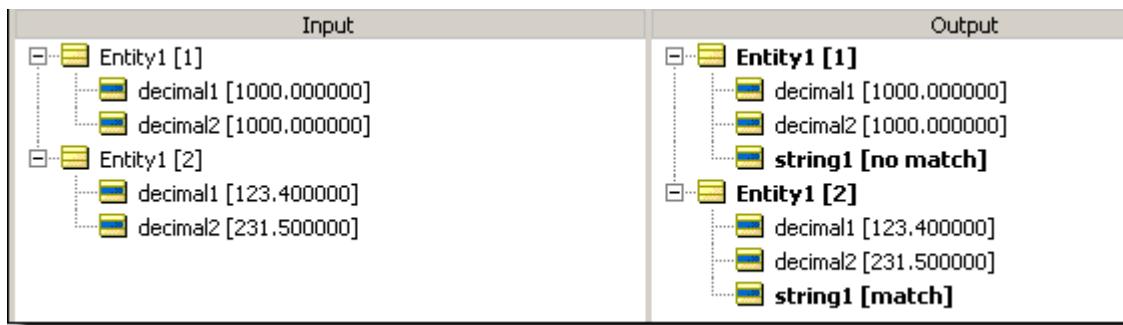
RULESHEET EXAMPLE

The following Rulesheet uses **not equal to** to test whether decimal1 equals decimal2, and assign a value to string1 based on the result of the comparison.



SAMPLE RULETEST

A sample Ruletest provides two examples. Input and Output panels are shown below:



Now

SYNTAX

`now`

DESCRIPTION

Returns the current system date and time when the rule is executed. This DateTime value is assigned the first time **now** is used in a Decision Service (Rule Set), then remains constant until the Decision Service finishes execution, regardless of how many additional times it is used. This means that every rule in a Rule Set containing **now** will use the same DateTime value.

USAGE RESTRICTIONS

The Literals row in the table of [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses **now** to determine how many hours have elapsed between now and `dateTime1` (see [.hoursBetween](#) for more details on this operator), and assign a value to `string1` based on the result.

Conditions		0	1	2
a	Entity1.dateTime1.hoursBetween(now) <2		T	F
b				
c				

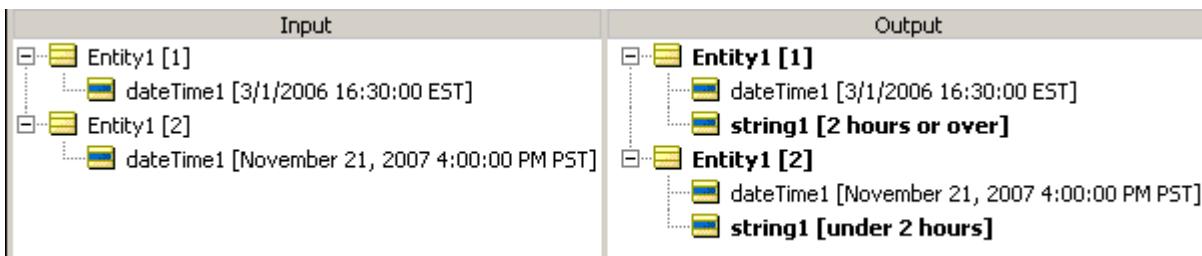
Actions			
Post Message(s)			
A	Entity1.string1	'under 2 hours'	'2 hours or over'

Overrides				

Rule Statements				
Ref	ID	Post	Alias	Text
1				If dateTime1 occurred within the last 2 hours, assign string1 a value of 'under 2 hours'
2				If dateTime1 occurred 2 hours or later from now, assign string1 a value of '2 hours or over'

SAMPLE RULETEST

A sample Ruletest provides two examples of `dateTime1`. Assume **now** is equal to Thursday, March 16, 2006 16:30:00 EST. Input and Output panels are shown below. Notice the variation in DateTime masks (formats).



Null

SYNTAX

null

DESCRIPTION

The null value corresponds to one of three different scenarios:

1. the absence of an attribute in a Ruletest Input pane or request message
2. the absence of data for an attribute in a Ruletest (the value zero counts as data)
3. a business object (supplied by an external application) that has an instance variable of null

A **null** value is different from an empty String (for String data types) or zero for numeric data types. An empty String is represented in a Ruletest as [] -- open then close square brackets. Any attribute value, including any empty strings, may be reset to **null** in a Ruletest by right-clicking the attribute and choosing **Set to null**. Mandatory attributes (property set in the Vocabulary) may not have a null value.

USAGE RESTRICTIONS

The Literals row in the table of [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses **null** to test for the existence of a real value in `decimal1`, and assign a value to `boolean1` as a result.

Ref	ID	Post	Alias	Text
1		Warning	Entity1	If decimal1 has the value of null, then assign boolean1 the value of true
2		Info	Entity1	If decimal1 has any value other than null (any real number), then assign boolean1 the value of false

SAMPLE TEST

A sample Ruletest provides four examples of `decimal1`. Input and Output panels are illustrated below. Posted messages are not shown.

The screenshot shows the Ruletest interface with two main panels: Input and Output, and a bottom section for messages.

Input Panel:

- Entity1 [1]: decimal1 [4.000000]
- Entity1 [2]: decimal1
- Entity1 [3]: decimal1 [0.000000]
- Entity1 [4]: decimal1 [-13.500000]

Output Panel:

- Entity1 [1]: boolean1 [false], decimal1 [4.000000]
- Entity1 [2]: boolean1 [true], decimal1
- Entity1 [3]: boolean1 [false], decimal1 [0.000000]
- Entity1 [4]: boolean1 [false], decimal1 [-13.500000]

Message Table:

Severity	Message	Entity
Warning	If decimal1 has the value of null, then assign boolean1 the value of true	Entity1[2]
Info	If decimal1 has any value other than null (any real number), then assign boolean1 the value of false	Entity1[4]
Info	If decimal1 has any value other than null (any real number), then assign boolean1 the value of false	Entity1[3]
Info	If decimal1 has any value other than null (any real number), then assign boolean1 the value of false	Entity1[1]

Other

SYNTAX

other

DESCRIPTION

When included in a condition's Values set (the drop-down list of values available in a Conditions Cell), **other** represents any value not explicitly included in the set, including **null**. If null is explicitly included in the Values set, then **other** does not include null.

USAGE RESTRICTIONS

The Literals row in the table of [Summary Table of Vocabulary Usage Restriction](#) does not apply. Special exception: **other** may only be used in Condition Cells (section 4 of the Sections of Rulesheet that correlate with usage restrictions) because it is a non-specific value used in comparisons.

RULESHEET EXAMPLE

The following Rulesheet uses **other** to test the value of `decimal1`. If `decimal1` has any value *other* than null, `boolean1` is assigned the value of `false`.

Conditions	0	1	2
a Entity1.decimal1		null	other
Actions			
Post Message(s)			
A Entity1.boolean1		T	F
B			
Overrides			
Rule Statements Rule Messages Properties			
Ref	ID	Post	Alias
1		Warning	Entity1
2		Info	Entity1
Text			
1		If decimal1 has the value of null, then assign boolean1 the value of true	
2		If decimal1 has any value other than null (any number), then assign boolean1 a value of false	

SAMPLE TEST

A sample Ruletest provides three examples of decimal1. Ruletest Input and Output panels are shown below:

Input		Output	
Entity1 [1]	decimal1 [0.000000]	Entity1 [1]	boolean1 [false]
Entity1 [2]	decimal1	Entity1 [2]	boolean1 [true]
Entity1 [3]	decimal1 [3.450000]	Entity1 [3]	boolean1 [false]
Rule Statements Rule Messages Properties			
Severity	Message	Entity	
Warning	If decimal1 has the value of null, then assign boolean1 the value of true	Entity1[2]	
Info	If decimal1 has any value other than null (any number), then assign boolean1 a value of false	Entity1[1]	
Info	If decimal1 has any value other than null (any number), then assign boolean1 a value of false	Entity1[3]	

Or

SYNTAX

<Expression1> or <Expression2> or

OR may also be used with [->forAll](#) and [->exists](#) expressions

DESCRIPTION

Returns a value of true if either <Expression1> or <Expression2> evaluates to true. When used between two or more expressions in the Preconditions section, creates a compound filter for the Rulesheet that follows. See *Rule Modeling Guide* for details on using Preconditions as filters. **OR** is not available in the Conditions section because the logical **OR** construction is implemented using multiple Columns in the decision table, or by value sets in Conditions Cells.

USAGE RESTRICTIONS

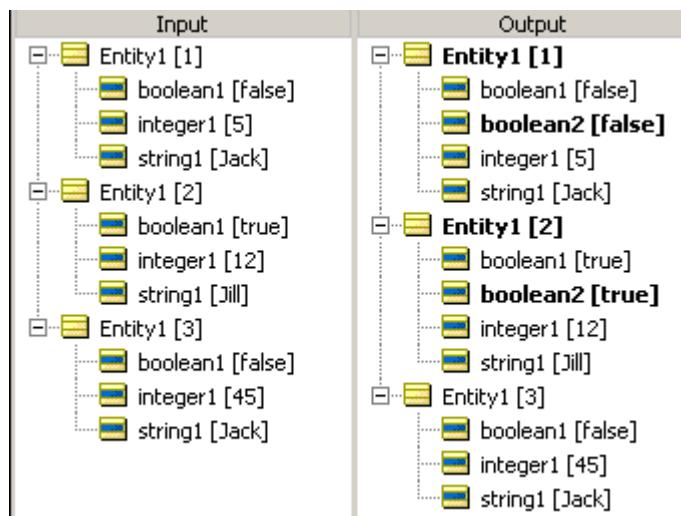
The Literals row in the table of Sections of Rulesheet that correlate with usage restrictions does not apply. Special exception: **or** may only be used in the Filters section of the Rulesheet to join 2 or more expressions, as shown above, or within **->forAll** and **->exists** expressions as described in those sections.

RULESHEET EXAMPLE

The following Rulesheet uses **or** to test the value of `integer1`, `boolean1`, and `string1` to set the value of `boolean2`

SAMPLE TEST

A sample Ruletest provides three examples. Input and Output panels are shown below:



Remove element

SYNTAX

```
<Entity>.remove
<Collection>.remove
```

DESCRIPTION

Removes <Entity> or removes elements from <Collection> and deletes it/them. If removing from a collection, then using a unique alias to represent the collection is optional since .remove is not a collection operator. If any elements in <Collection> have one-to-many associations with other entities, then those entities will also be deleted.

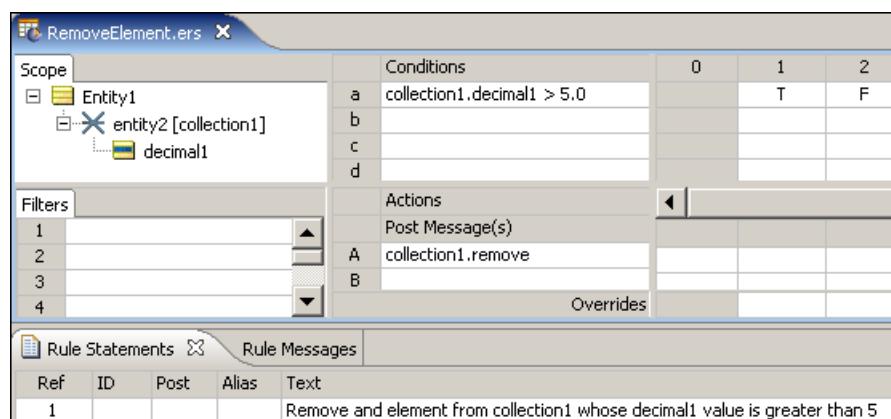
When using .remove to delete elements of a collection, lower-level associated entities will be removed. For example of this behavior, see [example 2](#) below.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) does not apply. Special exceptions: .remove may only be used in Action Rows (section 5 in [Sections of Rulesheet that correlate with usage restrictions](#)).

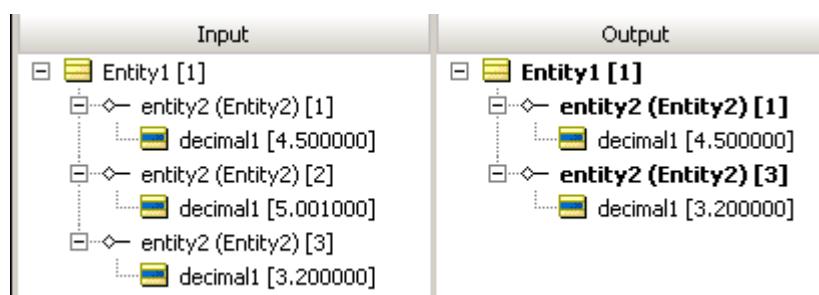
RULESHEET EXAMPLE

This Rulesheet uses the operator to remove elements from collection1 whose decimal1 value is greater than 5. Note the *optional* use of unique alias collection1 to represent the collection of Entity2 elements associated with Entity1.



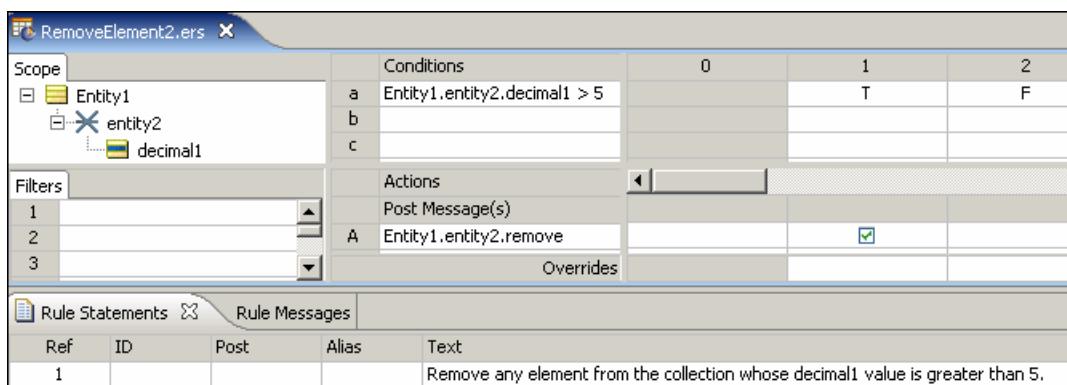
SAMPLE TEST

A sample Ruletest provides a collection with two elements. The illustration shows Ruletest Input and Output panels



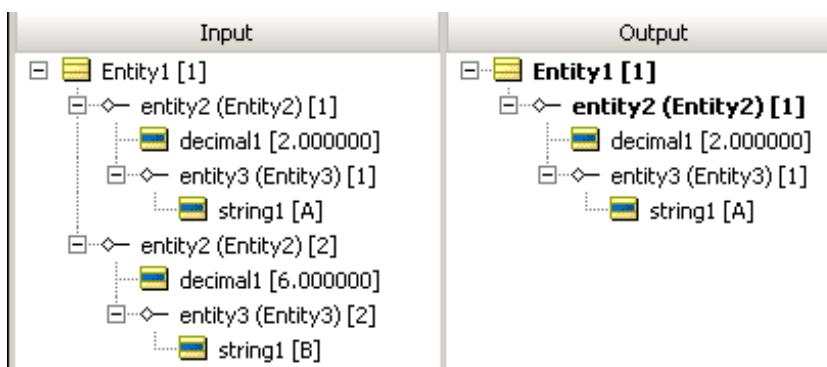
RULESHEET EXAMPLE 2

This Rulesheet uses the operator to remove elements from Entity1.entity2 whose decimal1 value is greater than 5. Note no unique alias has been used to represent the collection of Entity2 elements associated with Entity1.



SAMPLE RULETEST 2

A sample Ruletest provides an Entity1 with two entity2, each of which has an entity3 child of its own. The illustration shows Ruletest Input and Output panels. Note that when an entity2 is removed, its associated entity3 is also removed.



Replace element(s)

SYNTAX

```
<Collection1> = <Collection2>
<Collection> = <Entity>
```

DESCRIPTION

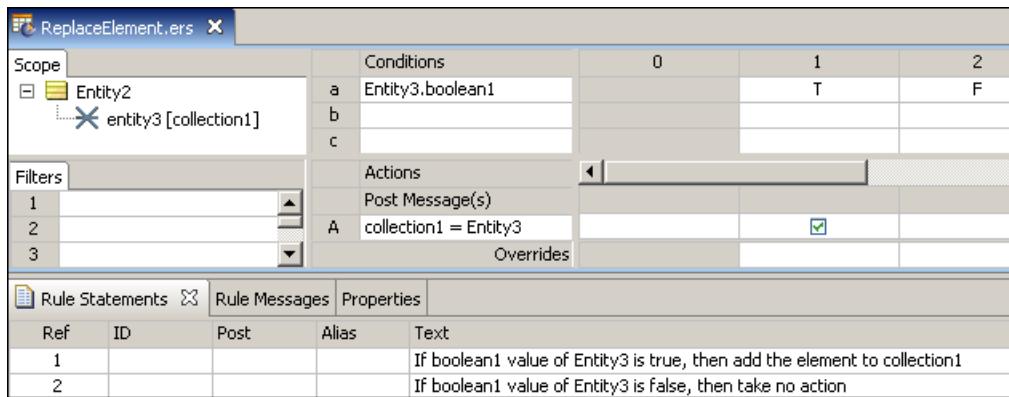
Replaces all elements in <Collection1> with the elements in <Collection2>, provided the association between the two is permitted by the Business Vocabulary. In the second syntax, <Entity> is associated with <Collection>, replacing the <Entity> already associated, when the association between the two is “one-to-one” in the Business Vocabulary. All collections must be expressed as unique aliases.

USAGE RESTRICTIONS

The Operators row in the table of [Summary Table of Vocabulary Usage Restriction](#) does not apply. Special exceptions: **replace elements** may only be used in Action Rows (section 5 in [Sections of Rulesheet that correlate with usage restrictions](#)).

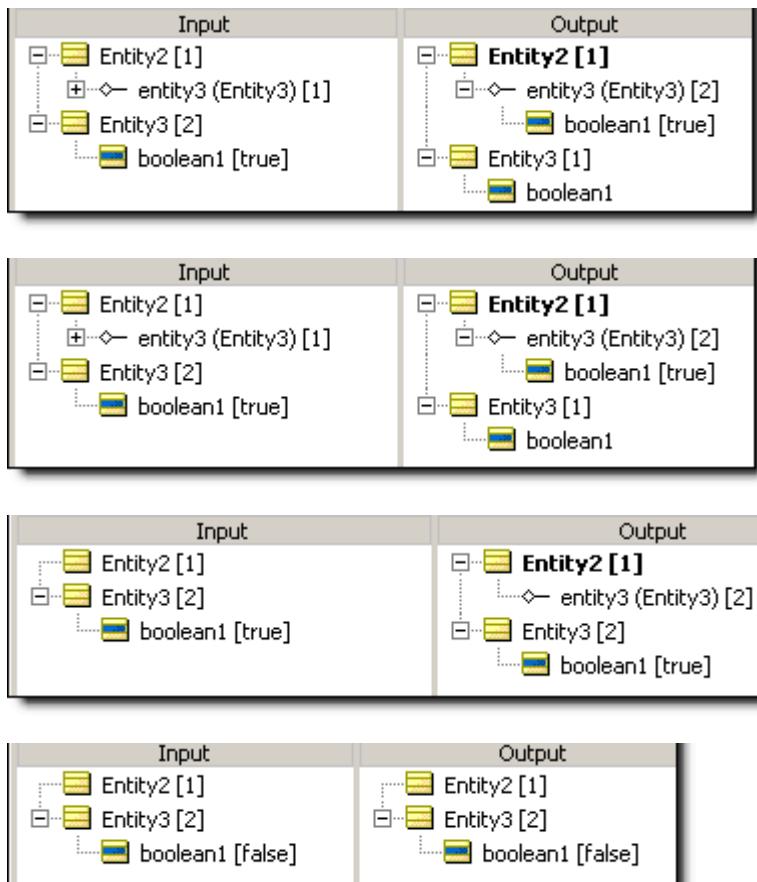
RULESHEET EXAMPLE

This sample Rulesheet uses the **replace element** operator to add Entity3 to collection1 if its boolean1 value is true. Note the use of unique alias collection1 to represent the collection of Entity3 elements associated with Entity2. Recall from the generic Business Vocabulary, shown in [Vocabulary used in this Language Guide](#) on page 12, that the association between Entity2 and Entity3 has a cardinality of “one-to-one”. This means that if multiple Entity3 are present, only one will be added to collection1.



SAMPLE TEST

Four sample tests provide scenarios of two elements which share a one-to-one association. Input and Output panels are illustrated below:



Round

SYNTAX

<Decimal>.round(<Integer>)

DESCRIPTION

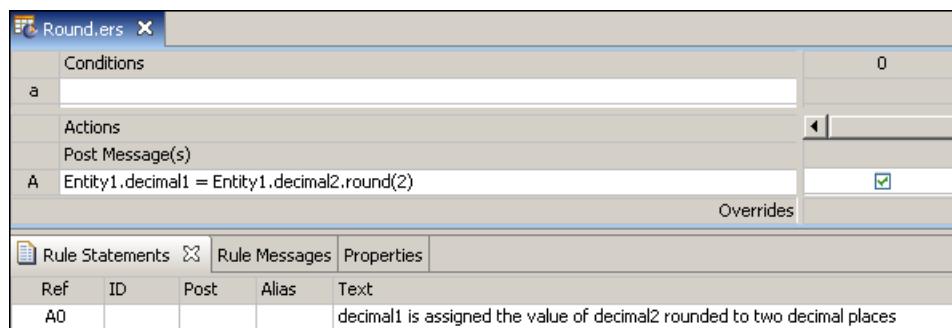
Rounds <Decimal> to the number of decimal places specified by <Integer>. Standard rounding conventions apply, meaning numbers ending with significant digits of 5 or more round up and numbers ending with significant digits less than 5 round down. <Integer> is optional – if no parameter is specified, then <Decimal> rounds to the nearest whole number of type Decimal.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

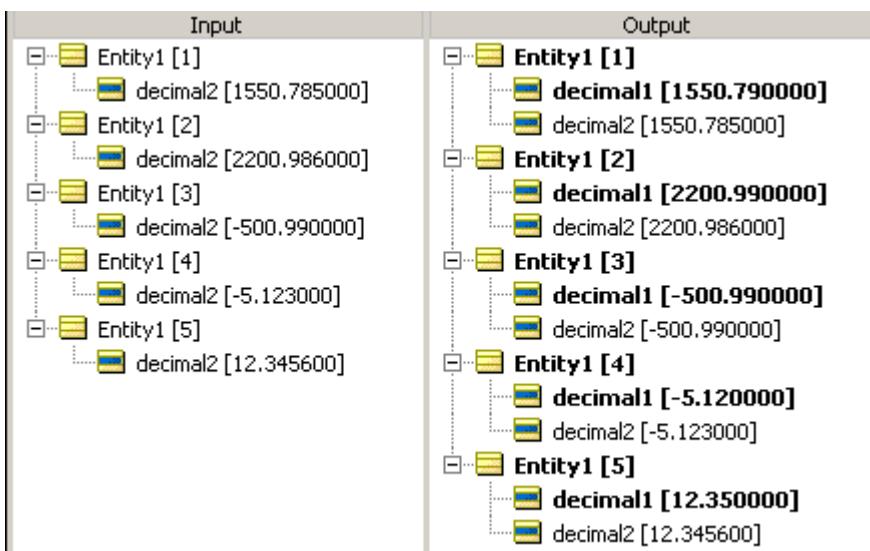
RULESHEET EXAMPLE

The following Rulesheet uses **.round** to round the value of decimal2 to the 2nd decimal place, and assigns it to decimal1.



SAMPLE TEST

A sample Ruletet provides results for five examples of decimal2.



Second

SYNTAX

<DateTime>.sec

<Time>.sec

DESCRIPTION

Returns the seconds portion of <DateTime> or <Time>. The returned value is an Integer between 0 and 59.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses the .sec function to evaluate `dateTime1`, return the seconds value, and assign it to `integer1`.

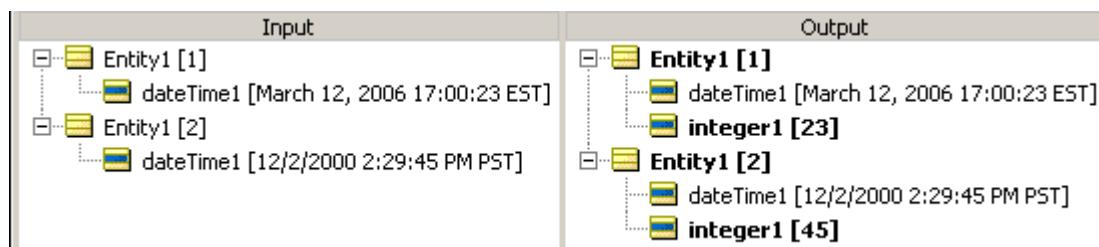
Conditions		0	1
a			
b			

Actions		
Post Message(s)		
A	Entity1.integer1 = Entity1.dateTime1.sec	<input checked="" type="checkbox"/>
B		

Overrides				
Rule Statements	Rule Messages	Problems		
Ref	ID	Post	Alias	Text
A0				integer1 is equal to the seconds portion of dateTime1

SAMPLE TEST

A sample Ruletest provides results for two examples of `dateTime1`.



Seconds between

SYNTAX

```
<DateTime1>.secsBetween(<DateTime2>)
<Time1>.secsBetween(<Time>)
```

DESCRIPTION

Returns the Integer number of seconds between DateTimes or between Times. The number of milliseconds in `<DateTime1>` is subtracted from that in `<DateTime2>`, and the result divided by 1000 (the number of milliseconds in a second). The result is truncated. This function returns a positive number if `<DateTime2>` is later than `<DateTime1>`.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses `.secsBetween` to determine the number of seconds that have elapsed between `dateTime1` and `dateTime2`, compare it to the Values set, and assign a value to `string1`.

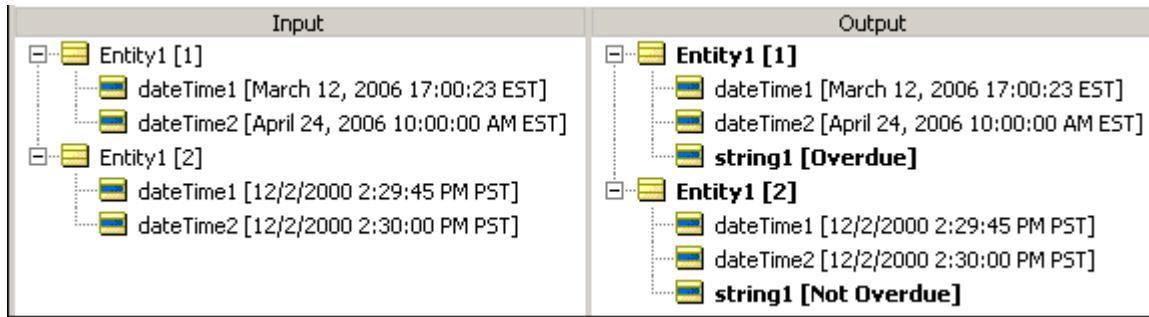
Conditions	0	1	2
a Entity1.dateTime1.secsBetween(Entity1.dateTime2)		<=60	>60
b			
c			
d			

Actions		
Post Message(s)		
A Entity1.string1	'Not Overdue'	'Overdue'
B		

Overrides				
Rule Statements	Rule Messages	Properties		
Ref	ID	Post	Alias	Text
1				If 60 or fewer seconds have elapsed between dateTime1 and dateTime2, then Entity1 is Not Overdue
2				If more than 60 seconds have elapsed between dateTime1 and dateTime2, then Entity1 is Overdue

SAMPLE TEST

A sample Ruletest provides `dateTime1` and `dateTime2` for two examples. Input and Output panels are shown below.



Size of string

SYNTAX

`<String>.size`

DESCRIPTION

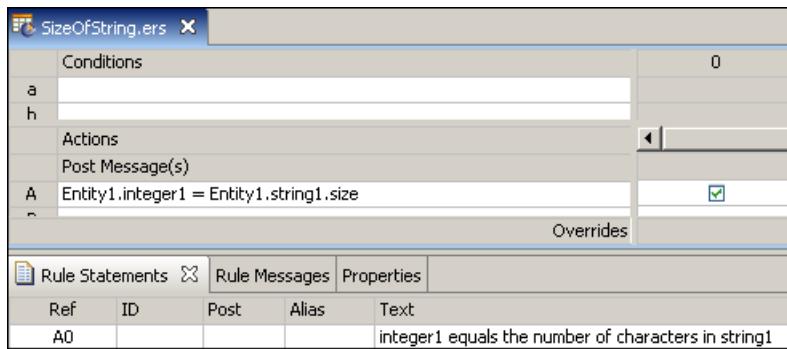
Returns the Integer number of characters in `<String>`. All characters, numbers, symbols, and punctuation marks are counted, including spaces before, within, and after words.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

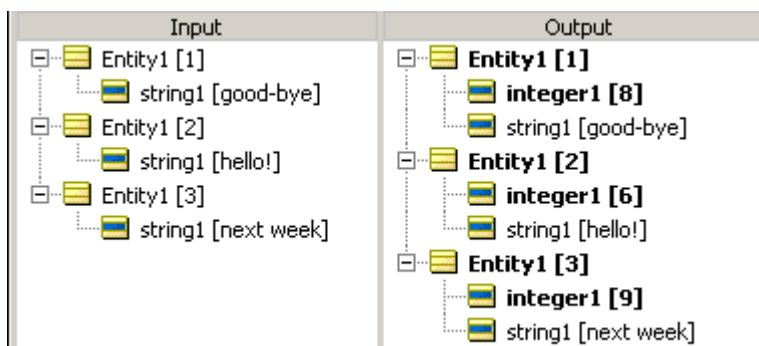
RULESHEET EXAMPLE

The following Rulesheet uses the `.size` function to determine the length of `string1` and assign it to `integer1`.



SAMPLE TEST

A sample Ruletest provides three examples. Input and Output panels are shown below:



Size of collection

SYNTAX

<Collection> ->size

DESCRIPTION

Returns the Integer number of elements in <Collection>. <Collection> must be expressed as a unique alias.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

This sample Rulesheet uses **->size** to count the number of elements in `collection1`, and assign a value to `boolean2`. Note the use of unique alias `collection1` to represent the collection of `Entity2` associated with `Entity1`.

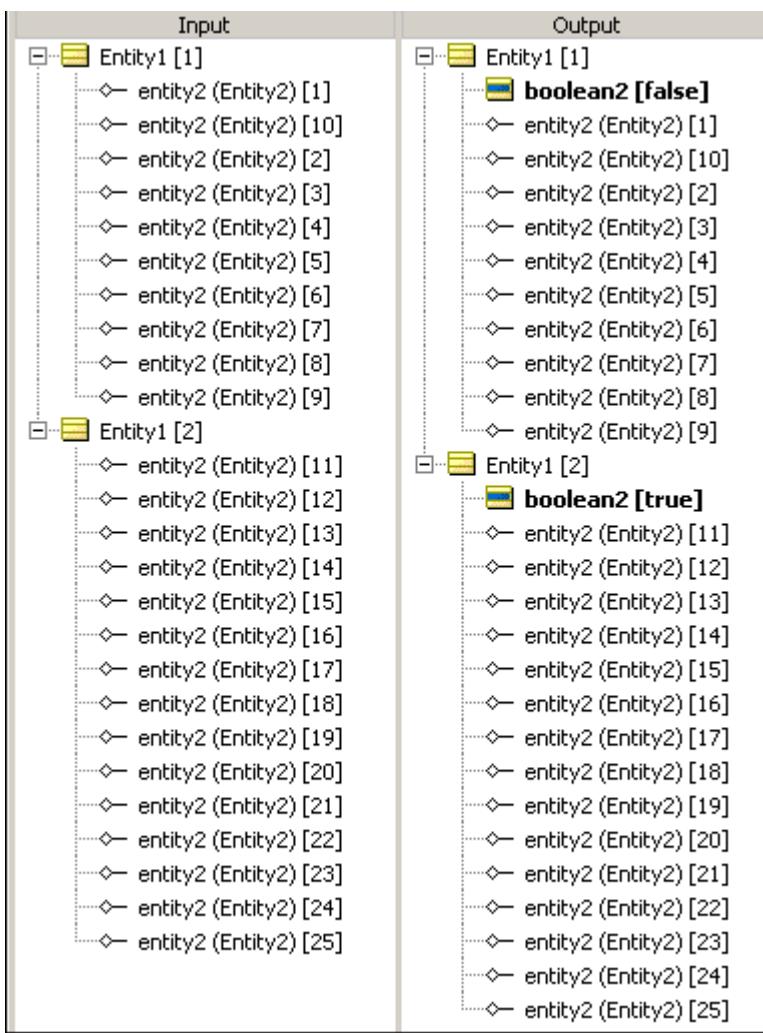
Scope		Conditions		
		0	1	2
a	collection1 -> size		<12	>=12
b				
c				
d				

Filters		Actions		
1	2	Post Message(s)	A	B
2	3		Entity1.boolean2	F T
3	4			

Rule Statements				
Ref	ID	Post	Alias	Text
1				If there are less than 12 elements in collection1, then no discount is applied (boolean2 is false)
2				If there are 12 or more elements in collection1, then a discount is applied (boolean2 is true)

SAMPLE TEST

A sample Ruletest provides three examples of `collection1`. Input and Output panels are shown below.



Sorted by

SYNTAX

<Collection> ->sortedBy(<Attribute2>) -> **sequence operator.**<Attribute1>

DESCRIPTION

Sequences the elements of <Collection> in ascending order, using the value of <Attribute2> as the index, and returns the <Attribute1> value of the element in the sequence position determined by the sequence operator. A sequence must be created before any sequence operator (**->first**, **->last**, or **->at**) is used to identify a particular element. <Attribute1> and <Attribute2> must be attributes of <Collection>.

<Attribute2> may be any data type except Boolean. Strings are sorted according to character precedence – see [Character precedence: Unicode and Java Collator](#) on page 181. <Collection> must be expressed as a unique alias.

See [Advanced Collection Syntax](#) and special [statement block](#) syntax for more examples of usage.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE 1 - USED IN A CONDITION

This sample Rulesheet uses `->sortedBy` in a conditional expression to create an ascending sequence from collection with `decimal1` as the index. `->first.string1` is used to return the value of the `string1` attribute of the first element of the sequence. If the value of `string1` is `Joe`, then `boolean1` attribute of Entity1 is assigned the value of `true`.

The screenshot shows the Rulesheet editor window titled "SortedBy.ers". The left pane displays the scope tree with nodes: Entity1, boolean1, entity2 [collection1], decimal1, and string1. The middle pane contains a "Conditions" table:

	Conditions	0	1	2
a	<code>collection1.string1 -> sortedBy(decimal1) -> first</code>		'Joe'	not 'Joe'
b				
c				
d				
e				

The right pane shows the "Actions" section with a "Post Message(s)" entry and a truth table for Entity1.boolean1:

	A	T	F
A	Entity1.boolean1		
B			
C			

Below the editor are two tabs: "Rule Statements" and "Rule Messages". The "Rule Statements" tab lists two entries:

Ref	ID	Post	Alias	Text
1				If the <code>string1</code> value of the first element in <code>collection1</code> , sequenced in ascending order by <code>decimal1</code> , is equal to <code>Joe</code> , then <code>boolean1</code> = <code>true</code>
2				If the <code>string1</code> value of the first element in <code>collection1</code> , sequenced in ascending order by <code>decimal1</code> , is not equal to <code>Joe</code> , then <code>boolean1</code> = <code>false</code>

SAMPLE RULETEST 1

A sample Ruletest provides a collection of three elements, each with a `decimal1` and `string1` value. Input and Output panels are shown below.

The screenshot shows the Ruletest editor with two panels: "Input" and "Output".

Input Panel: Displays a collection of three Entity1 objects. Each Entity1 has an associated entity2 object. The entity2 objects have `decimal1` and `string1` attributes. The values are:

- Entity1[1]: entity2[1] (decimal1: 2.500000, string1: Joe)
- Entity1[2]: entity2[2] (decimal1: 5.800000, string1: Mary)
- Entity1[3]: entity2[3] (decimal1: 3.300000, string1: Sue)

Output Panel: Displays the same collection after processing. The `boolean1` attribute is added to Entity1[1] with the value `true`. The other Entity1 objects remain unchanged.

RULESHEET EXAMPLE 2 – USED IN AN ACTION

This sample Rulesheet uses `->sortedBy` in an action expression to create an ascending sequence from collection with `decimal1` as the index. `->first.string1` is used to return the value of the `string1` attribute of the first element of the sequence. The value of `string1` is assigned the value of `Joe` if `boolean1` attribute of Entity1 is `true`, if `false` it is assigned the value of `Mary`.

Ref	ID	Post	Alias	Text
1				If Entity1.boolean1 is true, string1 value in the first element of collection1, sequenced in ascending order by decimal1, is equal to Joe
2				If Entity1.boolean1 is false, string1 value in the first element of collection1, sequenced in ascending order by decimal1, is equal to Mary

SAMPLE RULETEST 2

A sample Ruletest provides a collection of three elements, each with a decimal1 and string1 value. Input and Output panels are shown below.

Sorted by descending

SYNTAX

<Collection> ->sortedByDesc(<Attribute2>) -> sequence operator.<Attribute1>

DESCRIPTION

Sequences the elements of <Collection> in descending order, using the value of <Attribute2> as the index, and returns the <Attribute1> value of the element in the sequence position determined by the sequence operator. A sequence must be created before any sequence operator (*->first*, *->last*, or *->at*) is used to identify a particular element. <Attribute1> and <Attribute2> must be attributes of <Collection>.

<Attribute2> may be any data type except Boolean. Strings are sorted according to their ISO character precedence – see [Character precedence: Unicode and Java Collator](#) on page 181. <Collection> must be expressed as a unique alias.

See [Advanced Collection Syntax](#) and special [statement block](#) syntax for more examples of usage.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE 1 - USED IN A CONDITION

This sample Rulesheet uses `-> sortedByDesc` in a conditional expression to create an descending sequence from collection with `decimal1` as the index. `->first.string1` is used to return the value of the `string1` attribute of the first element of the sequence. If the value of `string1` is `Joe`, then `boolean1` attribute of Entity1 is assigned the value of `true`.

The screenshot shows the Rulesheet editor interface with the title "SortedByDescending.ers". The left pane displays the scope tree with nodes: Entity1, boolean1, entity2 [collection1], decimal1, and string1. The middle pane contains a "Conditions" table:

	Conditions	0	1	2
a	<code>collection1 -> sortedByDesc(decimal1) -> first.string1</code>		'Joe'	not 'Joe'
b				
c				

Below the conditions table is an "Actions" section with "Post Message(s)". The "Overrides" table shows a row for Entity1.boolean1 with columns T and F. The bottom pane shows two rule statements:

Ref	ID	Post	Alias	Text
1				If Entity1.boolean1 is true, string1 value in the first element of collection1, sequenced in descending order by decimal1, is equal to Joe
2				If Entity1.boolean1 is false, string1 value in the first element of collection1, sequenced in descending order by decimal1, is equal to Mary

SAMPLE RULETEST 1

A sample Ruletest provides a collection of three elements, each with a `decimal1` value. Input and Output panels are shown below.

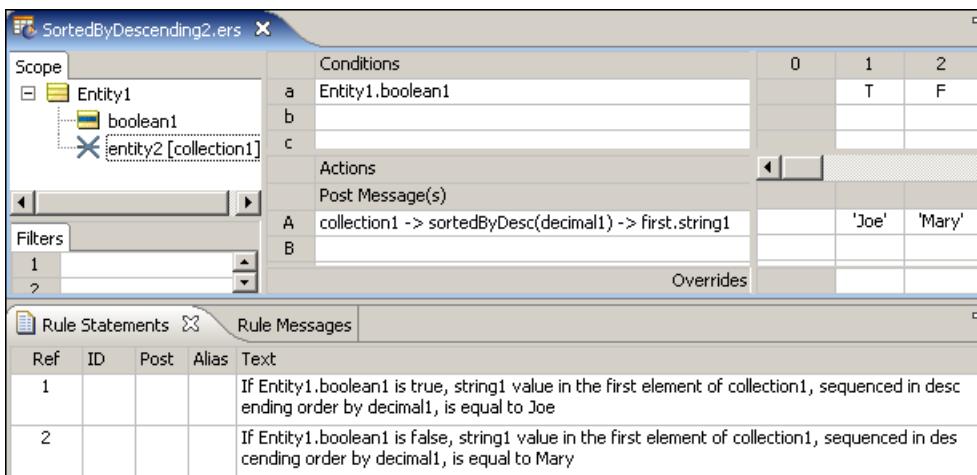
The screenshot shows the Ruletest editor interface with two panels: "Input" and "Output".

Input: Shows a collection of three elements, each containing an Entity1 object. Entity1[1] has decimal1 = 2.500000 and string1 = Joe. Entity1[2] has decimal1 = 5.800000 and string1 = Mary. Entity1[3] has decimal1 = 3.300000 and string1 = Sue.

Output: Shows the same collection after processing. Entity1[1] now has boolean1 = false, decimal1 = 2.500000, and string1 = Joe. Entity1[2] has boolean1 = false, decimal1 = 5.800000, and string1 = Mary. Entity1[3] has boolean1 = false, decimal1 = 3.300000, and string1 = Sue.

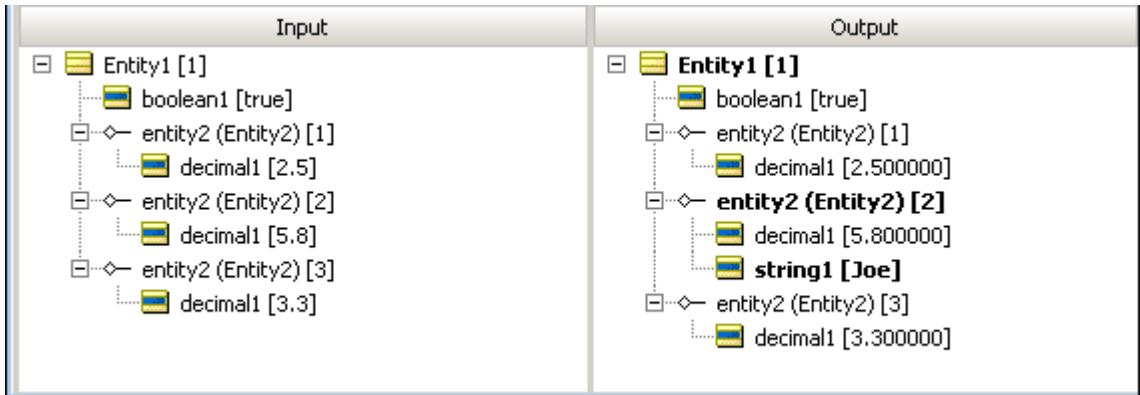
RULESHEET EXAMPLE 2 – USED IN AN ACTION

This sample Rulesheet uses `sortedByDesc` in an action expression to create an descending sequence from collection with `decimal1` as the index. `->first.string1` is used to return the value of the `string1` attribute of the first element of the sequence. The value of `string1` is assigned the value of `Joe` if `boolean1` attribute of Entity1 is `true`, if `false` it is assigned the value of `Mary`.



SAMPLE RULETEST 2

A sample Ruletest provides a collection of three elements, each with a decimal1 value. Input and Output panels are shown below.



Starts with

SYNTAX

```
<String1>.startsWith(<String2>)
```

DESCRIPTION

Returns a value of true if <String1> begins with the characters specified in <String2>. Comparisons are case-sensitive.

USAGE RESTRICTIONS

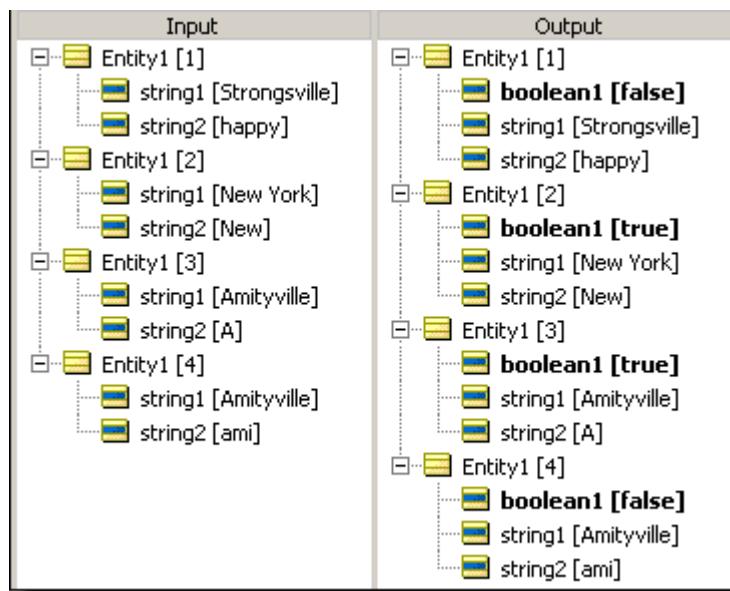
The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses `.startsWith` to evaluate whether `string1` begins with the value of `string2` and assigns a different value to `boolean1` for each outcome.

SAMPLE TEST

A sample Ruletest provides `string1` and `string2` values for four examples. Input and Output panels are shown below.



SubSequence

SYNTAX

`<Sequence> ->subSequence(integer1, integer2)`

DESCRIPTION

Returns a Sequence containing all elements of <Sequence> between the positions *integer1* and *integer2*. Another operator, such as `->sortedBy` or `->sortedByDesc`, must be used to transform a <Collection> into a <Sequence> before `->subSequence` may be used. <Sequence> must be expressed as a unique alias. See [Advanced Collection Syntax](#) for more examples of usage.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

Both integer values must be provided, separated by a comma. If *integer1* is larger than *integer2*, there are no results. When *integer1* is beyond the count of the collection, there are no results. When *integer2* is beyond the count of the collection, all data from *integer1* to the last entity is in the results collection. There are no results when both integers extend beyond the number of elements in the collection.

RULESHEET EXAMPLE

This sample Rulesheet uses `->subSequence(3,4)` to identify the 'middle' two elements of the sequence that resulted from the `sortedBy` operation.

Scope	Conditions	0
	a	
	b	
	Actions	
	Post Message(s)	
	A collection3=collection2->sortedBy(collection2.decimal1)->subSequence(3,4)	<input checked="" type="checkbox"/>
	B	

SAMPLE RULETEST

A sample Ruletest provides a collection of five elements, each with a `decimal1` value. Input and Output panels are shown below.

Input	Output
<ul style="list-style-type: none"> Entity_1 [1] <ul style="list-style-type: none"> entity_2 (Entity_2) [1] decimal1 [500.000000] entity_2 (Entity_2) [2] decimal1 [800.000000] entity_2 (Entity_2) [3] decimal1 [700.000000] entity_2 (Entity_2) [4] decimal1 [100.000000] entity_2 (Entity_2) [5] decimal1 [600.000000] Entity_3 [1] 	<ul style="list-style-type: none"> Entity_1 [1] <ul style="list-style-type: none"> entity_2 (Entity_2) [1] decimal1 [500.000000] entity_2 (Entity_2) [2] decimal1 [800.000000] entity_2 (Entity_2) [3] decimal1 [700.000000] entity_2 (Entity_2) [4] decimal1 [100.000000] entity_2 (Entity_2) [5] decimal1 [600.000000] Entity_3 [1] <ul style="list-style-type: none"> entity_2 (Entity_2) [3] entity_2 (Entity_2) [5]

Note: The selected entities and their values are highlighted to improve readability.

Substring

SYNTAX

<String>.substring(<Integer1>, <Integer2>)

DESCRIPTION

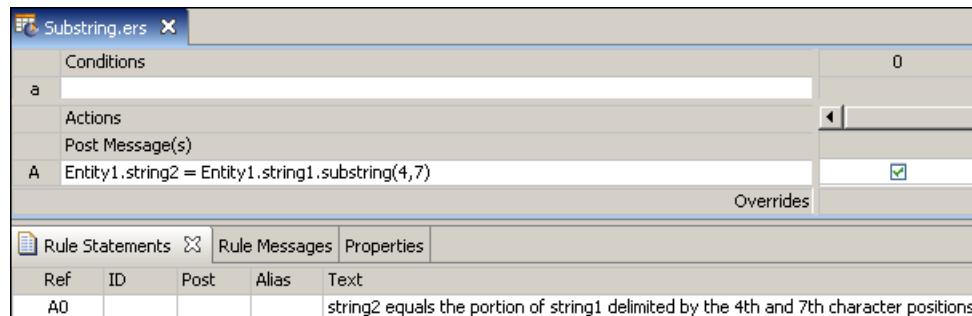
Returns the portion of <String> beginning with the character in position <Integer1> and ending with the character in position <Integer2>. The number of characters in <String> must be at least equal to <Integer2>, otherwise an error will be produced. Both <Integer1> and <Integer2> must be positive integers, and <Integer2> must be greater than <Integer1>.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

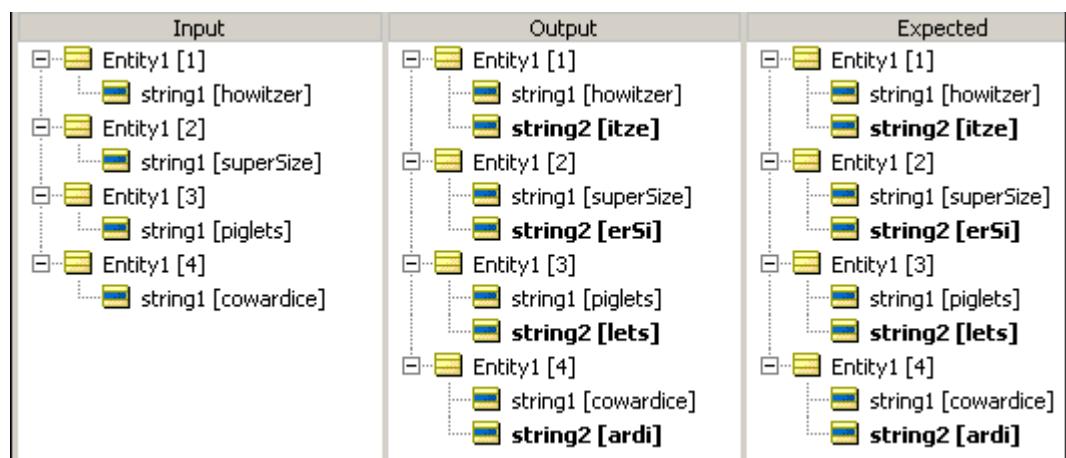
RULESHEET EXAMPLE

This sample Rulesheet uses **.substring** to return those characters of string1 between positions 4 and 7 (inclusive), and assign the resulting value to string2.



SAMPLE RULETEST 1

A sample Ruletest provides string1 values for four examples. Input and Output panels are shown below.



SAMPLE RULETEST 2

Another sample Ruletest shows `string1` values with two examples having insufficient character strings. The resulting error message is illustrated below:

The screenshot shows the Progress Corticon Studio interface with two main windows:

- Ruletest Table:** A table titled "untitled_1" under the path "/Log/Log Operator/Log.ers". It has three columns: Input, Output, and Expected. The Input column contains three entries: Entity_1 [1], Entity_1 [2], and Entity_1 [3]. The Output column contains three entries: Entity_1 [1] (with a warning icon), Entity_1 [2] (with a warning icon), and Entity_1 [3] (with a warning icon). The Expected column is empty.
- Rule Messages Table:** A table titled "Rule Statements" with tabs for "Rule Statements" and "Rule Messages". It has three columns: Severity, Message, and Entity. The table is currently empty.

Subtract

SYNTAX

`<Number1> - <Number2>`

DESCRIPTION

Subtracts the value of `<Number2>` from that of `<Number1>`. The resulting data type is the more expansive of those of `<Number1>` and `<Number2>`.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

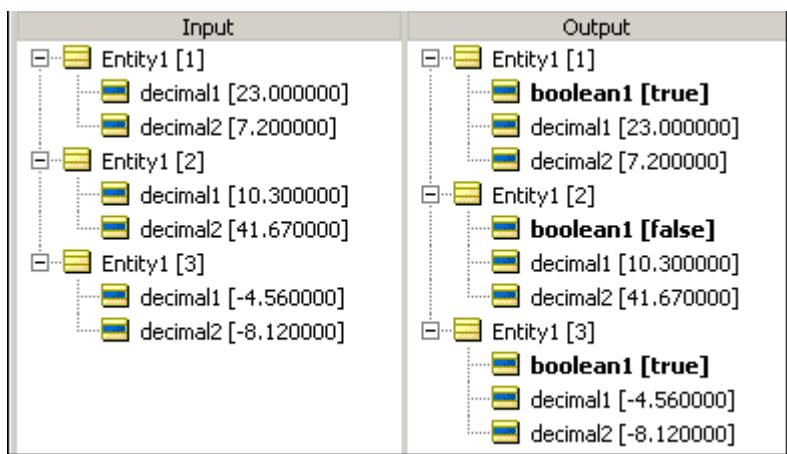
This sample Rulesheet uses **subtract** to reduce the value of `decimal1` by `decimal2`, compare the resulting value to zero, and assign a value to `boolean1`

Conditions	0	1	2
a Entity1.decimal1 - Entity1.decimal2 > 0		T	F
Actions			
Post Message(s)			
A Entity1.boolean1		T	F
B			
Overrides			

Rule Statements				
Ref	ID	Post	Alias	Text
1				decimal1 is greater than decimal2
2				decimal2 is greater than decimal1

SAMPLE TEST

A Ruletest provides three examples of decimal1 and decimal2. Input and Output panels are shown below.



Sum

SYNTAX

<Collection.attribute> ->sum

DESCRIPTION

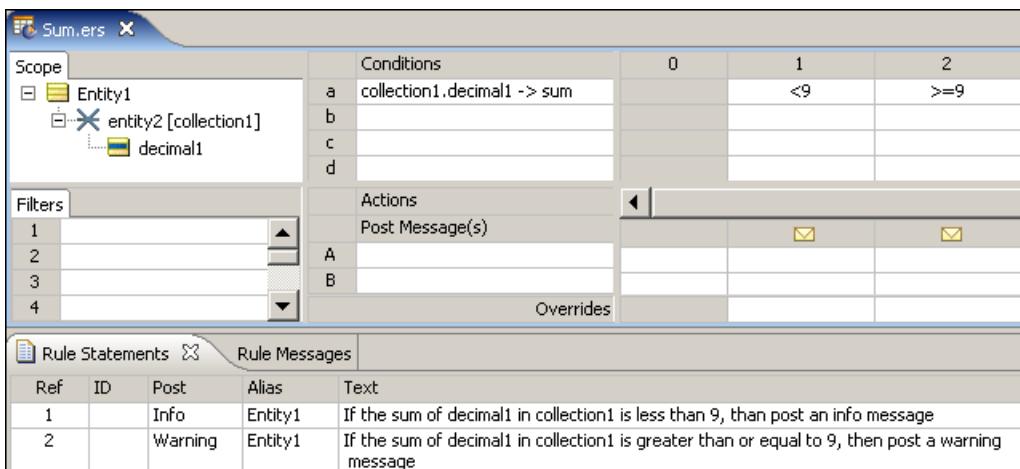
Sums the values of the specified <attribute> for all elements in <Collection>. <attribute> must be a numeric data type. <Collection> must be expressed as a unique alias.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

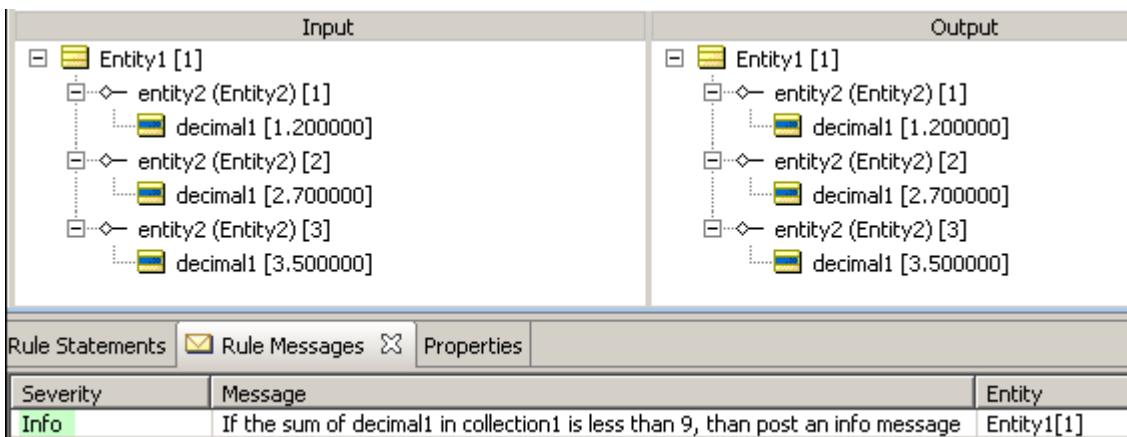
RULESHEET EXAMPLE

This Rulesheet uses the **->sum** function to add all decimal1 attributes within collection1. Note the use of unique alias collection1 to represent the collection of Entity2 associated with Entity1



SAMPLE TEST

A sample Ruletest provides 3 elements in `collection1`. Input and Output panels are shown below.



Today

SYNTAX

`today`

DESCRIPTION

Returns the current system date when the rule is executed. This Date Only value is assigned the first time `today` is used in a Decision Service, then remains constant until the Decision Service finishes execution, regardless of how many additional times it is used. This means that every rule in a Rule Set using `today` will use the same Date Only value. No time portion is assigned.

USAGE RESTRICTIONS

The Literals row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses **today** to determine how many days have elapsed between today and `dateTime1`, and assign a value to `string1` based on the result.

	Conditions	0	1	2
a	<code>Entity1.dateOnly1.daysBetween(today) < 5</code>		T	F
b				
c				
d				
e				
	Actions	0	1	2
	Post Message(s)			
A	<code>Entity1.string1</code>		'under 5 days'	'5 days or more'
B				
C				
	Overrides	0	1	2

Rule Statements				
Ref	ID	Post	Alias	Text
1				If <code>dateOnly1</code> occurred less than 5 days ago, assign <code>string1</code> a value of 'under 5 days'
2				If <code>dateOnly1</code> occurred 5 or more days ago, assign <code>string1</code> a value of '5 days or more'

SAMPLE TEST

A sample Ruletest provides three examples of `dateOnly1`. Assume **today** is equal to Friday, November 23, 2007. Input and Output panels are shown below:

Input	Output
<ul style="list-style-type: none"> Entity1 [1] <ul style="list-style-type: none"> dateOnly1 [11/20/2007] Entity1 [2] <ul style="list-style-type: none"> dateOnly1 [11/19/2007] Entity1 [3] <ul style="list-style-type: none"> dateOnly1 [11/11/2007] 	<ul style="list-style-type: none"> Entity1 [1] <ul style="list-style-type: none"> dateOnly1 [11/20/2007] string1 [under 5 days] Entity1 [2] <ul style="list-style-type: none"> dateOnly1 [11/19/2007] string1 [under 5 days] Entity1 [3] <ul style="list-style-type: none"> dateOnly1 [11/11/2007] string1 [5 days or more]

To date: Casting a dateTime to a date

SYNTAX

`<DateTime>.toDate`

DESCRIPTION

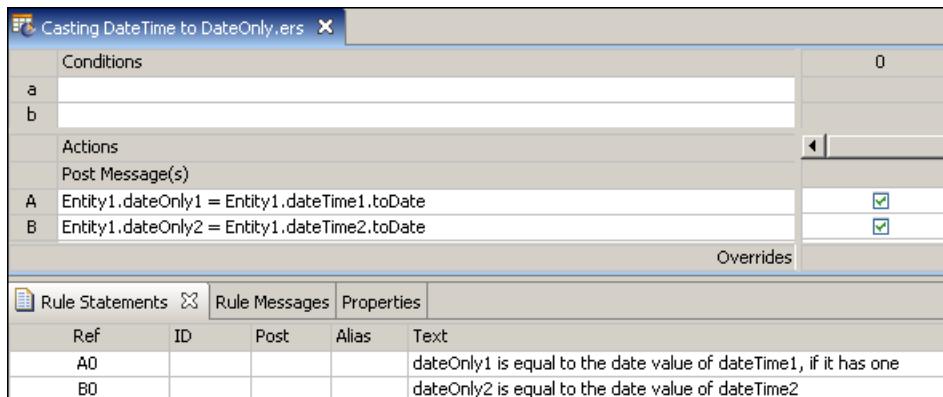
Converts the value in `<DateTime>` to a Date datatype, containing only the date portion of the DateTime. If `<DateTime>` contains no date information, then the system epoch is used.

USAGE RESTRICTIONS

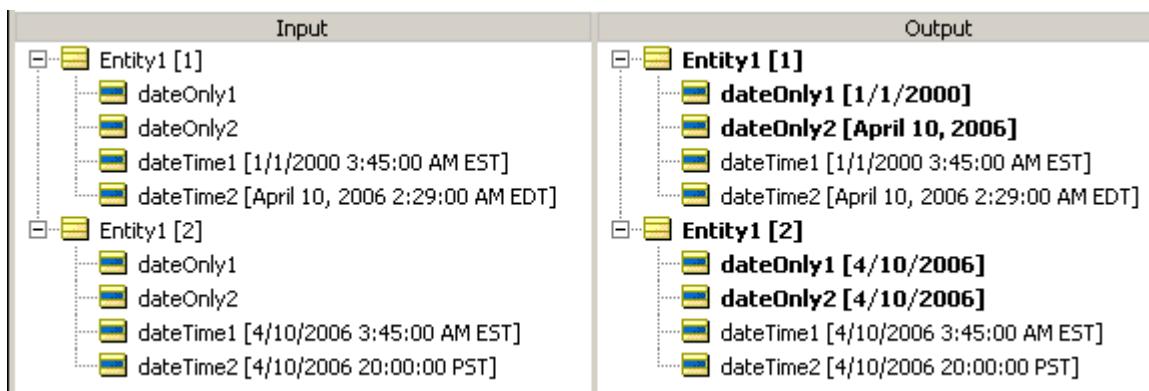
The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses `.toDate` to convert `dateTime1` and `dateTime2` to Date datatypes and assign the values to `dateOnly1` and `dateOnly2`.



SAMPLE TEST



To dateTime: Casting a string to a dateTime

SYNTAX

`<String>.toDateTime`

DESCRIPTION

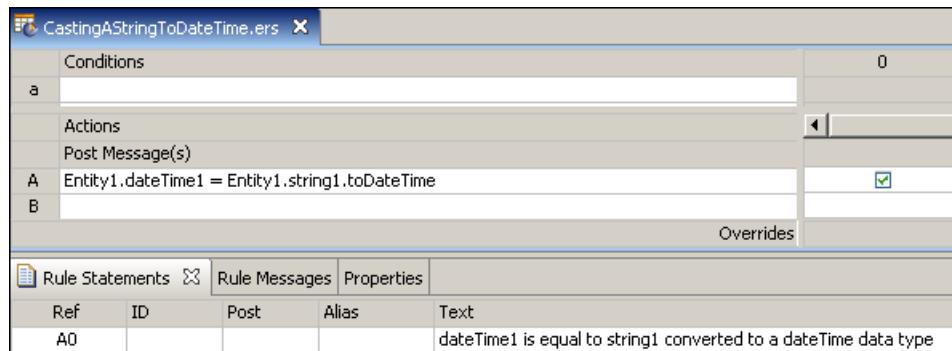
Converts the value in `<String>` to data type `DateTime` ONLY if all characters in `<String>` correspond to a valid Date, Time, or `DateTime` mask (format). For complete details on `DateTime` masks, see *Rule Modeling Guide*.

USAGE RESTRICTIONS

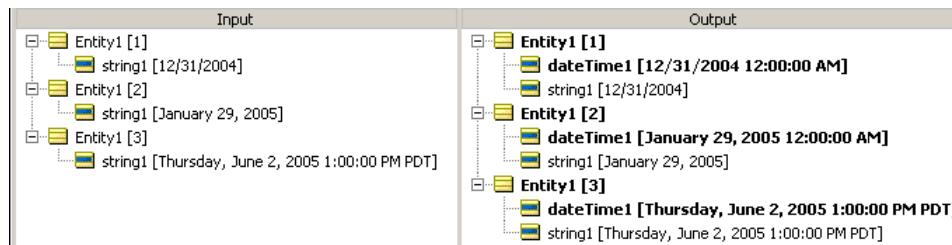
The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses **.toDateTime** to convert `string1` to type DateTime and assign the value to `dateTime1`.



SAMPLE TEST



To date**Time**: Casting a date to a date**Time**

SYNTAX

`<Date>.toDateTime`

DESCRIPTION

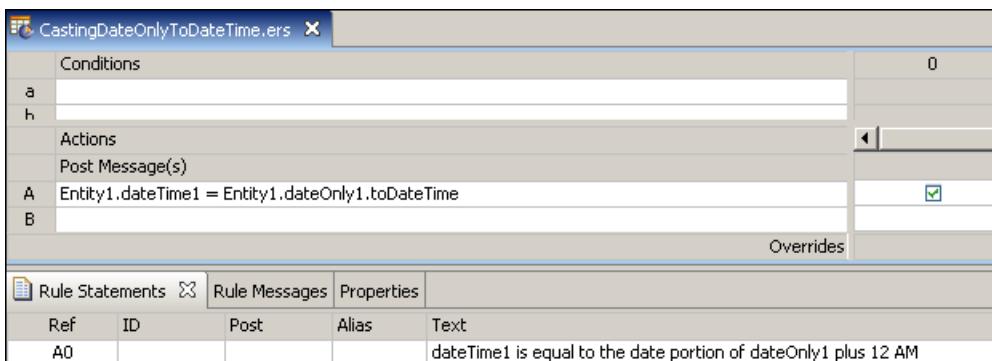
Converts the value in `<Date>` to data type DateTime. The date portion is the same as the `<Date>` value and the time portion is set to `12:00:00 AM` in the current timezone.

USAGE RESTRICTIONS

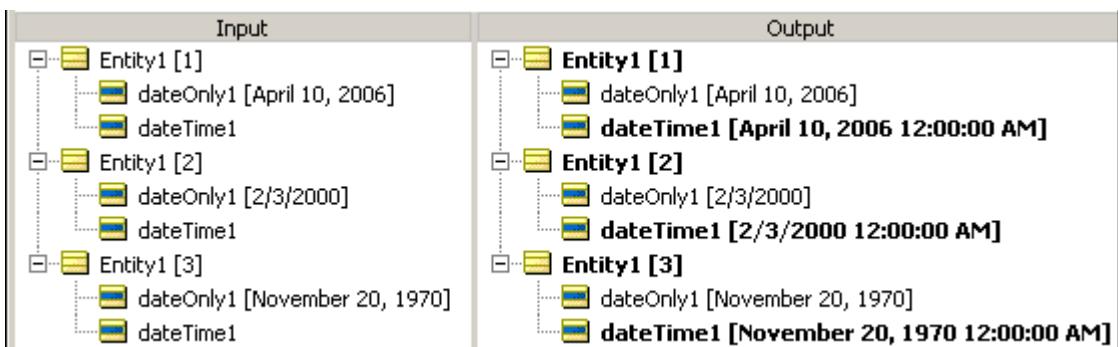
The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses **.toDateTime** to convert `dateOnly1` to type DateTime and assign the value to `dateTime1`.



SAMPLE TEST



To dateTime: Casting a time to a dateTime

SYNTAX

<Time>.toDateTime

DESCRIPTION

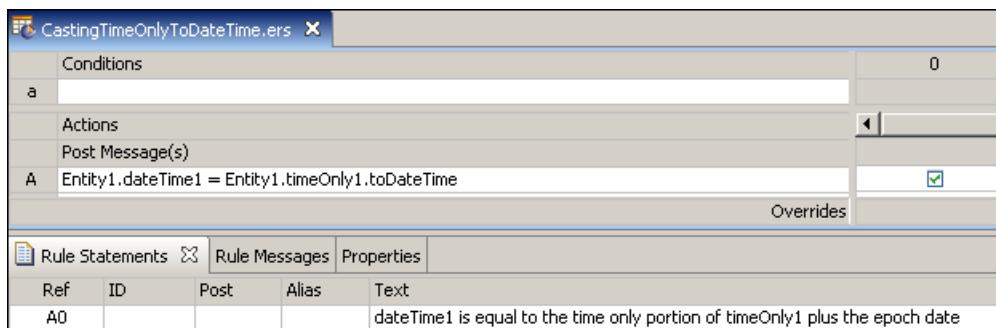
Converts the value in <Time> to data type DateTime ONLY if all characters in <Time> correspond to a valid DateTime mask (format). The time portion is the same as the <Time> value and the date portion is the epoch (see [.toTime](#) operator)

USAGE RESTRICTIONS

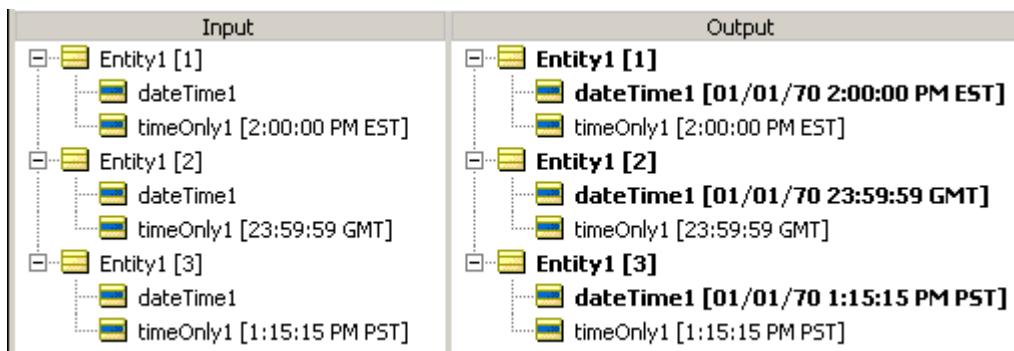
The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses **.toDateTime** to convert `timeOnly1` to type DateTime and assign the value to `dateTime1`.



SAMPLE TEST



To dateTIme: Timezone offset

SYNTAX

<Date>.toDateTime(<String>)

DESCRIPTION

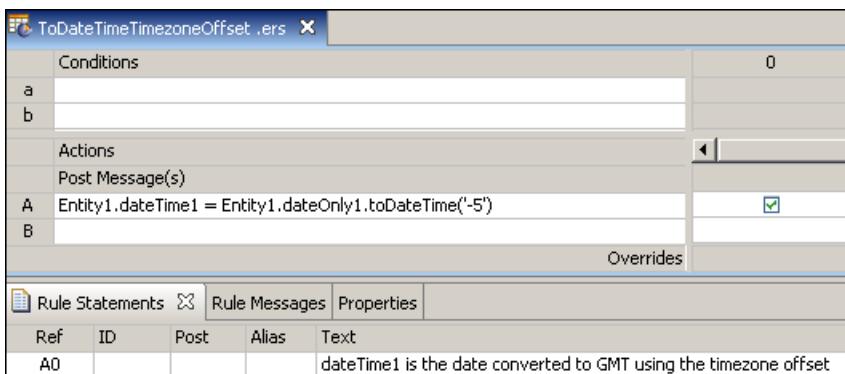
Converts the value in <Date> to data type DateTime ONLY if all characters in <Date> correspond to a valid DateTime mask (format). The date portion is the same as the <Date> value and the time portion is set to 00:00:00 in the timezone specified by <String>, which is the timeZoneOffset. The timeZoneOffset must take the form of a valid timezone offset such as '-08:00', '+03:30', '01:45'

USAGE RESTRICTIONS

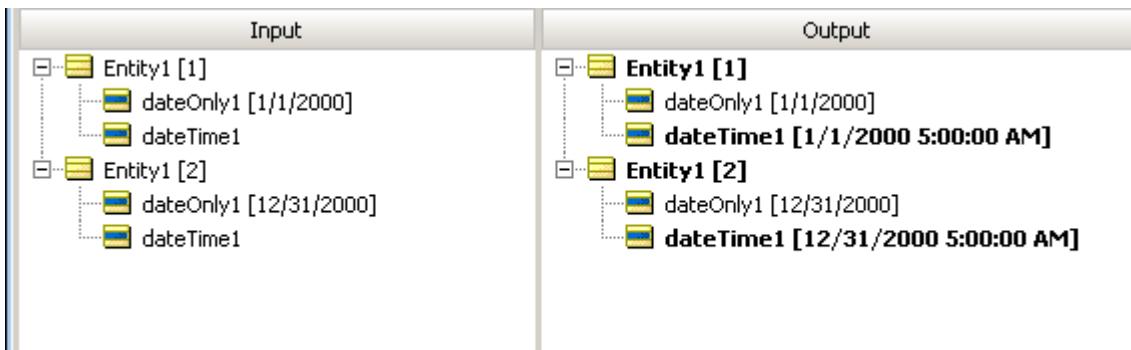
The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses .toDateTime to convert dateOnly1 to type DateTime and assign the value to dateTime1, with a timezone offset of -5.



SAMPLE TEST



To decimal

SYNTAX

```
<Integer>.toDecimal  
<String>.toDecimal
```

DESCRIPTION

Converts the value in <Integer> or all characters in <String> to data type Decimal. Converts a String to Decimal ONLY if all characters in <String> are numeric and contain not more than one decimal point. If any non-numeric characters are present in <String> (other than the single decimal point or a leading minus sign), no value is returned by the function.

Note: Integer values may be assigned directly to Decimal data types without using the **.toDecimal** operator because a Decimal data type is more expansive than an Integer.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses `.toDecimal` to convert `integer1` and `string1` to type Decimal and assign them to `decimal1` and `decimal2`, respectively.

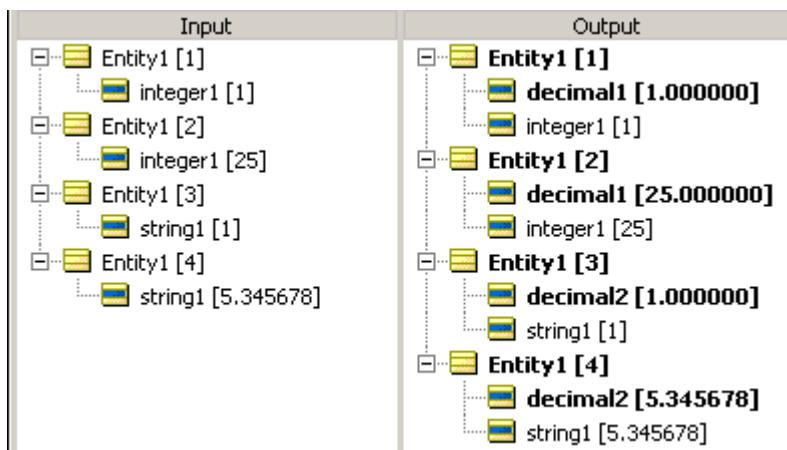
The screenshot shows the 'To Decimal.ers' rulesheet in the Rulesheet editor. The 'Actions' section contains two assignments:

- A: Entity1.decimal1 = Entity1.integer1.toDecimal
- B: Entity1.decimal2 = Entity1.string1.toDecimal

The 'Rule Statements' tab is selected, showing two entries:

- A0: decimal1 is equal to the value of integer1 converted into a decimal data type
- B0: decimal2 is equal to the value of integer1 converted into a decimal data type

SAMPLE TEST



To integer

SYNTAX

```
<Decimal>.toInteger  
<String>.toInteger
```

DESCRIPTION

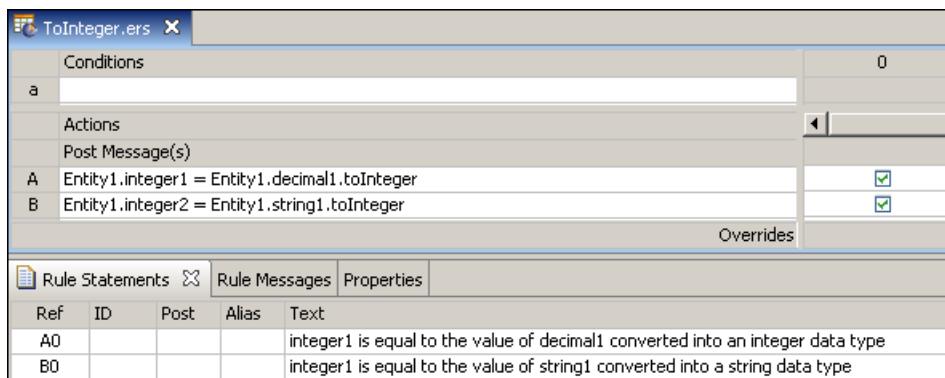
Converts the value in `<Decimal>` or all characters in `<String>` to data type Integer. All decimals have fractional portions truncated during the conversion. Strings are converted ONLY if all characters in `<String>` are numeric, without a decimal point. If any non-numeric characters (with the sole exception of a single leading minus sign for negative numbers) are present in `<String>`, no value is returned by the function. Do not use on String values of `null` or empty String (' ') -- a pair of single quote marks -- as this will generate an error message.

USAGE RESTRICTIONS

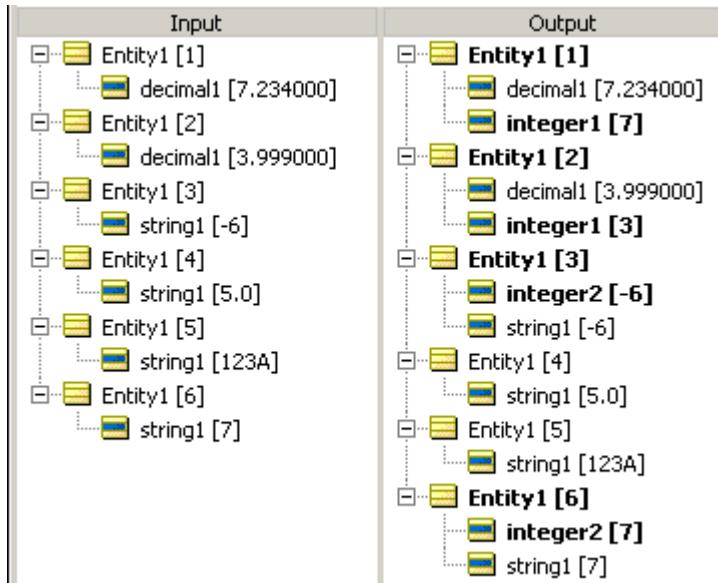
The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses `.toInteger` to convert `decimal1` and `string1` to type Integer and assign them to `integer1` and `integer2`, respectively.



SAMPLE TEST



To string

SYNTAX

`<Number>.toString`

`<DateTime*>.toString`

*includes DateTime, Date, and Time data types

DESCRIPTION

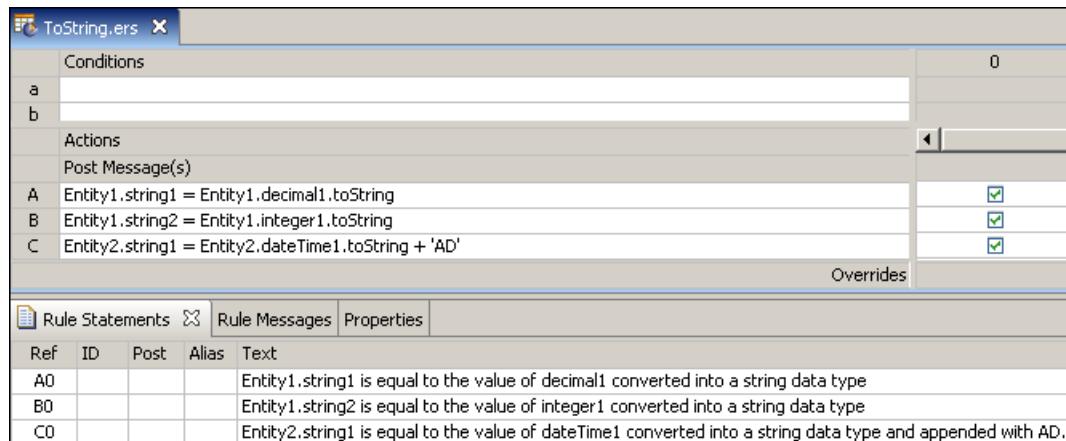
Converts a value to a data type of String.

USAGE RESTRICTIONS

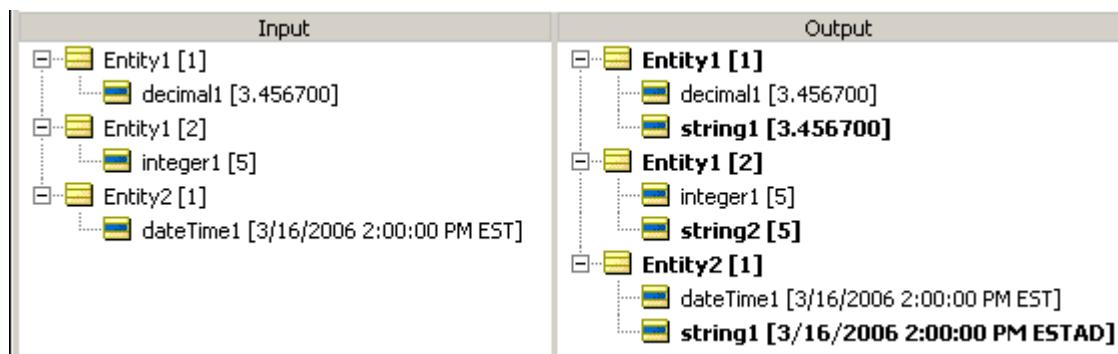
The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses `.toString` to convert 3 data types to strings. Rule N.3 also uses the alternative String concatenation syntax. See [Add Strings](#) for details.



SAMPLE TEST



To time: Casting a dateTIme to a time

SYNTAX

`<DateTime>.toTime`

DESCRIPTION

Converts the value in `<DateTime>` to a Time data type, containing only the time portion of the full DateTime. If `<DateTime>` contains no time information, then the time portion is set to `12:00:00 AM` in the current timezone.

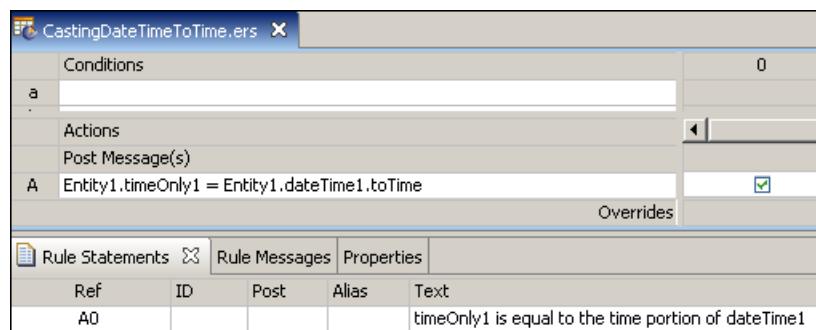
Note: Should this value be saved in a database field supporting a full DateTime, the date portion is saved as the epoch specified in `com.corticon.crml.OclDate.epochForTimeValues`. See *Server Integration & Deployment Guide* for a complete description of this and other Common properties.

USAGE RESTRICTIONS

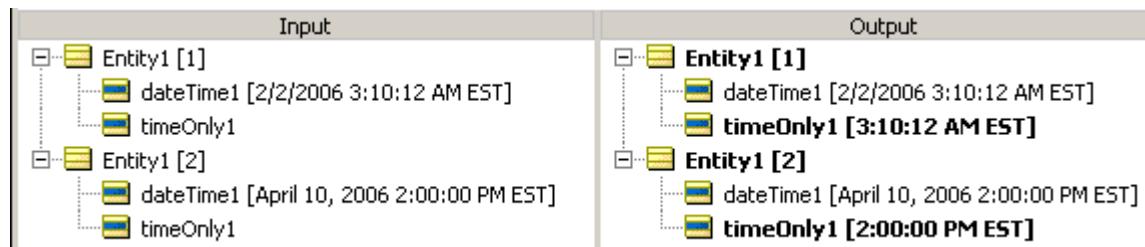
The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses `.toTime` to convert `dateTime1` to Time and assign the value to `TimeOnly1`.



SAMPLE TEST



Trend

SYNTAX

`<Collection.attribute> -> <Sequence>.trend`

DESCRIPTION

Returns one of the following 4-character strings depending on the trend of `<Collection.attribute>` once sequenced by the same or different attribute in `<Collection>`. `<Sequence>` is an ordered set of `<Collection>` in the form $\{x_1, x_2, x_3 \dots x_n\}$, where

INCR	the value of <attribute> of element x_{n+1} is greater than or equal to the value of <attribute> of element x_n for every element. At least one <attribute> value of element x must be greater than that of x_{n-1}
DECR	the value of <attribute> of element x_{n+1} is less than or equal to the value of <attribute> of element x_n for every element. At least one <attribute> value of element x must be less than that of x_{n-1}
CNST	the value of <attribute> of element x_{n+1} is equal to the value of <attribute> for element x_n for every element.
NONE	any <sequence> with elements not meeting the requirements for INCR, DECR, or CNST

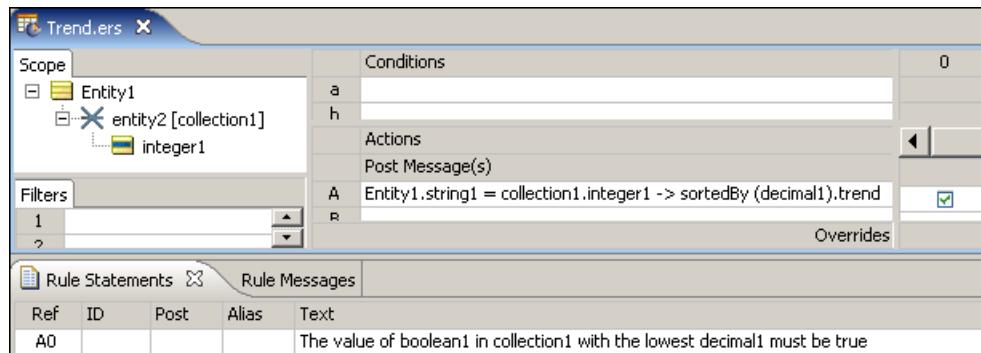
An alternative way to understand this operator is to view the index attribute used to sequence the collection as the *independent* variable (traditionally plotted along the "x" axis in a standard x-y graph) in a set of data pairs. The attribute evaluated by the **.trend** operator, <Collection.attribute>, is the *dependent* variable, plotted along the "y" axis. When so plotted, the 4-character words returned by **.trend** correspond to curves with positive, negative, zero (constant), or arbitrary slopes.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

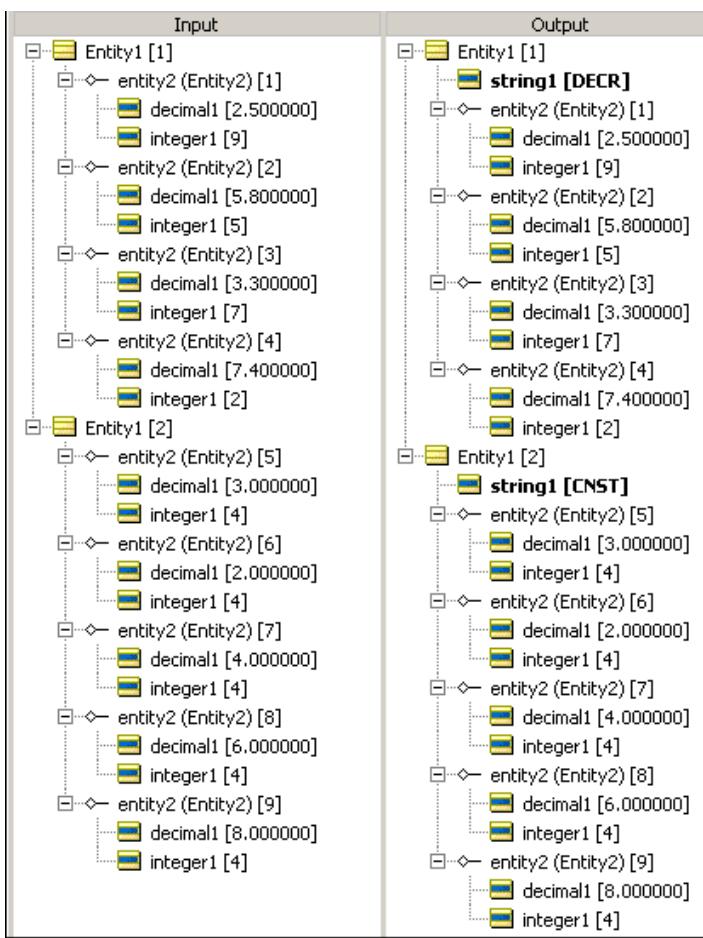
RULESHEET EXAMPLE

This Rulesheet uses the **.trend** function to analyze `integer1` attributes within `collection1` sorted by `decimal1`. The resulting trend value is assigned to `string1`.



SAMPLE TEST

Two sample tests provide two collections of elements, each with a `decimal1` and `integer1` values. Input and Output panels are shown below.



Note: Technically, the slope of an `INCR` curve need not be positive everywhere, but must have a first derivative (instantaneous slope) that is positive at some point along the curve and never be negative. The slope of a `CNST` curve must be zero everywhere.

True

SYNTAX

true or T

DESCRIPTION

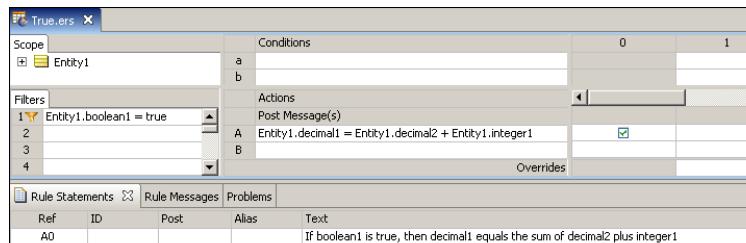
Represents Boolean value true. Recall from the discussion of [truth values](#) that an `<expression>` is evaluated for its truth value, so the expression `Entity1.boolean1=true` will evaluate to true only if `boolean1=true`. But since `boolean1` is Boolean and has a truth value all by itself without any additional syntax, we do not actually need the “`=true`” piece of the expression. Many examples in the documentation use explicit syntax like `boolean1=true` or `boolean2=false` for clarity and consistency, even though `boolean1` or not `boolean2` are equivalent logical expressions.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

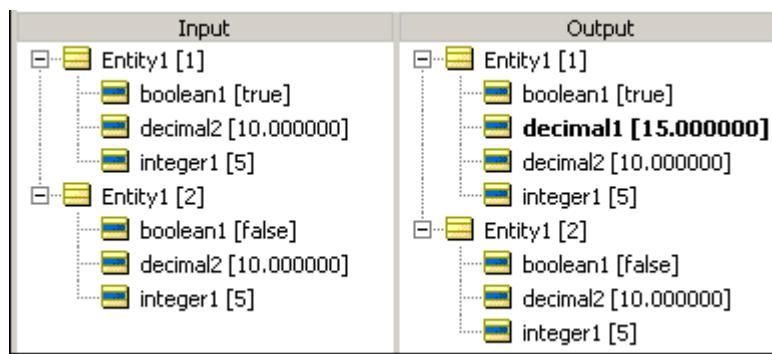
RULESHEET EXAMPLE

The following Rulesheet uses **true** in a Precondition to Ruletest whether `boolean1` is true, and perform the Nonconditional computation if it is. As discussed above, the alternative expression `Entity1.boolean1` is logically equivalent.



SAMPLE TEST

A sample Ruletest provides three examples. Assume `decimal2=10.0` and `integer1=5` for all examples. Input and Output panels are shown below:



Uppercase

SYNTAX

`<String>.toUpperCase`

DESCRIPTION

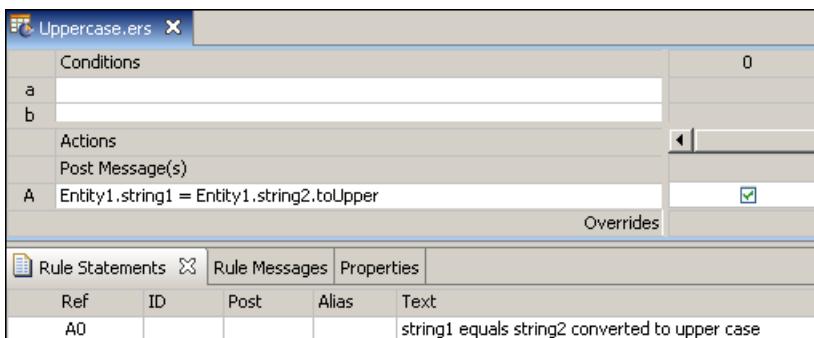
Converts all characters in `<String>` to uppercase.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

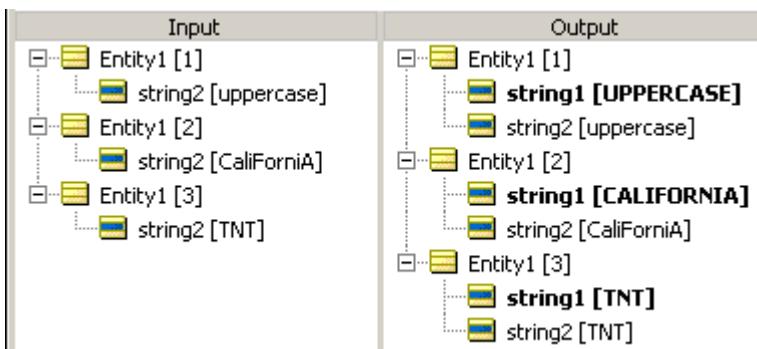
RULESHEET EXAMPLE

The following Rulesheet uses `.toUpperCase` to convert `string2` to uppercase and assign it to `string1`.



SAMPLE TEST

A sample Ruletest provides three examples. Input and Output panels are shown below:



Week of month

SYNTAX

<DateTime>.weekOfMonth

<Date>.weekOfMonth

DESCRIPTION

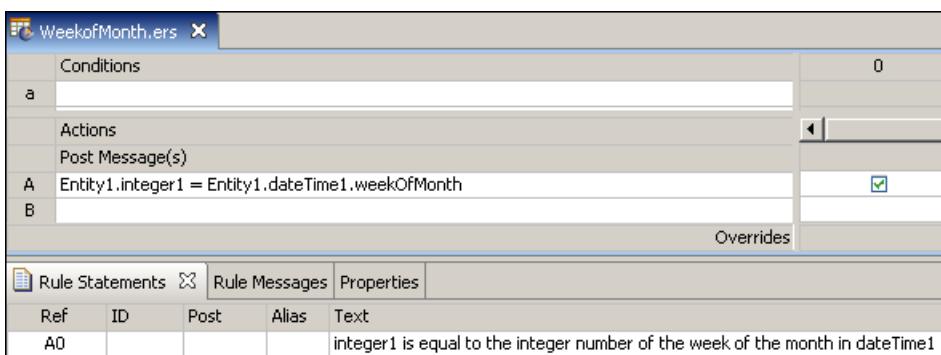
Returns an Integer from 1 to 6, equal to the week number within the month in <DateTime> or <Date>. A week begins on Sunday and ends on Saturday.

USAGE RESTRICTIONS

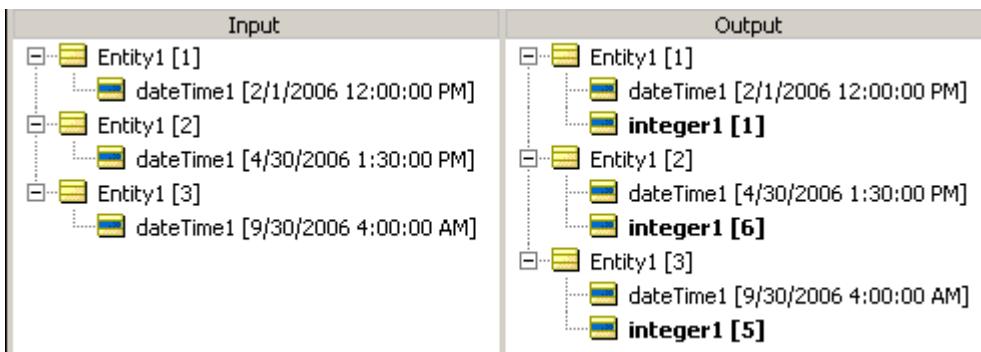
The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses **.weekOfMonth** to assign a value to `integer1`.



SAMPLE TEST



Week of year

SYNTAX

<DateTime>.weekOfYear

<Date>.weekOfYear

DESCRIPTION

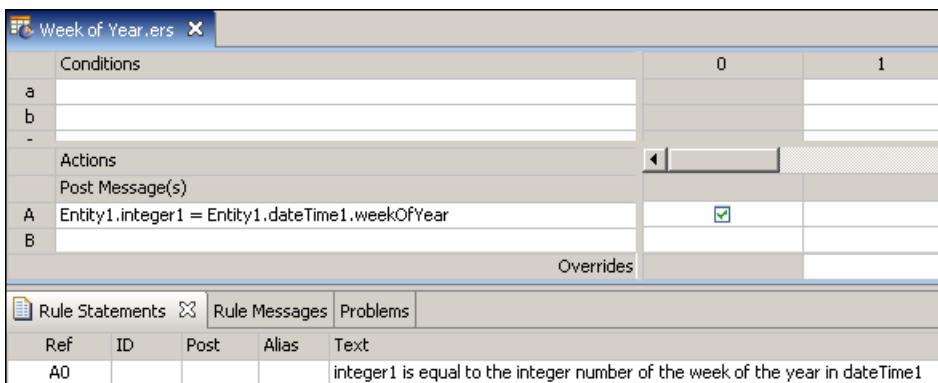
Returns an Integer from 1 to 52, equal to the week number within the year in <DateTime> or <Date>. A week begins on Sunday and ends on Saturday. When a year ends between Sunday and the next Friday, or in other words when a new year begins between Monday and the next Saturday, the final day(s) of December will be included in week 1 of the new year. For example, 12/29/2013 fell on a Sunday, so 12/29-31 are included in week 1 of 2014.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses .weekOfYear to assign a value to integer1.



SAMPLE TEST

Input		Output	
Entity1 [1]	dateTime1 [12/30/2013 2:00:00 PM] integer1	Entity1 [1]	dateTime1 [12/30/2013 2:00:00 PM] integer1 [1]
Entity1 [2]	dateTime1 [8/25/2014 11:45:00 AM] integer1	Entity1 [2]	dateTime1 [8/25/2014 11:45:00 AM] integer1 [35]
Entity1 [3]	dateTime1 [3/16/2016 10:30:00 PM] integer1	Entity1 [3]	dateTime1 [3/16/2016 10:30:00 PM] integer1 [12]

Year

SYNTAX

<DateTime>.year

<Date>.year

DESCRIPTION

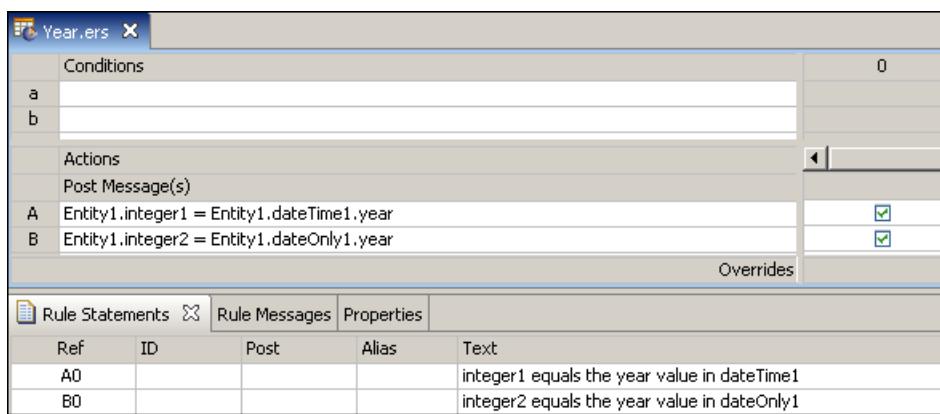
Returns the century/year portion of <DateTime> or <Date>. The returned value is a four digit Integer.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

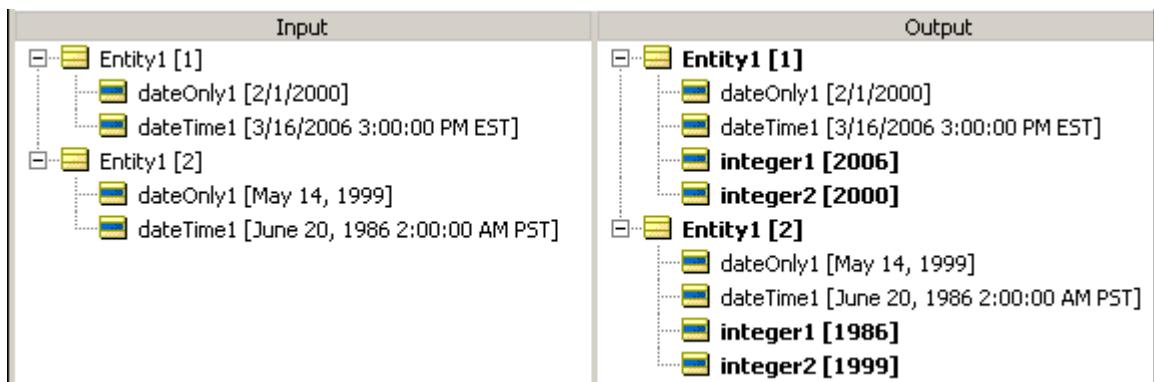
RULESHEET EXAMPLE

The following Rulesheet uses **.year** to evaluate `dateTime1` and `dateOnly1` and assign the year values to `integer1` and `integer2`, respectively.



SAMPLE TEST

A sample Ruletest provides three examples of `dateTime1` and `dateOnly1`. Input and Output panels are shown below:



Years between

SYNTAX

```
<DateTime1>.yearsBetween(<DateTime2>)
<Date1>.yearsBetween(<Date2>)
```

DESCRIPTION

Returns the Integer number of years between DateTimes or between Dates. The number of months in `<DateTime2>` is subtracted from the number of months in `<DateTime1>`, and the result is divided by 12 and truncated. This function returns a positive number if `<DateTime2>` is later than `<DateTime1>`.

USAGE RESTRICTIONS

The Operators row of the table in [Summary Table of Vocabulary Usage Restriction](#) applies. No special exceptions.

RULESHEET EXAMPLE

The following Rulesheet uses **.yearsBetween** to determine the number of months that have elapsed between `dateTime1` and `dateTime2`, compare it to the `Values` set, and assign a value to `string1`.

The screenshot shows the 'YearsBetween.ers' Rulesheet editor. The Conditions section contains two rows: 'a' (Entity1.dateTime1.yearsBetween(Entity1.dateTime2)) and 'b'. The Actions section contains 'Post Message(s)' and 'A Entity1.string1'. The Overrides section shows two rows: '0' (<=3) and '2' (>3). Below the editor are tabs for 'Rule Statements', 'Rule Messages', and 'Properties'. A table under 'Rule Statements' lists two entries:

Ref	ID	Post	Alias	Text
1				If 3 or fewer years have elapsed between dateTime1 and dateTime2, then Entity1 is not overdue
2				If more than 3 years have elapsed between dateTime1 and dateTime2, then Entity1 is overdue

SAMPLE TEST

A sample Ruletest provides `dateTime1` and `dateTime2` for two examples. Input and Output panels are shown below.

The screenshot shows the Ruletest interface with two panels: 'Input' and 'Output'. The 'Input' panel shows two entities: Entity1 [1] and Entity1 [2]. Entity1 [1] has two date-time values: dateTime1 [Tuesday, May 9, 2006 2:30:00 PM EST] and dateTime2 [Thursday, February 5, 2004 5:30:00 PM EST]. Entity1 [2] has two date-time values: dateTime1 [3/10/1992 2:00:00 PM PST] and dateTime2 [7/1/2006 11:30:00 AM PST]. The 'Output' panel shows the results for each entity. For Entity1 [1], it shows dateTime1, dateTime2, and string1 [Not Overdue]. For Entity1 [2], it shows dateTime1, dateTime2, and string1 [Overdue].

A

Character precedence: Unicode and Java Collator

The Unicode standard assigns a 4 digit (hexadecimal) code to every character, including many that can't be typed on standard keyboards. Java (and hence Progress Corticon software) uses a special method named `Collator` to sort these characters in specific sequences based on the I18n locale of the user.

While sorting by locale allows for regional variations of language-specific characters like accents, the combination of these two systems can also make determining character precedence very complicated. The Unicode code and Java Collator sequence for standard keyboards in US-English locale is shown in the table below.

Sequences for other languages and/or locales may differ, and many other Unicode characters are available but are not shown in the table. We recommend <http://www.unicode.org/charts> for more information on the Unicode system and <http://java.sun.com/docs/books/tutorial/i18n/text/locale.html> for more information on the Java Collator method.

- `'Z'='z'` evaluates to `true` because character `Z` has the same precedence as `z` ($69=69$). A given letter has the same precedence regardless of its case. This is an important difference between character precedence determined by ISO or ASCII systems, and the Java Collator system used by Corticon.
- `'C & S' < 'C and S'` evaluates to `true` because character `a` has a higher precedence than `&` ($26 < 44$). These characters are decisive because they are the first different characters encountered as the two strings are compared beginning with characters in position 1.
- `'B' > 'aardvark'` evaluates to `true` because character `B` has a higher precedence than `a` ($45 > 44$).
- `'Marilynn' < 'Marilyn'` evaluates to `false` because character `n` has a higher precedence than `<space>` ($57 > 1$). The first seven characters of each String are identical, so the final character comparison is decisive.

character	name	precedence	Unicode 5.0 code
	typed space	1	0020
-	dash or minus sign	2	002D
_	underline or underscore	3	005F
,	comma	4	002C
;	semicolon	5	003B
:	colon	6	003A
!	exclamation point	7	0021
?	question mark	8	003F
/	slash	9	002F
.	period	10	002E
`	grave accent	11	0060
^	circumflex	12	005E
~	tilde	13	007E
'	apostrophe	14	0027
"	quotation marks	15	0022
(left parenthesis	16	0028
)	right parenthesis	17	0029
[left bracket	18	005B
]	right bracket	19	005D
{	left brace	20	007B
}	right brace	21	007D
@	at symbol	22	0040
\$	dollar sign	23	0024
*	asterisk	24	002A
\	backslash	25	005C
&	ampersand	26	0026

character	name	precedence	Unicode 5.0 code
#	number sign or hash sign	27	0023
%	percent sign	28	0025
+	plus sign	29	002B
<	less than sign	30	003C
=	equals sign	31	003D
>	greater than sign	32	003E
	vertical line	33	007C
0..9	numbers 1 through 9	34-43	0031-0039
a, A	letter a, small and capital	44	0061, 0041
b, B	letter b, small and capital	45	0062, 0042
c, C	letter c, small and capital	46	0063, 0043
d, D	letter d, small and capital	47	0064, 0044
e, E	letter e, small and capital	48	0065, 0045
f, F	letter f, small and capital	49	0066, 0046
g, G	letter g, small and capital	50	0067, 0047
h, H	letter h, small and capital	51	0068, 0048
i, I	letter i, small and capital	52	0069, 0049
j, J	letter j, small and capital	53	006A, 004A
k, K	letter k, small and capital	54	006B, 004B
l, L	letter l, small and capital	55	006C, 004C
m, M	letter m, small and capital	56	006D, 004D
n, N	letter n, small and capital	57	006E, 004E
o, O	letter o, small and capital	58	006F, 004F
p, P	letter p, small and capital	59	0070, 0050
q, Q	letter q, small and capital	60	0071, 0051
r, R	letter r, small and capital	61	0072, 0052

character	name	precedence	Unicode 5.0 code
s, S	letter s, small and capital	62	0073, 0053
t, T	letter t, small and capital	63	0074, 0054
u, U	letter u, small and capital	64	0075, 0055
v, V	letter v, small and capital	65	0076, 0056
w, W	letter w, small and capital	66	0077, 0057
x, X	letter x, small and capital	67	0078, 0058
y, Y	letter y, small and capital	68	0079, 0059
z, Z	letter z, small and capital	69	007A, 005A

B

Precedence of rule operators

The precedence of operators affects the grouping and evaluation of expressions. Expressions with higher-precedence operators are evaluated first. Where several operators have equal precedence, they are evaluated from left to right. The following table summarizes Corticon's operator precedence.

Operator precedence	Operator	Operator Name	Example
1	()	Parenthetic expression	(5.5 / 10)
2	-	Unary negative	-10
	not	Boolean test	not 10
3	*	Arithmetic: Multiplication	5.5 * 10
	/	Arithmetic: Division	5.5 / 10
	**	Arithmetic: Exponentiation (Powers and Roots)	5 ** 2 25 ** 0.5 125 ** (1.0/3.0)
4	+	Arithmetic: Addition	5.5 + 10
	-	Arithmetic: Subtraction	10.0 – 5.5
5	<	Relational: Less Than	5.5 < 10
	<=	Relational: Less Than Or Equal To	5.5 <= 5.5
	>	Relational: Greater Than	10 > 5.5
	>=	Relational: Greater Than Or Equal To	10 >= 10
	=	Relational: Equal	5.5 = 5.5
	<>	Relational: Not Equal	5.5 <> 10
6	(expression, expression)	Logical: AND	(>5.5,<10)
	(expression or expression)	Logical: OR	(<5.5 or >10)

Note: While expressions within parentheses that are separated by logical AND / OR operators are valid, the component expressions are not evaluated individually when testing for completeness, and might cause unintended side effects during rule execution. Best practice within a Corticon Rulesheet is to represent AND conditions as separate condition rows and OR conditions as separate rules -- doing so allows you to get the full benefit of Corticon's logical analysis.

Note: It is recommended that you place arithmetic exponentiation expressions in parentheses.

C

DateTime data type

FORMATS OR MASKS

DateTime information may take many different formats. Corticon Studios use a common source of acceptable DateTime, Date Only, and Time Only formats, also known as “masks”.

For example, a date mask may specify `yyyy-MM-dd` as an acceptable date format, which means that an attribute of type DateTime (or Date) may “hold” or “contain” data that conforms to this format. ‘`2013-04-12`’ conforms to this mask; ‘`April12th, 2013`’ does not.

For proper execution, it is important to ensure that date formats used during rule development and testing (and are included in the rule builders' Corticon Studio installations) are also present in the Corticon Server's installation. See *Server Integration & Deployment Guide* for more details about setting and modifying masks in both Corticon Studio and Corticon Server.

PRESENTATION AND PERSISTENCE

Most commercial databases represent dates as DateTimes. Such DateTimes are frequently stored as UTC, namely the number of milliseconds that have transpired from an arbitrary epoch (for example, `1/1/1970 00:00:00` GMT); this is not a universal standard but is a very popular convention. UTC dates can be *rendered* in the user's local time zone, *but this is merely a matter of presentation*. A UTC represents a simultaneous point in time for two observers regardless of where on earth they reside.

However, some date or time concepts, such as *holiday*, cannot be expressed conveniently as a discrete time point. *Christmas* (`12/25/XX`) actually denotes different time frames depending on the observers' time zones; thus, Corticon *carries* (that is, holds in memory) all dates in GMT with the time portion zeroed (that is, midnight). This approach addresses the holiday problem because a user can enter holiday dates into the database and not have them shift when they are rendered in the user's local time zone.

Carrying GMT dates should be transparent to the user. Dates expressed as strings in incoming XML are parsed and the proper data type is inferred; for dates, they are immediately instantiated as GMT and rendered back in GMT with no conversion.

D

Access to Corticon knowledge resources

Complete online documentation for the current release

Corticon online tutorials available in the [Corticon Learning Center](#):

- Tutorial: Basic Rule Modeling in Corticon Studio
- Tutorial: Advanced Rule Modeling in Corticon Studio
- Modeling Progress Corticon Rules to Access a Database using EDC
- Connecting a Progress Corticon Decision Service to a Database using EDC
- Deploying a Progress Corticon Decision Service as a Web Service for Java
- Deploying a Progress Corticon Decision Service in process for Java
- Using Corticon Business Rules in a Progress OpenEdge Application

Corticon guides (PDF):

- [What's New in Corticon](#)
- [Corticon Installation Guide](#)
- [Corticon Studio: Rule Modeling Guide](#)
- [Corticon Studio: Quick Reference Guide](#)
- [Corticon Studio: Rule Language Guide](#)
- [Corticon Studio: Extensions Guide](#)
- [Corticon Server: Integration and Deployment Guide](#)

- [Corticon Server: Web Console Guide](#)
- [Corticon Server: Deploying Web Services with Java](#)
- [Corticon Server: Deploying Web Services with .NET](#)

Corticon JavaDoc API reference (HTML):

- [Corticon Server API](#)
- [CorticonExtensions API](#)

See also:

- [Introducing the Progress® Application Server](#)
- Corticon documentation for this release on the [Progress download site](#): What's New Guide (PDF), Installation Guide (PDF), PDF download package, and the online Eclipse help installed with Corticon Studio.