



# Corticon

## Rule Language



# Copyright

---

© 2020 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

These materials and all Progress<sup>®</sup> 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.

Corticon, DataDirect (and design), DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, DataRPM, Defrag This, Deliver More Than Expected, Icenium, Ipswitch, iMacros, Kendo UI, Kinvey, MessageWay, MOVEit, NativeChat, NativeScript, OpenEdge, Powered by Progress, Progress, Progress Software Developers Network, SequeLink, Sitefinity (and Design), Sitefinity, SpeedScript, Stylus Studio, TeamPulse, Telerik, Telerik (and Design), Test Studio, WebSpeed, WhatsConfigured, WhatsConnected, WhatsUp, and WS\_FTP are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries. Analytics360, AppServer, BusinessEdge, DataDirect Autonomous REST Connector, DataDirect Spy, SupportLink, DevCraft, Fiddler, iMail, JustAssembly, JustDecompile, JustMock, NativeScript Sidekick, OpenAccess, ProDataSet, Progress Results, Progress Software, ProVision, PSE Pro, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, and WebClient 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.

**Last updated with new content:** Corticon 6.1.1

**Updated:** 2020/09/24



# Table of Contents

<b>Introduction to Corticon Rule Language.....</b>	<b>9</b>
Rule structure.....	10
Basic data types.....	10
Truth values.....	11
Collection operators.....	11
Language operators.....	11
Vocabulary used in this Language Guide.....	12
<b>How to access rule operators.....</b>	<b>13</b>
<b>Usage restrictions.....</b>	<b>15</b>
<b>Rule operators.....</b>	<b>17</b>
Attribute operators.....	18
Boolean.....	18
Date.....	19
DateTime.....	22
Decimal.....	27
Integer.....	30
String.....	34
Time.....	38
Entity and Association operators.....	40
Entity.....	41
Collection.....	41
Sequence.....	44
General terms.....	45
<b>Rule operator details and examples.....</b>	<b>47</b>
Absolute value.....	51
Add numbers.....	52
Add strings.....	53
Add days.....	54
Add hours.....	55
Add minutes.....	56
Add months.....	57
Add seconds.....	58

Add years.....	60
Associate elements.....	61
At.....	63
Average.....	64
CellValue.....	65
Clone.....	69
Concatenate.....	72
Character at.....	74
Contains.....	75
Day.....	76
Days between.....	77
Day of week.....	79
Day of year.....	80
Decrement.....	81
Disassociate elements.....	82
Divide.....	83
Div.....	84
Ends with.....	85
Equals when used as an assignment.....	86
Equals when used as a comparison.....	88
Equals ignoring case.....	89
Equals when using Strings.....	90
Exists.....	91
Exponent.....	93
False.....	94
Floor.....	95
Get Milliseconds .....	97
First.....	98
First NUMBER.....	99
For all.....	102
Greater than.....	104
Greater than or equal to.....	105
Hour.....	107
Hour between.....	108
In LIST.....	109
In RANGE.....	111
Increment.....	113
Index of.....	114
Is integer.....	116
Is empty.....	117
Iterate.....	118
Last.....	119
Last NUMBER.....	120
Less than.....	123
Less than or equal to.....	125

---

Logarithm BASE 10.....	126
Logarithm BASE X.....	127
Lowercase.....	128
Matches.....	130
Maximum value.....	131
Maximum value COLLECTION.....	132
Minimum value.....	134
Minimum value COLLECTION.....	135
Minute.....	136
Minutes between.....	137
Mod.....	138
Month.....	139
Months between.....	141
Multiply.....	142
Natural logarithm.....	143
New.....	145
New unique.....	146
Not.....	149
Not empty.....	150
Not equal to.....	151
Now.....	153
Null.....	154
Other.....	156
Or.....	157
Random.....	159
Remove element.....	160
Replace elements.....	163
Replace String.....	165
Regular expression to replace String.....	166
Round.....	167
Second.....	169
Seconds between.....	170
Size of string.....	171
Size of collection.....	172
Sorted by.....	173
Sorted by descending.....	176
Starts with.....	178
SubSequence.....	179
Substring.....	181
Subtract.....	182
Sum.....	183
Today.....	185
To date Casting a dateTime to a date.....	186
To dateTime Casting a string to a dateTime.....	187
To dateTime Casting a date to a dateTime.....	188

To dateTime Casting a time to a dateTime.....	189
To dateTime Timezone offset.....	190
To decimal.....	191
To integer.....	193
To string.....	196
To time Casting a dateTime to a time.....	197
Trend.....	198
Trim spaces.....	200
True.....	201
Uppercase.....	202
Week of month.....	203
Week of year.....	204
Year.....	205
Years between.....	207
<b>Appendix A: Standard Boolean constructions.....</b>	<b>209</b>
Boolean AND.....	209
Boolean NAND.....	212
Boolean OR.....	212
Boolean XOR.....	213
Boolean NOR.....	214
Boolean XNOR.....	215
<b>Appendix B: Character precedence in Unicode and Java Collator.....</b>	<b>217</b>
<b>Appendix C: Precedence of rule operators.....</b>	<b>221</b>
<b>Appendix D: Formats for Date Time and DateTime properties.....</b>	<b>225</b>

## 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](http://www.uml.org).

---

**Note:** You will see many references to rule statements and rule messages in the examples in this guide. As the product is closely related to the Corticon Classic product, many of its examples are used in this guide. The rule message feature is currently not available in the Corticon.js product.

---

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 rules is dependent on the type of data involved. Each attribute in the Corticon Vocabulary has a data type, meaning that it has restrictions on the type of data it can contain. Corticon standard data types are as follows:

Data Type	Description
String	Any combination of alphanumeric characters, of any length,
Integer	A whole number, including zero and negative numbers, to the maximum values for a 64-bit long signed integer (-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)
Decimal	A number containing a decimal point, including zero and negative numbers to the limits of double precision (see <a href="#">IEEE_754</a> for details.)
Boolean	Values are <code>true</code> and <code>false</code> . <code>T</code> and <code>F</code> can 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 [Formats for Date Time and DateTime properties](#) on page 225 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.

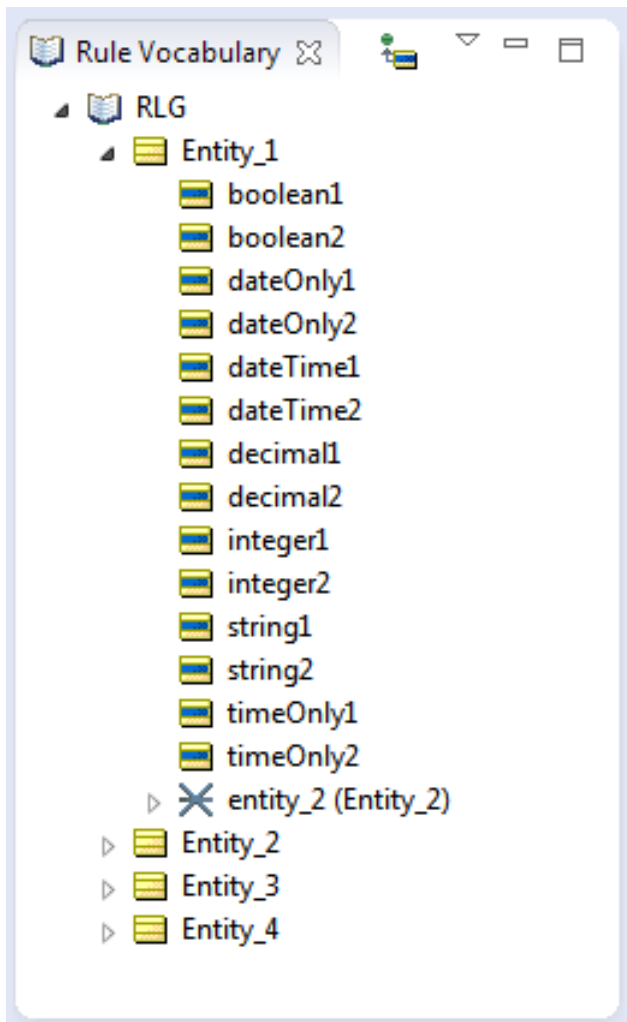
## Language operators

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

## 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**










## How to access rule operators

The Studio tools for accessing operators provide icons with decorations, and tooltips.

### Icons

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

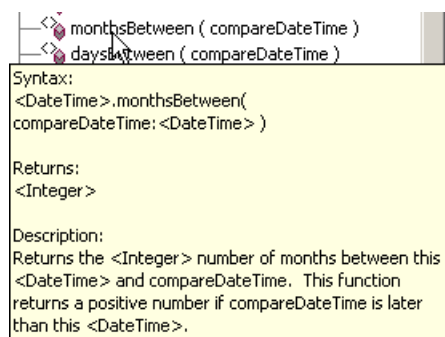
Icon	Where Found	Purpose	Examples
	General, Literals category	indicates special values or constants	<a href="#">null</a> , <a href="#">true</a> , <a href="#">other</a>
	General, Functions category	indicates system values that are automatically retrieved upon rule execution.	<a href="#">now</a> , <a href="#">today</a>
	Operators, Boolean category	this special “unary” operator icon is used only with not	<a href="#">not</a>
	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.	<a href="#">day</a> , <a href="#">round</a> , <a href="#">contains</a>

Icon	Where Found	Purpose	Examples
	Operators, all categories	indicates the operator is used between two operands. Most operators with this icon typically fell into the previous “comparison” category.	<a href="#">equals</a> , <a href="#">multiply</a>
	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.	<a href="#">sum</a> , <a href="#">size</a>
	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, causes 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 set of topics. For questions not answered by the tool tip, refer to the detailed operator descriptions in this publication. The following figure shows a typical tool tip for the date operator `.monthsBetween`:

**Figure 2: Typical Rule Operator Tool Tip**



## **Usage restrictions**

---

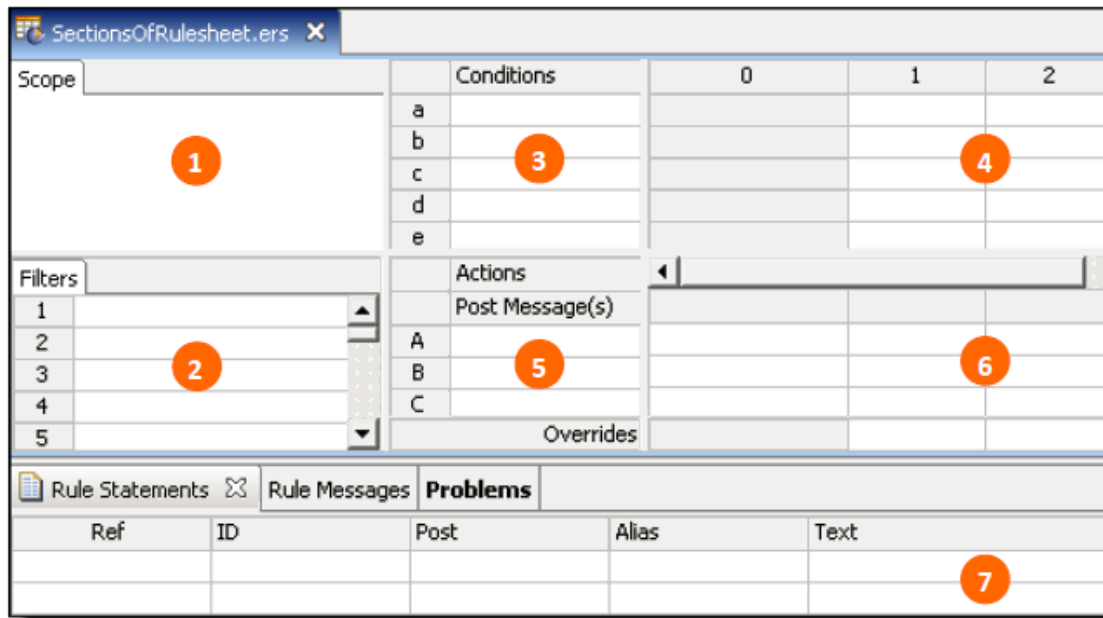
The following illustrations show the general usage restrictions for the various types of Vocabulary terms depending on where they are used in a Rulesheet. This table indicates, for example, that entities (terms from the Vocabulary) may be used in any section of the Rulesheet. Rule Operators, however, 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"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Functions			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Operators			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>		
Data	Values		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	Terms	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

**Figure 4: Sections of Rulesheet that correlate with usage restrictions**



## **Rule operators**

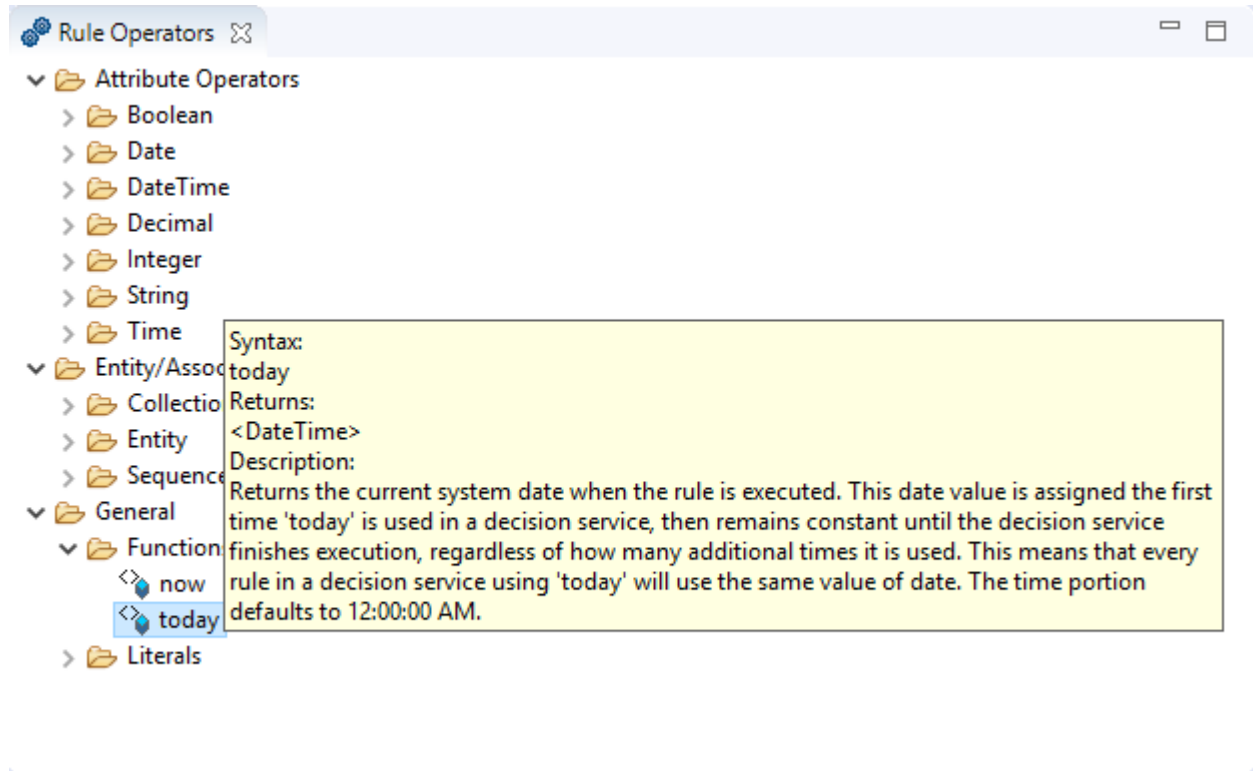
---

Rule operators are a structured listing of the *verbs* that you can apply to the *nouns* in the Vocabulary. Corticon Studio presents its rule operators in logical groups.

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

### Figure 5: Rule Operator categories

When you open an operator group and hover over an operator, a help window shows its syntax, and details about that operator:



For details, see the following topics:

- [Attribute operators](#)
- [Entity and Association operators](#)
- [General terms](#)

## 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
<a href="#">Equals</a> (used as a comparison)		

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

**Note:** See also related information in the topics [Precedence of rule operators](#) on page 221 and [Standard Boolean constructions](#) on page 209..

## Date

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

Name and Syntax	Returns	Description
<a href="#">Equals</a> (used as a comparison)		
<Date1> = <Date2>	Boolean	Returns a value of true if <Date1> is the same as <Date2>.
<a href="#">Equals</a> (used as an assignment)		
<Date1> = <Date2>	DateTime	Assigns the value of <Date2> to <Date1>

Name and Syntax	Returns	Description
<b>Not Equal To</b>		
<Date1> <> <Date2>	Boolean	Returns a value of true if <Date1> does not equal <Date2>
<b>Less than</b>		
<Date1> < <Date2>	Boolean	Returns a value of true if <Date1> is less than <Date2>
<b>Greater than</b>		
<Date1> > <Date2>	Boolean	Returns a value of true if <Date1> is greater than or equal to <Date2>
<b>Less than or Equal to</b>		
<Date1> <= <Date2>	Boolean	Returns a value of true if <Date1> is less than or equal to <Date2>
<b>Greater than or Equal to</b>		
<Date1> >= <Date2>	Boolean	Returns a value of true if <Date1> is greater than or equal to <Date2>
<b>In (Range)</b>		
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.
<b>In (List)</b>		
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.
<b>Year</b>		
<Date>.year	Integer	Returns the century/year portion of <Date> as a four digit Integer
<b>Month</b>		
<Date>.month	Integer	Returns the month in <Date> as an Integer between 1 and 12
<b>Day</b>		

Name and Syntax	Returns	Description
<Date>.day	Integer	Returns the day portion of <Date> as an Integer between 1 and 31
<a href="#">Add years</a>		
<Date>.addYears(<Integer>)	Date	Adds the number of years in <Integer> to the number of years in <Date>
<a href="#">Add months</a>		
<Date>.addMonth(<Integer>)	Date	Adds the number of months in <Integer> to the number of months in <DateTime>
<a href="#">Add days</a>		
<Date>.addDays(<Integer>)	Date	Adds the number of days in <Integer> to the number of days in <Date>
<a href="#">Years between</a>		
<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>.
<a href="#">Months between</a>		
<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>.
<a href="#">Days between</a>		
<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>.
<a href="#">Day of Week</a>		
<Date>.dayOfWeek	Integer	Returns an Integer corresponding to day of the week, with Sunday equal to 1, in <Date>.
<a href="#">Week of Year</a>		
<Date>.weekOfYear	Integer	Returns an Integer from 1 to 52, equal to the week number within the year in <Date>
<a href="#">Day of Year</a>		

Name and Syntax	Returns	Description
<code>&lt;Date&gt;.dayOfYear</code>	Integer	Returns an Integer from 1 to 366, equal to the day number within the year in <code>&lt;Date&gt;</code>
<a href="#">Week of Month</a>		
<code>&lt;Date&gt;.weekOfMonth</code>	Integer	Returns an Integer from 1 to 6, equal to the week number within the month in <code>&lt;DateTime&gt;</code> or <code>&lt;Date&gt;</code> . A week begins on Sunday and ends on Saturday.
<a href="#">To String</a>		
<code>&lt;Date&gt;.toString</code>	String	Converts <code>DateTime</code> to a String with date and time information
<a href="#">To DateTime</a>		
<code>&lt;Date&gt;.toDateTime</code>	<code>DateTime</code>	Returns a <code>DateTime</code> where the date portion is equal to the value of <code>&lt;Date&gt;</code> and the time portion is equal to 00:00:00 in the system's local timezone
<a href="#">To DateTime with Timezone Offset</a>		
<code>&lt;Date&gt;.toDateTime (&lt;string&gt;)</code>	<code>DateTime</code>	Returns a <code>DateTime</code> where the date portion is equal to the value of <code>&lt;Date&gt;</code> and the time portion is equal to 00:00:00 in the timezone specified by the value of <code>&lt;string&gt;</code>
<a href="#">getMilliseconds</a>		
<code>&lt;Date&gt;.getMilliseconds</code>	Integer	Returns the internal date/time, namely the number of milliseconds that have transpired since the epoch 1/1/1970 00:00:00 GMT.
<a href="#">nextDay</a>		
<code>&lt;Date&gt;.nextDay</code>	Date	Returns the Date that represents the date that follows this Date instance.

## DateTime

---

**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
<b>Equals</b> (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
<b>Equals</b> (used as an assignment)		
<DateTime1> = <DateTime2>	DateTime	Assigns the value of <DateTime2> to <DateTime1>
<b>Not Equal To</b>		
<DateTime1> <> <DateTime2>	Boolean	Returns a value of true if <DateTime1> does not equal <DateTime2>
<b>Less than</b>		
<DateTime1> < <DateTime2>	Boolean	Returns a value of true if <Date1> is less than <Date2>
<b>Greater than</b>		
<DateTime1> > <DateTime2>	Boolean	Returns a value of true if <DateTime1> is greater than or equal to <DateTime2>
<b>Less than or Equal to</b>		
<DateTime1> <= <DateTime2>	Boolean	Returns a value of true if <DateTime1> is less than or equal to <DateTime2>
<b>Greater than or Equal to</b>		
<DateTime1> >= <DateTime2>	Boolean	Returns a value of true if <DateTime1> is greater than or equal to <DateTime2>
<b>In (Range)</b>		
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.
<b>In (List)</b>		
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.

Name and Syntax	Returns	Description
<b>Year</b>		
<DateTime>.year	Integer	Returns the century/year portion of <DateTime> as a four digit Integer
<b>Month</b>		
<DateTime>.month	Integer	Returns the month in <DateTime> as an Integer between 1 and 12
<b>Day</b>		
<DateTime>.day	Integer	Returns the day portion of <DateTime> as an Integer between 1 and 31
<b>Hour</b>		
<DateTime>.hour	Integer	Returns the hour portion of <DateTime>. The returned value is based on a 24-hour clock.
<b>Minute</b>		
<DateTime>.min	Integer	Returns the minute portion of <DateTime> as an Integer between 0 and 59
<b>Second</b>		
<DateTime>.sec	Integer	Returns the seconds portion of <DateTime> as an Integer between 0 and 59
<b>Add years</b>		
<DateTime>.addYears (<Integer>)	Date	Adds the number of years in <Integer> to the number of years in <DateTime>
<b>Add months</b>		
<DateTime>.addMonths (<Integer>)	Date	Adds the number of months in <Integer> to the number of months in <DateTime>
<b>Add days</b>		
<DateTime>.addDays (<Integer>)	Date	Adds the number of days in <Integer> to the number of days in <DateTime>
<b>Add hours</b>		
<DateTime>.addHours (<Integer>)	Date	Adds the number of hours in <Integer> to the number of hours in the Time portion of <DateTime>

Name and Syntax	Returns	Description
<a href="#">Add minutes</a>		
<DateTime>.addMinutes (<Integer>)	Date	Adds the number of minutes in <Integer> to the number of minutes in the Time portion of <DateTime>
<a href="#">Add seconds</a>		
<DateTime>.addSeconds (<Integer>)	Date	Adds the number of seconds in <Integer> to the number of seconds in the Time portion of <DateTime>
<a href="#">Years between</a>		
<DateTime1>.yearsBetween (<DateTime2>)	Integer	Returns the Integer number of years between <DateTime1> and <Date2>. This function returns a positive number if <DateTime2> is later than <DateTime1>.
<a href="#">Months between</a>		
<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>.
<a href="#">Days between</a>		
<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>.
<a href="#">Hours between</a>		
<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>.
<a href="#">Minutes between</a>		
<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>.
<a href="#">Seconds between</a>		

Name and Syntax	Returns	Description
<code>&lt;DateTime1&gt;.secsBetween (&lt;DateTime2&gt;)</code>	Integer	Returns the Integer number of seconds between <code>&lt;DateTime1&gt;</code> and <code>&lt;DateTime2&gt;</code> . This function returns a positive number if <code>&lt;DateTime2&gt;</code> is later than <code>&lt;DateTime1&gt;</code> .
<a href="#">Day of Week</a>		
<code>&lt;DateTime&gt;.dayOfWeek</code>	Integer	Returns an Integer corresponding to day of the week, with Sunday equal to 1, in <code>&lt;DateTime&gt;</code> .
<a href="#">Week of Year</a>		
<code>&lt;DateTime&gt;.weekOfYear</code>	Integer	Returns an Integer from 1 to 52, equal to the week number within the year in <code>&lt;DateTime&gt;</code>
<a href="#">Day of Year</a>		
<code>&lt;DateTime&gt;.dayOfYear</code>	Integer	Returns an Integer from 1 to 366, equal to the day number within the year in <code>&lt;DateTime&gt;</code>
<a href="#">Week of Month</a>		
<code>&lt;DateTime&gt;.weekOfMonth</code>	Integer	Returns an Integer from 1 to 6, equal to the week number within the month in <code>&lt;DateTime&gt;</code> or <code>&lt;Date&gt;</code> . A week begins on Sunday and ends on Saturday.
<a href="#">To Date</a>		
<code>&lt;DateTime&gt;.toDate</code>	Date	Returns the date portion only of <code>DateTime</code>
<a href="#">To Time</a>		
<code>&lt;DateTime&gt;.toTime</code>	Time	Returns the time portion only of <code>DateTime</code>
<a href="#">To String</a>		
<code>&lt;DateTime&gt;.toString</code>	String	Converts <code>DateTime</code> to a String with date and time information
<a href="#">getMilliseconds</a>		
<code>&lt;DateTime&gt;.getMilliseconds</code>	Integer	Returns the internal date/time, namely the number of milliseconds that have transpired since the epoch 1/1/1970 00:00:00 GMT.
<a href="#">toZulu</a>		
<code>&lt;DateTime&gt;.toZulu</code>	String	Returns an ISO-8601-compliant date-time as a String.

## 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
<b>Equals</b> (used as a comparison)		
<Number1> = <Number2>	Boolean	Returns a value of true if <Number1> is the same as <Number2>.
<b>Equals</b> (used as an assignment)		
<Number1> = <Number2>	Number	Assigns the value of <Number2> to the value of <Number1>.
<b>Not Equal To</b>		
<Number1> <> <Number2>	Boolean	Returns a value of true if <Number1> is not equal to <Number2>.
<b>Less than</b>		
<Number1> < <Number2>	Boolean	Returns a value of true if <Number1> is less than <Number2>.
<b>Greater than</b>		
<Number1> > <Number2>	Boolean	Returns a value of true if <Number1> is greater than <Number2>.
<b>Less than or Equal to</b>		
<Number1> <= <Number2>	Boolean	Returns a value of true if <Number1> is less than or equal to <Number2>.
<b>Greater than or Equal to</b>		
<Number1> >= <Number2>	Boolean	Returns a value of true if <Number1> is greater than or equal to <Number2>.
<b>In (Range)</b>		
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.
<b>In (List)</b>		

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.
<b>Add</b>		
<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 <a href="#">Precedence of rule operators</a> on page 221.
<b>Subtract</b>		
<Number1> - <Number2>	Number	Subtracts <Number2> from <Number1>. The resulting data type is the more expansive of either <Number1> or <Number2>. See <a href="#">Precedence of rule operators</a> on page 221.
<b>Multiply</b>		
<Number1> * <Number2>	Number	Returns the product of <Number1> and <Number2>. The resulting data type is the more expansive of either <Number1> or <Number2>. See <a href="#">Precedence of rule operators</a> on page 221.
<b>Divide</b>		
<Number1> / <Number2>	Number	Divides <Number1> by <Number2>. The resulting data type is the more expansive of either <Number1> or <Number2>. See <a href="#">Precedence of rule operators</a> on page 221.
<b>Exponent</b>		
<Number1> ** <Number2>	Number	Raises <Number1> to the power of <Number2>. The resulting data type is the more expansive of either <Number1> or <Number2>. See <a href="#">Precedence of rule operators</a> on page 221.
<b>Increment</b>		
<Number1> += <Number2>	Number	Increments <Number1> by <Number2>. The data type of <Number1> must accommodate the addition of <Number2>. See <a href="#">Precedence of rule operators</a> on page 221.
<b>Decrement</b>		

Name and Syntax	Returns	Description
<code>&lt;Number1&gt; -= &lt;Number2&gt;</code>	Number	Decrements <code>&lt;Number1&gt;</code> by the value of <code>&lt;Number2&gt;</code> . The data type of <code>&lt;Number1&gt;</code> must accommodate the addition of <code>&lt;Number2&gt;</code> . See <a href="#">Precedence of rule operators</a> on page 221.
<a href="#">Absolute Value</a>		
<code>&lt;Decimal&gt;.absVal</code>	Decimal	Returns the absolute value of <code>&lt;Number&gt;</code> . If the <code>&lt;Number&gt;</code> is positive, <code>&lt;Number&gt;</code> itself is returned; if <code>&lt;Number&gt;</code> is negative, the negation of <code>&lt;Number&gt;</code> is returned.
<a href="#">Floor</a>		
<code>&lt;Decimal&gt;.floor</code>	Integer	Returns the largest (closest to positive infinity) Integer that is not greater than <code>&lt;Number&gt;</code> .
<a href="#">Round</a>		
<code>&lt;Decimal&gt;.round</code>	Decimal	Rounds <code>&lt;Decimal&gt;</code> to the nearest Integer.
<a href="#">Round(n)</a>		
<code>&lt;Decimal&gt;.round(&lt;Integer&gt;)</code>	Decimal	Rounds <code>&lt;Decimal&gt;</code> to the number of decimal places specified by <code>&lt;Integer&gt;</code> .
<a href="#">To Integer</a>		
<code>&lt;Decimal&gt;.toInteger</code>	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.
<a href="#">To String</a>		
<code>&lt;Decimal&gt;.toString</code>	String	Converts an attribute of type Decimal to type string
<a href="#">Maximum Value</a>		
<code>&lt;Decimal&gt;.max(&lt;Number&gt;)</code>	Number	Returns the greater of <code>&lt;Decimal&gt;</code> and <code>&lt;Number&gt;</code> .
<a href="#">Minimum Value</a>		
<code>&lt;Decimal&gt;.min(&lt;Number&gt;)</code>	Number	Returns the lesser of <code>&lt;Decimal&gt;</code> and <code>&lt;Number&gt;</code> .
<a href="#">Logarithm (base 10)</a>		
<code>&lt;Decimal&gt;.log</code>	Decimal	Returns the logarithm (base 10) of <code>&lt;Decimal&gt;</code> . <code>&lt;Decimal&gt;</code> may not be zero.
<a href="#">Logarithm (base x)</a>		

Name and Syntax	Returns	Description
<code>&lt;Decimal1&gt;.log(&lt;Decimal2&gt;)</code>	Decimal	Returns the logarithm (base <code>&lt;Decimal2&gt;</code> ) of <code>&lt;Decimal1&gt;</code> . <code>&lt;Decimal1&gt;</code> may not be zero.
<a href="#">Natural Logarithm</a>		
<code>&lt;Decimal&gt;.ln</code>	Decimal	Returns the logarithm (base e) of <code>&lt;Decimal&gt;</code> . <code>&lt;Decimal&gt;</code> may not be zero.
<a href="#">Random</a>		
<code>&lt;Decimal&gt;.random</code>	Decimal	Returns a random decimal between <code>minRange</code> and <code>maxRange</code> .
<a href="#">truncate</a>		
<code>&lt;Decimal&gt;.truncate</code>	Integer	Truncates "this" Decimal value to an integer by removing the fractional portion.
<a href="#">fraction</a>		
<code>&lt;Decimal&gt;.fraction</code>	Decimal	Extracts the fraction portion of "this" Decimal.
<a href="#">movePoint(places)</a>		
<code>&lt;Decimal&gt;.movePoint (places:Integer)</code>	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
<a href="#">Equals</a> (used as a comparison)		
<code>&lt;Number1&gt; = &lt;Number2&gt;</code>	Boolean	Returns a value of true if <code>&lt;Number1&gt;</code> is the same as <code>&lt;Number2&gt;</code> .
<a href="#">Equals</a> (used as an assignment)		
<code>&lt;Number1&gt; = &lt;Number2&gt;</code>	Number	Assigns the value of <code>&lt;Number2&gt;</code> to the value of <code>&lt;Number1&gt;</code> . The data type of <code>&lt;Number1&gt;</code> must be expansive enough to accommodate <code>&lt;Number2&gt;</code> .
<a href="#">Not Equal To</a>		

Name and Syntax	Returns	Description
<Number1> <> <Number2>	Boolean	Returns a value of true if <Number1> is not equal to <Number2>.
<a href="#">Less than</a>		
<Number1> < <Number2>	Boolean	Returns a value of true if <Number1> is less than <Number2>.
<a href="#">Greater than</a>		
<Number1> > <Number2>	Boolean	Returns a value of true if <Number1> is greater than <Number2>.
<a href="#">Less than or Equal to</a>		
<Number1> <= <Number2>	Boolean	Returns a value of true if <Number1> is less than or equal to <Number2>.
<a href="#">Greater than or Equal to</a>		
<Number1> >= <Number2>	Boolean	Returns a value of true if <Number1> is greater than or equal to <Number2>.
<a href="#">In (Range)</a>		
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.
<a href="#">In (List)</a>		
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.
<a href="#">Add</a>		
<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 <a href="#">Precedence of rule operators</a> on page 221.
<a href="#">Subtract</a>		

Name and Syntax	Returns	Description
<Number1> - <Number2>	Number	Subtracts <Number2> from <Number1>. The resulting data type is the more expansive of either <Number1> or <Number2>. See <a href="#">Precedence of rule operators</a> on page 221.
<a href="#">Multiply</a>		
<Number1> * <Number2>	Number	Returns the product of <Number1> and <Number2>. The resulting data type is the more expansive of either <Number1> or <Number2>. See <a href="#">Precedence of rule operators</a> on page 221.
<a href="#">Divide</a>		
<Number1> / <Number2>	Number	Divides <Number1> by <Number2>. The resulting data type is the more expansive of either <Number1> or <Number2>. See <a href="#">Precedence of rule operators</a> on page 221.
<a href="#">Increment</a>		
<Number1> += <Number2>	Number	Increments <Number1> by <Number2>. The data type of <Number1> must accommodate the addition of <Number2>. See <a href="#">Precedence of rule operators</a> on page 221.
<a href="#">Decrement</a>		
<Number1> -= <Number2>	Number	Decrements <Number1> by the value of <Number2>. The data type of <Number1> must accommodate the addition of <Number2>. See <a href="#">Precedence of rule operators</a> on page 221.
<a href="#">Absolute value</a> on page 51.		
<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.
<a href="#">To Decimal</a>		
<Integer>.toDecimal	Decimal	Converts an attribute of type Integer to type Decimal.
<a href="#">To String</a>		
<Integer>.toString	String	Converts an attribute of type Integer to type String.
<a href="#">Maximum Value</a>		

Name and Syntax	Returns	Description
<code>&lt;Integer1&gt;.max(&lt;Integer2&gt;)</code>	Integer	Returns the greater of <code>&lt;Integer1&gt;</code> and <code>&lt;Integer2&gt;</code> .
<b>Minimum Value</b>		
<code>&lt;Integer1&gt;.min(&lt;Integer2&gt;)</code>	Integer	Returns the lesser of <code>&lt;Integer1&gt;</code> and <code>&lt;Integer2&gt;</code> .
<b>Div</b>		
<code>&lt;Integer1&gt;.div(&lt;Integer2&gt;)</code>	Integer	Returns the whole number of times that <code>&lt;Integer2&gt;</code> fits within <code>&lt;Integer1&gt;</code> - any remainder is discarded.
<b>Mod</b>		
<code>&lt;Integer1&gt;.mod(&lt;Integer2&gt;)</code>	Integer	Returns the whole number remainder that results from dividing <code>&lt;Integer1&gt;</code> by <code>&lt;Integer2&gt;</code> . If the remainder is a fraction, then zero is returned.
<b>Logarithm (base 10)</b>		
<code>&lt;Integer&gt;.log</code>	Decimal	Returns the logarithm (base 10) of <code>&lt;Integer&gt;</code> . <code>&lt;Integer&gt;</code> may not be zero.
<b>Logarithm (base x)</b>		
<code>&lt;Integer&gt;.log(&lt;Decimal&gt;)</code>	Decimal	Returns the logarithm (base <code>&lt;Decimal&gt;</code> ) of <code>&lt;Integer&gt;</code> . <code>&lt;Integer&gt;</code> may not be zero.
<b>Natural Logarithm</b>		
<code>&lt;Integer&gt;.ln</code>	Decimal	Returns the natural logarithm (base e) of <code>&lt;Number&gt;</code> . <code>&lt;Integer&gt;</code> may not be zero.
<b>Random</b>		
<code>&lt;Integer&gt;.random</code>	Integer	Returns a random integer between <code>minRange</code> and <code>maxRange</code> .
<b>isProbablePrime(certainty)</b>		
<code>&lt;Integer&gt;.isProbablePrime (certainty: Integer)</code>	Boolean	Returns true if this Integer is probably prime; false if definitely is not prime.
<b>gcd(val)</b>		
<code>&lt;Integer&gt;.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> .

Name and Syntax	Returns	Description
negate		
<Integer>.negate	Integer	Returns the negative value of this integer.

## String

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

Name and Syntax	Returns	Description
<a href="#">Equals</a> (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 <a href="#">Character precedence in Unicode and Java Collator</a> on page 217 for character precedence.
<a href="#">Equals</a> (used as an assignment)		
<String1> = <String2>	String	Assigns the value of <String2> to the value of <String1>.
<a href="#">Not Equal to</a>		
<String1> <> <String2>	Boolean	Returns a value of true if <String1> is not equal to <String2>.
<a href="#">Less than</a>		
<String1> < <String2>	Boolean	Returns a value of true if <String1> is less than <String2>. See <a href="#">Character precedence in Unicode and Java Collator</a> on page 217 for character precedence.
<a href="#">Greater than</a> on page 104		
<String1> > <String2>	Boolean	Returns a value of true if <String1> is greater than <String2>. See <a href="#">Character precedence in Unicode and Java Collator</a> on page 217 for character precedence.
<a href="#">Less than or Equal to</a>		
<String1> <= <String2>	Boolean	Returns a value of true if <String1> is less than or equal to <String2>. See <a href="#">Character precedence in Unicode and Java Collator</a> on page 217 for character precedence.
<a href="#">Greater than or Equal to</a>		

	Name and Syntax	Returns	Description
	<code>&lt;String1&gt; &gt;= &lt;String2&gt;</code>	Boolean	Returns a value of true if <code>&lt;String1&gt;</code> is greater than or equal to <code>&lt;String2&gt;</code> . See <a href="#">Character precedence in Unicode and Java Collator</a> on page 217 for character precedence.
<a href="#">In (Range)</a>			
	<code>attributeReference in [   (rangeExpression)   ]</code>	Boolean	Returns a value of true if <code>attributeReference</code> 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.
<a href="#">In (List)</a>			
	<code>attributeReference in {listExpression}</code>	Boolean	Returns a value of true if <code>attributeReference</code> is in the comma-delimited list of literal values, defined enumeration values, or - if in use - enumeration labels.
<a href="#">Adding Strings</a>			
	<code>&lt;String1&gt; + &lt;String2&gt;</code>	String	Concatenates <code>&lt;String1&gt;</code> to <code>&lt;String2&gt;</code> . Alternative syntax.
<a href="#">Size</a>			
	<code>&lt;String&gt;.size</code>	String	Returns the number of characters in <code>&lt;String&gt;</code> .
<a href="#">Concatenate</a>			
	<code>&lt;String1&gt;.concat(&lt;String2&gt;)</code>	String	Concatenates <code>&lt;String1&gt;</code> to <code>&lt;String2&gt;</code> .
<a href="#">Uppercase</a>			
	<code>&lt;String&gt;.toUpper</code>	String	Converts all characters <code>&lt;String&gt;</code> to uppercase.
<a href="#">Lowercase</a>			
	<code>&lt;String&gt;.toLowerCase</code>	String	Converts all characters in <code>&lt;String&gt;</code> to lowercase.
<a href="#">To DateTime</a>			
	<code>&lt;String&gt;.toDateTime</code>	DateTime	Converts the value in <code>&lt;String&gt;</code> to data type DateTime ONLY if all characters in <code>&lt;String&gt;</code> correspond to a valid DateTime mask (format)
<a href="#">To Decimal</a>			

	Name and Syntax	Returns	Description
	<code>&lt;String&gt;.toDecimal</code>	Decimal	Converts an attribute of type String to data type Decimal ONLY if all characters in <code>&lt;String&gt;</code> 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.
<b>To Integer</b>			
	<code>&lt;String&gt;.toInteger</code>	Integer	Converts an attribute of type String to type Integer ONLY if all characters in <code>&lt;String&gt;</code> are numeric. If any non-numeric characters are present, no value is returned.
<b>Substring</b>			
	<code>&lt;String&gt;.substring (&lt;Integer1&gt;,&lt;Integer2&gt;)</code>	String	Returns that portion of <code>&lt;String&gt;</code> between character positions <code>&lt;Integer1&gt;</code> and <code>&lt;Integer2&gt;</code> .
<b>Equals Ignoring Case</b>			
	<code>&lt;String1&gt;.equalsIgnoreCase (&lt;String2&gt;)</code>	Boolean	Returns a value of true if <code>&lt;String1&gt;</code> is the same as <code>&lt;String2&gt;</code> , irrespective of case.
<b>Starts with</b>			
	<code>&lt;String1&gt;.startsWith (&lt;String2&gt;)</code>	Boolean	Returns a value of true if the <code>&lt;String1&gt;</code> begins with the characters specified in <code>&lt;String2&gt;</code> .
<b>Ends with</b>			
	<code>&lt;String1&gt;.endsWith (&lt;String2&gt;)</code>	Boolean	Evaluates the contents of <code>&lt;String1&gt;</code> and returns a value of true if the String ends with the characters specified in <code>&lt;String2&gt;</code> .
<b>Contains</b>			
	<code>&lt;String1&gt;.contains (&lt;String2&gt;)</code>	Boolean	Evaluates the contents of <code>&lt;String1&gt;</code> and returns a value of true if it contains the exact characters defined by <code>&lt;String2&gt;</code>
<b>Equals</b>			
	<code>&lt;String1&gt;.equals (&lt;String2&gt;)</code>	Boolean	Returns a value of true if <code>&lt;String1&gt;</code> is the same as <code>&lt;String2&gt;</code> .
<b>Index Of</b>			

	Name and Syntax	Returns	Description
	<code>&lt;String1&gt;.indexOf (&lt;String2&gt;)</code>	Integer	Returns the beginning character position number of <code>&lt;String2&gt;</code> within <code>&lt;String1&gt;</code> , if <code>&lt;String1&gt;</code> contains <code>&lt;String2&gt;</code> . If it does not, the function returns a value of zero.
<a href="#">Replace String</a>			
	<code>&lt;String&gt;.replace (&lt;String1&gt;, &lt;String2&gt;)</code>	String	Returns a new String where the instances of the String to be replaced are replaced by the value of the replacement String.
<a href="#">Regular expression replace String</a>			
	<code>&lt;String&gt;.replaceAll (&lt;regularExpression&gt;, &lt;replacementString&gt;)</code>	String	Returns a new String where the Strings matching the regular expression are replaced by the replacement String.
<a href="#">Matches</a>			
	<code>&lt;String&gt;.matches (&lt;regularExpression&gt;:String)</code>	Boolean	Returns true if the regular expression matches the String.
<a href="#">containsBlanks</a>			
	<code>&lt;String&gt;.containsBlanks</code>	Boolean	Determines whether the specified String contains any blanks.
<a href="#">characterAt(index)</a>			
	<code>&lt;String&gt;.characterAt (index: Integer)</code>	String	Returns the character at the specified position in the String.
<a href="#">isInteger</a>			
	<code>&lt;String&gt;.isInteger</code>	Boolean	Determines whether "this" String contains only integer digits. <hr/> <b>Note:</b> 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.) <hr/>
<a href="#">trimSpaces</a>			

	Name and Syntax	Returns	Description
	<code>&lt;String&gt;.trimSpaces</code>	String	Trims leading and trailing spaces from "this" String.
<code>charsIn(validSet)</code>			
	<code>&lt;String&gt;.charsIn(validSet:String)</code>	Boolean	Determines whether "this" String contains only characters specified in the validSet.

## Time

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

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

Name and Syntax	Returns	Description
<b>In (Range)</b>		
<code>attributeReference in [   (rangeExpression)   ]</code>	Boolean	Returns a value of true if <code>attributeReference</code> 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.
<b>In (List)</b>		
<code>attributeReference in {listExpression}</code>	Boolean	Returns a value of true if <code>attributeReference</code> is in the comma-delimited list of literal values, defined enumeration values, or - if in use - enumeration labels.
<b>Hour</b>		
<code>&lt;Time&gt;.hour</code>	Integer	Returns the hour portion of <code>&lt;Time&gt;</code> . The returned value is based on a 24-hour clock.
<b>Minute</b>		
<code>&lt;Time&gt;.min</code>	Integer	Returns the minute portion of <code>&lt;Time&gt;</code> as an Integer between 0 and 59
<b>Second</b>		
<code>&lt;Time&gt;.sec</code>	Integer	Returns the seconds portion of <code>&lt;Time&gt;</code> as an Integer between 0 and 59
<b>Add hours</b>		
<code>&lt;Time&gt;.addHours (&lt;Integer&gt;)</code>	Date	Adds the number of hours in <code>&lt;Integer&gt;</code> to the number of hours in the Time portion of <code>&lt;Time&gt;</code>
<b>Add minutes</b>		
<code>&lt;Time&gt;.addMinutes (&lt;Integer&gt;)</code>	Date	Adds the number of minutes in <code>&lt;Integer&gt;</code> to the number of minutes in the Time portion of <code>&lt;Time&gt;</code>
<b>Add seconds</b>		
<code>&lt;Time&gt;.addSeconds (&lt;Integer&gt;)</code>	Date	Adds the number of seconds in <code>&lt;Integer&gt;</code> to the number of seconds in the Time portion of <code>&lt;Time&gt;</code>
<b>Hours between</b>		

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

## Entity and 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
<b>New</b>		
<code>&lt;Entity&gt; .new [&lt;Expression1&gt;,...]</code>	Entity	Creates a new instance of <code>&lt;Entity&gt;</code> . Expressions (optional to assign attribute values) in square brackets <code>[..]</code> must be written in the form: <i>attribute = value</i> .
<b>New Unique</b>		
<code>&lt;Entity&gt; .newUnique [&lt;Expression1&gt;,...]</code>	Entity	Creates a new instance of <code>&lt;Entity&gt;</code> only if the instance created is unique as defined by optional <code>&lt;Expression1&gt;,...</code>
<b>Clone</b>		
<code>&lt;Entity&gt;.clone [&lt;Expression1&gt;,...]</code>	Entity	Creates a new instance of <code>&lt;Entity&gt;</code> with the same attributes and their respective values. Expressions (optional to override attribute values) in square brackets <code>[..]</code> must be written in the form: <i>attribute = value</i> .
<b>Remove</b>		
<code>&lt; Entity&gt;.remove [(true)   (false)]</code>	Entity	Deletes the entity from memory and from the resultant XML document. Children can be removed as well when set to <code>(true)</code> , or retained after moving to root <code>(false)</code> . Blank or no value defaults to <code>true</code> .

## Collection

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

Name and Syntax	Returns	Description
<b>Replace element(s)</b>		
<code>&lt;Collection1&gt; = &lt;Collection2&gt;</code> <code>&lt;Collection1&gt; = &lt;Entity&gt;</code>	<i>modifies a collection</i>	replaces all elements in <code>&lt;Collection1&gt;</code> with elements of <code>&lt;Collection2&gt;</code> or with <code>&lt;Entity&gt;</code> , provided the new associations are allowed by the Business Vocabulary.
<b>Associate element(s)</b>		

Name and Syntax	Returns	Description
<code>&lt;Collection1&gt; += &lt;Collection2&gt;</code> <code>&lt;Collection1&gt; += &lt;Entity&gt;</code>	<i>modifies a collection</i>	Associates all elements of <code>&lt;Collection2&gt;</code> or <code>&lt;Entity&gt;</code> with <code>&lt;Collection1&gt;</code> . Every <code>&lt;Collection&gt;</code> must be expressed as a unique alias.
<b>Disassociate element(s)</b>		
<code>&lt;Collection1&gt; -= &lt;Collection2&gt;</code>	<i>modifies a collection</i>	Disassociates all elements of <code>&lt;Collection2&gt;</code> from <code>&lt;Collection1&gt;</code> . Does not delete the disassociated elements. Every <code>&lt;Collection&gt;</code> must be expressed as a unique alias.
<b>Is empty</b>		
<code>&lt;Collection&gt; -&gt;isEmpty</code>	Boolean	Returns a value of true if <code>&lt;Collection&gt;</code> contains <i>no</i> elements
<b>Not empty</b>		
<code>&lt;Collection&gt; -&gt;notEmpty</code>	Boolean	Returns a value of true if <code>&lt;Collection&gt;</code> contains <i>at least one</i> element.
<b>Exists</b>		
<code>&lt;Collection&gt; -&gt;exists (&lt;Expression&gt;)</code>	Boolean	Returns a value of true if <code>&lt;Expression&gt;</code> holds true for <i>at least one</i> element of <code>&lt;Collection&gt;</code>
<b>For all</b>		
<code>&lt;Collection&gt; -&gt;forAll (&lt;Expression&gt;)</code>	Boolean	Returns a value of true if <i>every</i> <code>&lt;Expression&gt;</code> holds true for <i>every</i> element of <code>&lt;Collection&gt;</code>
<b>Sorted by</b>		
<code>&lt;Collection&gt; -&gt;sortedBy (&lt;Attribute&gt;)</code>	<i>converts a collection into a sequence</i>	Sequences the elements of <code>&lt;Collection&gt;</code> in <u>ascending</u> order, using the value of <code>&lt;Attribute&gt;</code> as the index. <code>&lt;Collection&gt;</code> must be expressed as a unique alias.
<b>Sorted by descending</b>		
<code>&lt;Collection&gt; -&gt;sortedByDesc (&lt;Attribute&gt;)</code>	<i>converts a collection into a sequence</i>	Sequences the elements of <code>&lt;Collection&gt;</code> in <u>descending</u> order, using the value of <code>&lt;Attribute&gt;</code> as the index. <code>&lt;Collection&gt;</code> must be expressed as a unique alias.
<b>Iterate</b>		
<code>&lt;Collection&gt; -&gt;iterate (&lt;Expression&gt;)</code>		Executes <code>&lt;Expression&gt;</code> for every element in <code>&lt;Collection&gt;</code> . <code>&lt;Collection&gt;</code> must be expressed as a unique alias.

Name and Syntax	Returns	Description
<b>Size of collection</b>		
<code>&lt;Collection&gt; -&gt;size</code>	Integer	Returns the number of elements in <code>&lt;Collection&gt;</code> . <code>&lt;Collection&gt;</code> must be expressed as a unique alias.
<b>Sum</b>		
<code>&lt;Collection.attribute&gt; -&gt;sum</code>	Number	Sums the values of the specified <code>&lt;attribute&gt;</code> for all elements in <code>&lt;Collection&gt;</code> . <code>&lt;attribute&gt;</code> must be a numeric data type.
<b>Average</b>		
<code>&lt;Collection.attribute&gt; -&gt;avg</code>	Number	Averages all of the specified attributes in <code>&lt;Collection&gt;</code> . <code>&lt;Collection&gt;</code> must be expressed as a unique alias. <code>&lt;attribute&gt;</code> must be a numeric data type
<b>Minimum</b>		
<code>&lt;Collection.attribute&gt; -&gt;min</code>	Number	Returns the lowest value of <code>&lt;attribute&gt;</code> for all elements in <code>&lt;Collection&gt;</code> . <code>&lt;attribute&gt;</code> must be a numeric data type
<b>Maximum</b>		
<code>&lt;Collection.attribute&gt; -&gt;max</code>	Number	Returns the highest value of <code>&lt;attribute&gt;</code> for all elements in <code>&lt;Collection&gt;</code> . <code>&lt;attribute&gt;</code> must be a numeric data type
<b>toSet</b>		
<code>Collection.toSet</code>	String	Returns a single String that is the set of Strings in this collection.
<b>allContain(lookFor)</b>		
<code>Collection.allContain (lookFor:String)</code>	Boolean	Determines whether all the strings in this collection contain the <code>lookFor</code> String
<b>uniqueCount</b>		
<code>Collection.uniqueCount</code>	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
<b>At</b>			
	<code>&lt;Sequence&gt; -&gt;at(&lt;Integer&gt;)</code>	Entity	Returns the element at position <code>&lt;Integer&gt;</code> . <code>&lt;Sequence&gt;</code> must be expressed as a unique alias.
<b>First</b>			
	<code>&lt;Sequence&gt; -&gt;first</code>	Entity	Returns the first element of <code>&lt;Sequence&gt;</code> . <code>&lt;Sequence&gt;</code> must be expressed as a unique alias.
<b>Last</b>			
	<code>&lt;Sequence&gt; -&gt;last</code>	Entity	Returns the last element of <code>&lt;Sequence&gt;</code> . <code>&lt;Sequence&gt;</code> must be expressed as a unique alias.
<b>SubSequence</b>			
	<code>&lt;Sequence&gt; -&gt;subSequence(<i>integer1</i>, <i>integer2</i>)</code>	Entity	Returns a Sequence containing all elements of <code>&lt;Sequence&gt;</code> between the positions <code><i>integer1</i></code> and <code><i>integer2</i></code> .
<b>First(number)</b>			

	Name and Syntax	Returns	Description
	<code>&lt;Sequence&gt; -&gt;first(integer)</code>	Entity	Returns a Sequence containing elements of <code>&lt;Sequence&gt;</code> from the first element to <code>integer</code> ; in other words, <code>-&gt;first(x)</code> is effectively <code>&gt;subSequence(1,x)</code>
<a href="#">Last(number)</a>			
	<code>&lt;Sequence&gt; -&gt;last(integer)</code>	Entity	Returns a Sequence containing elements of <code>&lt;Sequence&gt;</code> between the end position of the collection and <code>integer</code> ; in other words, in a sequence of <code>n</code> elements, <code>-&gt;last(x)</code> is effectively <code>&gt;subSequence(n-x+1,n)</code>
<a href="#">Trend</a>			
	<code>&lt;Attribute&gt; -&gt; &lt;Sequence&gt;.trend</code>	String	Returns a 4-character string, INCR, DECR, CNST, or NONE depending on the trend of <code>&lt;Attribute&gt;</code> within <code>&lt;Sequence&gt;</code> .
<a href="#">mavg(elements)</a>			
	<code>&lt;Sequence.decimal&gt; .mavg(elements:Integer)</code>	Decimal	Returns a single decimal value that is the average of the number of elements specified.
<a href="#">Sorted Alias: next</a>			
	<code>-&gt;next</code>		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 <a href="#">Sorted aliases "Sorted Alias" in the Collections chapter of the Corticon Studio: Rule Modeling Guide.</a>

## 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
<a href="#">Null</a>		

Name and Syntax	Returns	Description
null	<i>none</i>	The null value corresponds to one of three different scenarios: <ul style="list-style-type: none"> <li>the absence of an attribute in a Ruletest scenario</li> <li>the absence of data for an attribute in a Ruletest scenario</li> <li>an object that has a value of null</li> </ul>
<b>True</b>		
true or T	Boolean	Represents Boolean value true
<b>False</b>		
false or F	Boolean	Represents the Boolean value false
<b>Other</b>		
other	<i>any</i>	When included in a condition's Values set, other represents any value not explicitly included in the set, including <a href="#">null</a> .
<b>CellValue</b>		
cellValue	<i>any</i>	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
<b>Now</b>		
now	Date	Returns the current system date and time when the rule is executed.
<b>Today</b>		
today	Date	Returns the current system date when the rule is executed.

---

## 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 elements
- At
- Average
- CellValue
- Clone
- Concatenate
- Character at
- Contains
- Day
- Days between
- Day of week
- Day of year
- Decrement
- Disassociate elements
- 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
- Floor
- Get Milliseconds
- First
- First NUMBER
- For all
- Greater than

- 
- Greater than or equal to
  - Hour
  - Hour between
  - In LIST
  - In RANGE
  - Increment
  - Index of
  - Is integer
  - Is empty
  - Iterate
  - Last
  - Last NUMBER
  - Less than
  - Less than or equal to
  - Logarithm BASE 10
  - Logarithm BASE X
  - Lowercase
  - Matches
  - 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
- Random
- Remove element
- Replace elements
- Replace String
- Regular expression to replace String
- 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

- Trim spaces
- 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`

The screenshot shows a rulesheet editor window titled 'AbsoluteValue.ers'. It contains a table with 'Conditions' and 'Actions' sections, and a 'Rule Statements' table at the bottom.

Conditions		0
a		
b		
Actions		<
Post Message(s)		
A	Entity1.decimal1 = Entity1.decimal2.absVal	<input checked="" type="checkbox"/>
B		
Overrides		

Ref	ID	Post	Alias	Text
A0				decimal1 equals the absolute value of decimal2

### SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>decimal2 [0.000000]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>decimal2 [23.000000]</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>decimal2 [-17.000000]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [0.000000]</li> <li>decimal2 [0.000000]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>decimal1 [23.000000]</li> <li>decimal2 [23.000000]</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>decimal1 [17.000000]</li> <li>decimal2 [-17.000000]</li> </ul> </li> </ul>

## 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 221.

### 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`

Conditions		0	1
a			
b			
Actions		<	
	Post Message(s)		
A	Entity1.decimal1 = Entity1.decimal2 + Entity1.integer1	<input checked="" type="checkbox"/>	
B			
Overrides			
Rule Statements			
Ref	ID	Post	Text
A0			decimal1 equals the sum of decimal2 plus integer1

## 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.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]           <ul style="list-style-type: none"> <li>decimal2 [1000.000000]</li> <li>integer1 [300]</li> </ul> </li> <li>Entity1 [2]           <ul style="list-style-type: none"> <li>decimal2 [500.000000]</li> <li>integer1 [300]</li> </ul> </li> <li>Entity1 [3]           <ul style="list-style-type: none"> <li>decimal2 [1550.000000]</li> <li>integer1 [300]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]           <ul style="list-style-type: none"> <li><b>decimal1 [1300.000000]</b></li> <li>decimal2 [1000.000000]</li> <li>integer1 [300]</li> </ul> </li> <li>Entity1 [2]           <ul style="list-style-type: none"> <li><b>decimal1 [800.000000]</b></li> <li>decimal2 [500.000000]</li> <li>integer1 [300]</li> </ul> </li> <li>Entity1 [3]           <ul style="list-style-type: none"> <li><b>decimal1 [1850.000000]</b></li> <li>decimal2 [1550.000000]</li> <li>integer1 [300]</li> </ul> </li> </ul>

## 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`

Conditions		0
a		
b		
Actions		<
Post Message(s)		
A	Entity1.string1 = 'AAA' + Entity1.string2 + 'ZZZ'	<input checked="" type="checkbox"/>
B		
Overrides		

Ref	ID	Post	Alias	Text
A0				string1 equals string2, prepended with 'ZZZ'

## SAMPLE RULETEST

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string2 [Hello]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string2 [-Goodbye-]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string2 [Au Revoir]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [AAAHelloZZZ]</li> <li>string2 [Hello]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [AAA-Goodbye-ZZZ]</li> <li>string2 [-Goodbye-]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [AAAAu RevoirZZZ]</li> <li>string2 [Au Revoir]</li> </ul> </li> </ul>

## 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`.

Conditions		0
a		
b		
Actions		<
Post Message(s)		
A	Entity1.dateTime1 = Entity1.dateTime2.addDays(45)	<input checked="" type="checkbox"/>
B		
Overrides		

Ref	ID	Post	Alias	Text
A0				dateTime1 must be given a value 45 days after dateTime2

**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.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>dateTime2 [5/14/2020 2:00:00 PM]</li> </ul> </li> <li>Entity1 [2]                             <ul style="list-style-type: none"> <li>dateTime2 [08/07/2006 3:00:00 PM EST]</li> </ul> </li> <li>Entity1 [3]                             <ul style="list-style-type: none"> <li>dateTime2 [2019/12/25 5:00:00 AM]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>dateTime1 [6/28/2020 2:00:00 PM]</li> <li>dateTime2 [5/14/2020 2:00:00 PM]</li> </ul> </li> <li>Entity1 [2]                             <ul style="list-style-type: none"> <li>dateTime1 [09/21/2006 3:00:00 PM EST]</li> <li>dateTime2 [08/07/2006 3:00:00 PM EST]</li> </ul> </li> <li>Entity1 [3]                             <ul style="list-style-type: none"> <li>dateTime1 [2020/02/08 5:00:00 AM]</li> <li>dateTime2 [2019/12/25 5:00:00 AM]</li> </ul> </li> </ul>

# 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`.

Conditions		0	1
a			
b			
Actions		<	
Post Message(s)			
A	Entity1.dateTime1=Entity1.dateTime2.addHours(30)	<input checked="" type="checkbox"/>	
B			
Overrides			

Ref	ID	Post	Alias	Text
A0				dateTime1 must be given a value 30 hours after dateTime2

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]           <ul style="list-style-type: none"> <li>dateTime2 [5/14/2020 2:00:00 PM]</li> </ul> </li> <li>Entity1 [2]           <ul style="list-style-type: none"> <li>dateTime2 [08/07/2006 3:00:00 PM EST]</li> </ul> </li> <li>Entity1 [3]           <ul style="list-style-type: none"> <li>dateTime2 [2019/12/25 5:00:00 AM]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]           <ul style="list-style-type: none"> <li>dateTime1 [5/15/2020 8:00:00 PM]</li> <li>dateTime2 [5/14/2020 2:00:00 PM]</li> </ul> </li> <li>Entity1 [2]           <ul style="list-style-type: none"> <li>dateTime1 [08/08/2006 9:00:00 PM EST]</li> <li>dateTime2 [08/07/2006 3:00:00 PM EST]</li> </ul> </li> <li>Entity1 [3]           <ul style="list-style-type: none"> <li>dateTime1 [2019/12/26 11:00:00 AM]</li> <li>dateTime2 [2019/12/25 5:00:00 AM]</li> </ul> </li> </ul>

## 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`.

Conditions		0	1
a			
b			
Actions		<	
Post Message(s)			
A	Entity1.dateTime1=Entity1.dateTime2.addMinutes(90)	<input checked="" type="checkbox"/>	
B			
Overrides			

Ref	ID	Post	Alias	Text
A0				dateTime1 must be given a value of 90 minutes after dateTime2

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateTime2 [5/14/2020 2:00:00 PM]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateTime2 [08/07/2006 3:00:00 PM EST]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>dateTime2 [2019/12/25 5:00:00 AM]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [5/14/2020 3:30:00 PM]</li> <li>dateTime2 [5/14/2020 2:00:00 PM]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [08/07/2006 4:30:00 PM EST]</li> <li>dateTime2 [08/07/2006 3:00:00 PM EST]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>dateTime1 [2019/12/25 6:30:00 AM]</li> <li>dateTime2 [2019/12/25 5:00:00 AM]</li> </ul> </li> </ul>

# 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`.

Conditions		0	1
a			
b			
Actions		<	
Post Message(s)			
A	Entity1.dateTime1=Entity1.dateTime2.addMonths(10)	<input checked="" type="checkbox"/>	
B			
Overrides			

Rule Statements				
Ref	ID	Post	Alias	Text
A0				dateTime1 must be given a value 10 months after dateTime2

## 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.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]           <ul style="list-style-type: none"> <li>dateTime2 [5/14/2020 2:00:00 PM]</li> </ul> </li> <li>Entity1 [2]           <ul style="list-style-type: none"> <li>dateTime2 [08/07/2006 3:00:00 PM EST]</li> </ul> </li> <li>Entity1 [3]           <ul style="list-style-type: none"> <li>dateTime2 [2019/12/25 5:00:00 AM]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]           <ul style="list-style-type: none"> <li>dateTime1 [3/14/2021 2:00:00 PM]</li> <li>dateTime2 [5/14/2020 2:00:00 PM]</li> </ul> </li> <li>Entity1 [2]           <ul style="list-style-type: none"> <li>dateTime1 [06/07/2007 3:00:00 PM EST]</li> <li>dateTime2 [08/07/2006 3:00:00 PM EST]</li> </ul> </li> <li>Entity1 [3]           <ul style="list-style-type: none"> <li>dateTime1 [2020/10/25 5:00:00 AM]</li> <li>dateTime2 [2019/12/25 5:00:00 AM]</li> </ul> </li> </ul>

# 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`.

Conditions		0
a		
b		
Actions		<
Post Message(s)		
A	Entity1.dateTime1=Entity1.dateTime2.addSeconds(90)	<input checked="" type="checkbox"/>
B		
		Overrides

Ref	ID	Post	Alias	Text
A0				dateTime1 must be given a value 90 seconds after dateTime2

## 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
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateTime2 [5/14/2020 2:00:00 PM]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateTime2 [08/07/2006 3:00:00 PM EST]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>dateTime2 [2019/12/25 5:00:00 AM]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [5/14/2020 2:01:30 PM]</li> <li>dateTime2 [5/14/2020 2:00:00 PM]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [08/07/2006 3:01:30 PM EST]</li> <li>dateTime2 [08/07/2006 3:00:00 PM EST]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>dateTime1 [2019/12/25 5:01:30 AM]</li> <li>dateTime2 [2019/12/25 5:00:00 AM]</li> </ul> </li> </ul>

# 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`.

Conditions		0	1
a			
b			

Actions		<
Post Message(s)		
A	Entity1.dateTime1=Entity1.dateTime2.addYears(10)	<input checked="" type="checkbox"/>
B		

Rule Statements				
Ref	ID	Post	Alias	Text
A0				dateOnly1 must be given a value 10 years after dateOnly2

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>dateTime2 [5/14/2020 2:00:00 PM]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>dateTime2 [08/07/2006 3:00:00 PM EST]</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>dateTime2 [2019/12/25 5:00:00 AM]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [5/14/2030 2:00:00 PM]</li> <li>dateTime2 [5/14/2020 2:00:00 PM]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [08/07/2016 3:00:00 PM EST]</li> <li>dateTime2 [08/07/2006 3:00:00 PM EST]</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>dateTime1 [2029/12/25 5:00:00 AM]</li> <li>dateTime2 [2019/12/25 5:00:00 AM]</li> </ul> </li> </ul>

## Associate elements

### 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 uniquely identified with an alias or role.

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.

Scope	Conditions	1	2
a	collection2.boolean1	T	F
b			
c			
Actions			
Post Message(s)			
A	collection1 += collection2	<input checked="" type="checkbox"/>	<input type="checkbox"/>
B			
C			
Overrides			

Ref	ID	Post	Alias	Text
A1				If boolean1 value of an element in collection2 is true, then add an element to collection1
A2				If boolean1 value of Entity2 is false, then take no action

### SAMPLE RULETEST: HIER

A sample Ruletest provides two examples of Entity2 with boolean1 values, and a single Entity1. Input and Output panels shows the association embedded in the parent entity:

### SAMPLE RULETEST: FLAT

Setting two properties in the Studio's brms.properties file enables a Flat payload:

```
com.corticon.tester.ccservice.execute.format=XML
com.corticon.designer.tester.xmlmessagingstyle=Flat
```

After restarting Studio, running the same sample Ruletest shows the association dropping to the root with an href entity:

Input	Output
Entity1 [1]	Entity1 [1]
Entity2 [1]	entity2 (Entity2) [1]
boolean1 [true]	Entity2 [1]
Entity2 [2]	boolean1 [true]
boolean1 [false]	Entity2 [2]
	boolean1 [false]

# 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 sorting syntax](#) "Advanced collection sorting syntax" in the *Rule Modeling Guide* for more examples of usage.

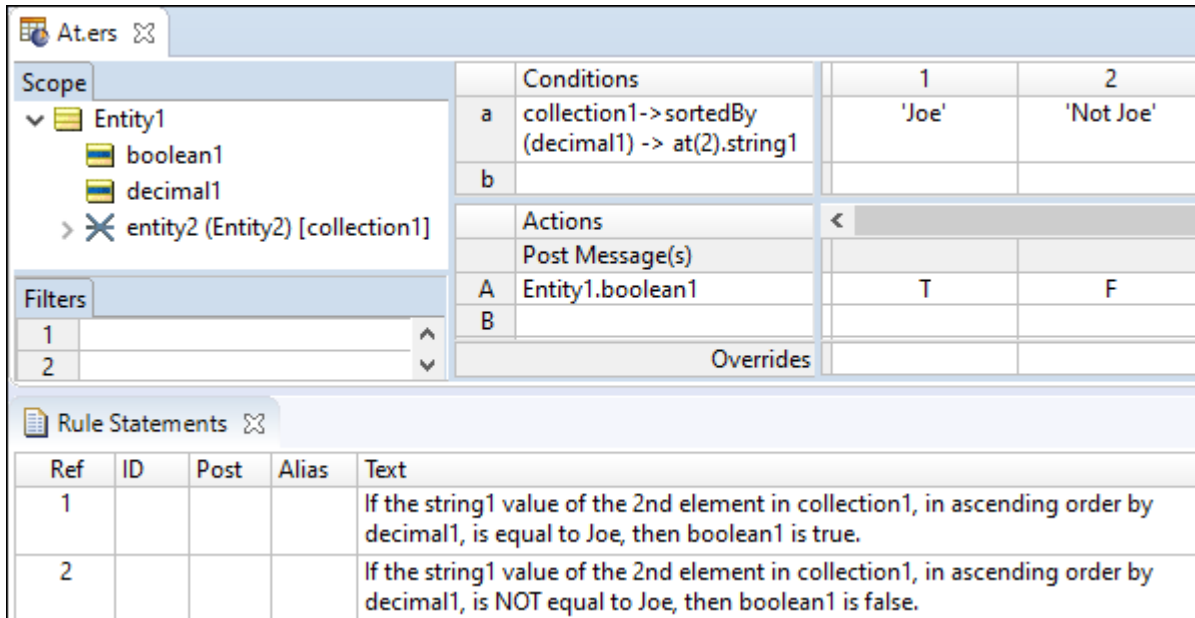
`<Attribute1>` may be of any data type.

## 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		Conditions	1	2
Entity1	a	collection1->sortedBy (decimal1) -> at(2).string1	'Joe'	'Not Joe'
	b			
Filters		Actions		
		Post Message(s)		
1		A	T	F
2		B		
		Overrides		

Ref	ID	Post	Alias	Text
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.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [2.500000]</li> <li>string1 [Sally]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [5.800000]</li> <li>string1 [Moe]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [3.300000]</li> <li>string1 [Joe]</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [2.500000]</li> <li>string1 [Sally]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [5.800000]</li> <li>string1 [Moe]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [3.300000]</li> <li>string1 [Joe]</li> </ul> </li> </ul> </li> </ul>

## 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

The screenshot shows the 'Average.ers' rulesheet editor. The 'Scope' panel on the left shows a tree view: Entity1 (expanded) contains decimal1 and entity2 (Entity2) [collection2] (expanded), which contains integer1. The 'Filters' panel shows two filter slots. The 'Actions' table has one row: 'Entity1.decimal1 = collection2.integer1->avg' with a checked checkbox. The 'Rule Statements' table at the bottom has one row: Ref 'A0', ID, Post, Alias, and Text 'decimal1 of Entity1 is equal to the average of all integer1 values in collection2'.

Ref	ID	Post	Alias	Text
A0				decimal1 of Entity1 is equal to the average of all integer1 values in collection2

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>integer1 [1520]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>integer1 [1300]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>integer1 [750]</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [1190.000000]</li> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>integer1 [1520]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>integer1 [1300]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>integer1 [750]</li> </ul> </li> </ul> </li> </ul>

# CellValue

## SYNTAX

Various, see Examples below

## DESCRIPTION

When used in an expression, **cellValue** performs text replacement where the 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.

CellValue1.ers				
Conditions		1	2	
a	Entity1.boolean1	T	F	
b				
Actions		<		
Post Message(s)				
A	Entity1.integer1 += cellValue	3	6	
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**:

CellValue2.ers					
Conditions		0	1	2	
a	Entity1.boolean1		T	F	
b					
c					
Actions		<			
Post Message(s)					
A	Entity1.integer1 += 3	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<input type="checkbox"/>
B	Entity1.integer1 += 6	<input type="checkbox"/>	<input type="checkbox"/>		<input checked="" type="checkbox"/>
C					
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.

Input	Output
<ul style="list-style-type: none"> <li>▼ Entity1 [1]                             <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>integer1 [2]</li> </ul> </li> <li>▼ Entity1 [2]                             <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>integer1 [4]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1]                             <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>integer1 [5]</li> </ul> </li> <li>▼ Entity1 [2]                             <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>integer1 [10]</li> </ul> </li> </ul>

## 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.

CellValue3.ers

Scope	Conditions	1	2
<ul style="list-style-type: none"> <li>▼ Entity1                                     <ul style="list-style-type: none"> <li>boolean1</li> <li>&gt; ✕ entity2 (Entity2) [collection1]</li> <li>&gt; Entity2</li> </ul> </li> </ul>	a	'David'	'Jennifer'
	b		
Actions		<	
Post Message(s)		✉	✉
A			
B			
Overrides			

Rule Statements

Ref	ID	Post	Alias	Text
1		Info	Entity1	We have a David
2		Info	Entity1	We have a Jennifer

## SAMPLE RULETEST 2

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>string1 [Jose]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>string1 [David]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>string1 [Clint]</li> </ul> </li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>string1 [Howard]</li> </ul> </li> <li>entity2 (Entity2) [5] <ul style="list-style-type: none"> <li>string1 [Emil]</li> </ul> </li> <li>entity2 (Entity2) [6] <ul style="list-style-type: none"> <li>string1 [Jennifer]</li> </ul> </li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>entity2 (Entity2) [7] <ul style="list-style-type: none"> <li>string1 [Doug]</li> </ul> </li> <li>entity2 (Entity2) [8] <ul style="list-style-type: none"> <li>string1 [Ling]</li> </ul> </li> <li>entity2 (Entity2) [9] <ul style="list-style-type: none"> <li>string1 [Frances]</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>string1 [Jose]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>string1 [David]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>string1 [Clint]</li> </ul> </li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>string1 [Howard]</li> </ul> </li> <li>entity2 (Entity2) [5] <ul style="list-style-type: none"> <li>string1 [Emil]</li> </ul> </li> <li>entity2 (Entity2) [6] <ul style="list-style-type: none"> <li>string1 [Jennifer]</li> </ul> </li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>entity2 (Entity2) [7] <ul style="list-style-type: none"> <li>string1 [Doug]</li> </ul> </li> <li>entity2 (Entity2) [8] <ul style="list-style-type: none"> <li>string1 [Ling]</li> </ul> </li> <li>entity2 (Entity2) [9] <ul style="list-style-type: none"> <li>string1 [Frances]</li> </ul> </li> </ul> </li> </ul>

Severity	Message	Entity
Info	We have a David	Entity1[1]
Info	We have a Jennifer	Entity1[2]

### 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.

Scope	Conditions	1	2
Entity1	a Entity1.boolean1	T	F
	b		
Actions		<	
Post Message(s)			
A	collection1 += Entity2.new [string1 = cellValue]	'Joe'	'David'
B			
Overrides			

Ref	ID	Post	Alias	Text
1				If boolean1 is true, create a new member of collection1 named Joe
2				If boolean1 is false, create a new member of collection1 named David

## 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.

Input	Output
<ul style="list-style-type: none"> <li>▼ Entity1 [1]           <ul style="list-style-type: none"> <li>boolean1 [true]               <ul style="list-style-type: none"> <li>entity2 (Entity2) [1]</li> </ul> </li> </ul> </li> <li>▼ Entity1 [2]           <ul style="list-style-type: none"> <li>boolean1 [false]               <ul style="list-style-type: none"> <li>entity2 (Entity2) [2]</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1]           <ul style="list-style-type: none"> <li>boolean1 [true]               <ul style="list-style-type: none"> <li>entity2 (Entity2) [1]                   <ul style="list-style-type: none"> <li>entity2 (Entity2) [3]                       <ul style="list-style-type: none"> <li>string1 [Joe]</li> </ul> </li> </ul> </li> </ul> </li> <li>▼ Entity1 [2]           <ul style="list-style-type: none"> <li>boolean1 [false]               <ul style="list-style-type: none"> <li>entity2 (Entity2) [2]                   <ul style="list-style-type: none"> <li>entity2 (Entity2) [4]                       <ul style="list-style-type: none"> <li>string1 [David]</li> </ul> </li> </ul> </li> </ul> </li> </ul> </li> </ul> </li></ul>

# 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.

---

**Null values in the attribute set expressions** - When there are any null attributes on the right hand side of a `clone` assignment expression, the assignment does not occur. For example, in `A.clone[attr=B.attr]` where `B.attr` is null, it will not override the value of the cloned `A.attr`. There are cases where the null result is preferred. In a Studio's `brms.properties` you can add the property `com.corticon.reactor.rulebuilder.DisableNullCheckingOnClone=true` so that the null checks are removed. In the example, the value of the cloned `A.attr` will be null. Be aware that using this setting on a Studio machine should be applied on any other machine that will work on a related project.

## 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.

Scope		Conditions		1
Entity1	a	Entity1.qtyOrdered > Entity1.qtyShipped		T
qtyBackordered	b			
qtyOrdered	c			
qtyShipped				
string1				
entity2 (Entity2)				
Filters		Actions		<
1			Post Message(s)	✉
2			A Entity1.qtyBackordered = Entity1.qtyOrdered - Entity1.qtyShipped	✓
			B Entity1.clone[qtyOrdered = qtyBackordered, qtyShipped=0, qtyBackordered=0, string1 = string1 + '-BO']	✓
			Overrides	
Rule Statements				
Ref	ID	Post	Alias	Text
1		Warning	Entity1	Incomplete shipment. Backorder was generated.

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>qtyBackordered</li> <li>qtyOrdered [8]</li> <li>qtyShipped [8]</li> <li>string1 [PO-123]</li> <li>string2 [ABLE]</li> </ul> </li> <li>Entity1 [2]                             <ul style="list-style-type: none"> <li>qtyBackordered</li> <li>qtyOrdered [14]</li> <li>qtyShipped [5]</li> <li>string1 [PO-456]</li> <li>string2 [BAKER]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>qtyBackordered</li> <li>qtyOrdered [8]</li> <li>qtyShipped [8]</li> <li>string1 [PO-123]</li> <li>string2 [ABLE]</li> </ul> </li> <li>Entity1 [2]                             <ul style="list-style-type: none"> <li><b>qtyBackordered [9]</b></li> <li>qtyOrdered [14]</li> <li>qtyShipped [5]</li> <li>string1 [PO-456]</li> <li>string2 [BAKER]</li> </ul> </li> <li>Entity1 [3]                             <ul style="list-style-type: none"> <li><b>qtyBackordered [0]</b></li> <li><b>qtyOrdered [9]</b></li> <li><b>qtyShipped [0]</b></li> <li><b>string1 [PO-456-BO]</b></li> <li><b>string2 [BAKER]</b></li> </ul> </li> </ul>

Severity	Message	Entity
Warning	Incomplete shipment. Backorder was generated.	Entity1[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.

Scope	Conditions	0	1
<ul style="list-style-type: none"> <li>Entity1 [e1]                             <ul style="list-style-type: none"> <li>qtyBackordered</li> <li>qtyOrdered</li> <li>qtyShipped</li> <li>string1</li> <li>entity2 (Entity2) [collection1]</li> </ul> </li> </ul>	a	e1.qtyOrdered > 0	T
	b		
	c		
	d		
Filters	Actions		
	Post Message(s)		
	A	e1.clone	<input checked="" type="checkbox"/>
	B		
		Overrides	

Ref	ID	Post	Alias	Text
1		Info	e1	Double all orders.

## SAMPLE RULETEST: COLLECTION

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]               <ul style="list-style-type: none"> <li>qtyOrdered [3]</li> <li>string1 [PO-20141003.23]</li> <li>string2 [Alpha Industries]</li> </ul> </li> <li>Entity1 [2]               <ul style="list-style-type: none"> <li>qtyOrdered [17]</li> <li>string1 [PO-20141003.24]</li> <li>string2 [Beta Industries]</li> </ul> </li> <li>Entity1 [3]               <ul style="list-style-type: none"> <li>qtyOrdered [0]</li> <li>string1 [PO-20141003.25]</li> <li>string2 [Omega Industries]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]               <ul style="list-style-type: none"> <li>qtyOrdered [3]</li> <li>string1 [PO-20141003.23]</li> <li>string2 [Alpha Industries]</li> </ul> </li> <li>Entity1 [2]               <ul style="list-style-type: none"> <li>qtyOrdered [17]</li> <li>string1 [PO-20141003.24]</li> <li>string2 [Beta Industries]</li> </ul> </li> <li>Entity1 [3]               <ul style="list-style-type: none"> <li>qtyOrdered [0]</li> <li>string1 [PO-20141003.25]</li> <li>string2 [Omega Industries]</li> </ul> </li> <li>Entity1 [4]               <ul style="list-style-type: none"> <li>qtyOrdered [17]</li> <li>string1 [PO-20141003.24]</li> <li>string2 [Beta Industries]</li> </ul> </li> <li>Entity1 [5]               <ul style="list-style-type: none"> <li>qtyOrdered [3]</li> <li>string1 [PO-20141003.23]</li> <li>string2 [Alpha Industries]</li> </ul> </li> </ul>

Severity	Message	Entity
Info	Double all orders.	Entity1[2]
Info	Double all orders.	Entity1[1]

## 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`.

The screenshot shows the 'Concatenate.ers' rulesheet editor. On the left is a 'Scope' tree with 'Entity1' containing 'string1' and 'entity2 (Entity2) [e2]' which contains 'string1' and 'string2'. Below the scope tree are 'Filters' (1, 2) and 'Rule Statements'.

Conditions		0
a		
b		
c		
d		
Actions		<
Post Message(s)		
A	Entity1.string1 = e2.string1.concat(e2.string2)	<input checked="" type="checkbox"/>
B		
C		
D		
Overrides		

Ref	ID	Post	Alias	Text
A0				string1 of Entity1 is equal to the concatenation of string1 and string2 of Entity2

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]           <ul style="list-style-type: none"> <li>entity2 (Entity2) [1]               <ul style="list-style-type: none"> <li>string1 [Joe]</li> <li>string2 [Montana]</li> </ul> </li> </ul> </li> <li>Entity1 [2]           <ul style="list-style-type: none"> <li>entity2 (Entity2) [2]               <ul style="list-style-type: none"> <li>string1 [Swedish]</li> <li>string2 [Meatball]</li> </ul> </li> </ul> </li> <li>Entity1 [3]           <ul style="list-style-type: none"> <li>entity2 (Entity2) [3]               <ul style="list-style-type: none"> <li>string1 [easy as]</li> <li>string2 [123]</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]           <ul style="list-style-type: none"> <li>string1 [JoeMontana]</li> <li>entity2 (Entity2) [1]               <ul style="list-style-type: none"> <li>string1 [Joe]</li> <li>string2 [Montana]</li> </ul> </li> </ul> </li> <li>Entity1 [2]           <ul style="list-style-type: none"> <li>string1 [SwedishMeatball]</li> <li>entity2 (Entity2) [2]               <ul style="list-style-type: none"> <li>string1 [Swedish]</li> <li>string2 [Meatball]</li> </ul> </li> </ul> </li> <li>Entity1 [3]           <ul style="list-style-type: none"> <li>string1 [easy as123]</li> <li>entity2 (Entity2) [3]               <ul style="list-style-type: none"> <li>string1 [easy as]</li> <li>string2 [123]</li> </ul> </li> </ul> </li> </ul>

# Character at

## SYNTAX

`<String>.characterAt(index:Integer)`

## DESCRIPTION

Returns the character at the specified position in the String.

## USAGE RESTRICTIONS

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

## RULESHEET EXAMPLE

This action-only operator parses the specified string, and then returns that character to the return character string.

Scope	Conditions	
Entity1	a	0
string1	b	
string2	c	
Filters	Actions	
1	Post Message(s)	<
2	A Entity1.string1=Entity1.string2.characterAt(4)	<input checked="" type="checkbox"/>
	B	
	C	
	Overrides	

Ref	ID	Post	Alias	Text
A0				Return the character at index 4 of Entity1.string2

## SAMPLE RULETEST

A sample Ruletest provides three elements that point out (1) the expected behavior, (2) the result when the character is not alphanumeric, and (3) a null when there is no character at that position in the String.

Input	Output
Entity1 [1] string2 [abcde]	Entity1 [1] string1 [d] string2 [abcde]
Entity1 [2] string2 [555-1212]	Entity1 [2] string1 [-] string2 [555-1212]
Entity1 [3] string2 [abc]	Entity1 [3] string2 [abc]

# 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.

The screenshot shows a rulesheet editor window titled 'Contains.ers'. It contains a table with two columns for test cases (1 and 2) and two rows for conditions and actions. Below the table is a 'Rule Statements' section with a table listing two rules: A1 and A2.

Conditions		1	2
a	Entity1.string1.contains('silver')	T	F
b			
Actions			
Post Message(s)		✉	✉
A	Entity1.boolean1	T	F
B			
Overrides			

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.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [Hi Ho Silver]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [hi ho silver]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [silvery]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>string1 [Hi Ho Silver]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>string1 [hi ho silver]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>string1 [silvery]</li> </ul> </li> </ul>

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

## 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.

day.ers		0	1	2
Conditions				
a	Entity1.dateTime1.day		< 15	>= 15
b				
Actions		<		
Post Message(s)			✉	✉
A	Entity1.string1		'Hold'	'Ship'
B				
Overrides				

Ref	ID	Post	Alias	Text
A1		Warning	Entity1	If the day of dateTime1 is earlier than the 15th, then assign string1 a value of 'Hold' and issue a Warning Message
A2		Info	Entity1	If the day of dateTime1 is on or after the 15th, then assign string1 a value of 'Ship' and issue an Info Message

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [5/14/2020 2:00:00 PM]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [08/07/2006 3:00:00 PM EST]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>dateTime1 [2019/12/25 5:00:00 AM]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [5/14/2020 2:00:00 PM]</li> <li>string1 [Hold]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [08/07/2006 3:00:00 PM EST]</li> <li>string1 [Hold]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>dateTime1 [2019/12/25 5:00:00 AM]</li> <li>string1 [Ship]</li> </ul> </li> </ul>

Severity	Message
Warning	If the day of dateTime1 is earlier than the 15th, then assign string1 a value of 'Hold'
Warning	If the day of dateTime1 is earlier than the 15th, then assign string1 a value of 'Hold'
Info	If the day of dateTime1 is on or after the 15th, then assign string1 a value of 'Ship' a

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

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

Rule Statements				
Ref	ID	Post	Alias	Text
A1				If 30 or fewer days have elapsed between <code>dateTime1</code> and <code>dateTime2</code> , then Entity1 is not overdue
A2				If more than 30 days have elapsed between <code>dateTime1</code> and <code>dateTime2</code> , then Entity1 is overdue

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>dateTime1 [11/24/1960 00:00:00]</li> <li>dateTime2 [12/15/1960 00:00:00]</li> </ul> </li> <li>Entity1 [2]                             <ul style="list-style-type: none"> <li>dateTime1 [11/24/1960 00:00:00]</li> <li>dateTime2 [12/15/2012 00:00:00]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>dateTime1 [11/24/1960 00:00:00]</li> <li>dateTime2 [12/15/1960 00:00:00]</li> <li>string1 [Not Overdue]</li> </ul> </li> <li>Entity1 [2]                             <ul style="list-style-type: none"> <li>dateTime1 [11/24/1960 00:00:00]</li> <li>dateTime2 [12/15/2012 00:00:00]</li> <li>string1 [Overdue]</li> </ul> </li> </ul>

# 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`.

dayOfWeek.ers				
Conditions		1	2	3
a	Entity1.dateOnly1.dayOfWeek	{1, 7}	{2, 3, 4, 5, 6}	
b				
Actions		<		
Post Message(s)				
A	Entity1.boolean1	T	F	
B				
Overrides				
Rule Statements				
Ref	ID	Post	Alias	Text
1				dateOnly1 falls on a weekend, boolean1 = true
2				dateOnly1 does not fall on a weekend, boolean1 = false

## SAMPLE RULETEST

Input	Output
<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>dateOnly1 [5/14/2020]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>dateOnly1 [1/1/2000]</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>dateOnly1 [2012-05-14]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>dateOnly1 [5/14/2020]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>dateOnly1 [1/1/2000]</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>dateOnly1 [2012-05-14]</li> </ul> </li> </ul>

## 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`.

dayOfYear.ers				
Conditions		0	1	2
a	Entity1.dateOnly1.dayOfYear		<= 183	> 183
b				
Actions		<		
Post Message(s)				
A	Entity1.string1		'1st Half'	'2nd Half'
B				
Overrides				
Rule Statements				
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
<ul style="list-style-type: none"> <li>▼ Entity1 [1]                             <ul style="list-style-type: none"> <li>dateOnly1 [5/14/2020]</li> </ul> </li> <li>▼ Entity1 [2]                             <ul style="list-style-type: none"> <li>dateOnly1 [1/1/2000]</li> </ul> </li> <li>▼ Entity1 [3]                             <ul style="list-style-type: none"> <li>dateOnly1 [7/4/2025]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1]                             <ul style="list-style-type: none"> <li>dateOnly1 [5/14/2020]</li> <li>string1 [1st Half]</li> </ul> </li> <li>▼ Entity1 [2]                             <ul style="list-style-type: none"> <li>dateOnly1 [1/1/2000]</li> <li>string1 [1st Half]</li> </ul> </li> <li>▼ Entity1 [3]                             <ul style="list-style-type: none"> <li>dateOnly1 [7/4/2025]</li> <li>string1 [2nd Half]</li> </ul> </li> </ul>

## 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](#)) 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.

Conditions		0	1
a	Entity1.boolean1		F
b			
Actions		<	
Post Message(s)			
A	Entity1.integer1 -= Entity1.integer2	<input type="checkbox"/>	<input checked="" type="checkbox"/>
B			
Overrides			

Rule Statements				
Ref	ID	Post	Alias	Text
A1				If boolean1 is false, then decrement integer1 by the value of integer2

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>▼ Entity1 [1]           <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>integer1 [10]</li> <li>integer2 [5]</li> </ul> </li> <li>▼ Entity1 [2]           <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>integer1 [12]</li> <li>integer2 [4]</li> </ul> </li> <li>▼ Entity1 [3]           <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>integer1 [25]</li> <li>integer2 [10]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1]           <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>integer1 [10]</li> <li>integer2 [5]</li> </ul> </li> <li>▼ Entity1 [2]           <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li><b>integer1 [8]</b></li> <li>integer2 [4]</li> </ul> </li> <li>▼ Entity1 [3]           <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>integer1 [25]</li> <li>integer2 [10]</li> </ul> </li> </ul>

## Disassociate elements

### 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.

The screenshot shows the 'DisassociateElement.ers' rule editor. It features a tree view on the left for the 'Scope' (Entity1 > entity2 (Entity2) [collection1]), a 'Filters' section, and a table for 'Conditions' and 'Actions'. Below these is a 'Rule Statements' table.

Conditions		1	2
a	collection1.boolean1	T	F
b			

Actions			
Post Message(s)			
A	collection1 -= collection1	<input checked="" type="checkbox"/>	<input type="checkbox"/>
B			

Ref	ID	Post	Alias	Text
A1				If boolean1 of any Entity2 inside collection1 is true, then disassociate that Entity2 element from collection1
A2				If boolean1 value of Entity2 is false, then take no action

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>boolean1 [true]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>boolean1 [true]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>boolean1 [false]</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>boolean1 [false]</li> </ul> </li> <li>Entity2 [1]</li> <li>Entity2 [2]</li> </ul> </li> </ul>

# 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`

Conditions		0
a		
b		
Actions		<
Post Message(s)		
A	Entity1.decimal2 = Entity1.decimal1 / Entity1.integer1	<input checked="" type="checkbox"/>
B		
Overrides		

Ref	ID	Post	Alias	Text
A0				decimal2 is equal to the value of decimal1 divided by integer1

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [1000.000000]</li> <li>integer1 [4]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>decimal1 [500.000000]</li> <li>integer1 [5]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>decimal1 [1550.000000]</li> <li>integer1 [10]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [1000.000000]</li> <li><b>decimal2 [250.000000]</b></li> <li>integer1 [4]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>decimal1 [500.000000]</li> <li><b>decimal2 [100.000000]</b></li> <li>integer1 [5]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>decimal1 [1550.000000]</li> <li><b>decimal2 [155.000000]</b></li> <li>integer1 [10]</li> </ul> </li> </ul>

## 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`.

The screenshot shows a Rulesheet editor with two tabs: 'Div.ers' and 'Rule Statements'. The 'Div.ers' tab is active and contains a table with 'Conditions' and 'Actions' sections.

Conditions		0	1
a			
b			
Actions		<	
Post Message(s)			
A	Entity1.integer1 = Entity1.integer2.div(3)	<input checked="" type="checkbox"/>	
B			
Overrides			

Ref	ID	Post	Alias	Text
A0				integer1 is equal to the whole number of times 3 divides into integer2

### SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>integer2 [1500]</li> </ul> </li> <li>Entity1 [2]                             <ul style="list-style-type: none"> <li>integer2 [700]</li> </ul> </li> <li>Entity1 [3]                             <ul style="list-style-type: none"> <li>integer2 [2543]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>integer1 [500]</li> <li>integer2 [1500]</li> </ul> </li> <li>Entity1 [2]                             <ul style="list-style-type: none"> <li>integer1 [233]</li> <li>integer2 [700]</li> </ul> </li> <li>Entity1 [3]                             <ul style="list-style-type: none"> <li>integer1 [847]</li> <li>integer2 [2543]</li> </ul> </li> </ul>

## 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.

Conditions		0	1	2
a	Entity1.string1.endsWith('ville')		T	F
b				
Actions		< [ ]		
Post Message(s)				
A	Entity1.string2		'Small'	'Big'
B				
Overrides				

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.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [Strongsville]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [New York]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [Amityville]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [Strongsville]</li> <li>string2 [Small]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [New York]</li> <li>string2 [Big]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [Amityville]</li> <li>string2 [Small]</li> </ul> </li> </ul>

## Equals when used as an assignment

### SYNTAX

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

## DESCRIPTION

Boolean	Assigns the truth value of <Expression1> to <Boolean1>.
DateTime*	Assigns the value of <DateTime2> to <DateTime1>.
Number	Assigns the value of <Number2> to <Number1>. Automatic <i>casting</i> (the process of changing a value's data type) 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 <String2> to <String1>.

## 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 a Rulesheet editor with the following components:

- Conditions Table:**

Conditions	0	1	2
a Entity1.boolean1		T	F
b			
- Actions Table:**

Actions	0	1	2
Post Message(s)			
A Entity1.decimal1 = 5.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
B Entity1.string1 = 'yes'	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
- Rule Statements Table:**

Ref	ID	Post	Alias	Text
A0				decimal1 is assigned a value of 5
B0				If boolean1 is true, assign the value of [yes] to string1
2				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:

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [true]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>boolean1 [false]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>decimal1 [5.000000]</li> <li>string1 [yes]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>boolean1 [false]</li> </ul> </li> </ul>

## Equals when used as a comparison

### SYNTAX

Boolean	<code>&lt;Expression1&gt; = &lt;Expression2&gt;</code>
DateTime*	<code>&lt;DateTime1&gt; = &lt;DateTime2&gt;</code>
Number	<code>&lt;Number1&gt; = &lt;Number2&gt;</code>
String	<code>&lt;String1&gt; = &lt;String2&gt;</code>

### DESCRIPTION

Boolean	Returns a value of true if <code>&lt;Expression1&gt;</code> is the same as <code>&lt;Expression2&gt;</code> .
DateTime*	Returns a value of true if <code>&lt;DateTime1&gt;</code> is the same as <code>&lt;DateTime2&gt;</code> , including both the Date and the Time portions
Number	Returns a value of true if <code>&lt;Number1&gt;</code> is the same as <code>&lt;Number2&gt;</code> . Different numeric data types may be compared in the same expression.
String	Returns a value of true if <code>&lt;String1&gt;</code> is the same as <code>&lt;String2&gt;</code> . Both case and length are examined to determine equality. Corticon Studio uses the ISO character precedence in comparing String values. See <a href="#">Character precedence in Unicode and Java Collator</a> on page 217.

\*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.

EqualsUsedAsAComparison.ers				
Conditions		0	1	2
a	Entity1.decimal1 = Entity1.decimal2		T	F
b				
Actions		<		
Post Message(s)				
A	Entity1.string1		'match'	'no match'
B				
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:

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [1000.000000]</li> <li>decimal2 [1001.230000]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>decimal1 [123.400000]</li> <li>decimal2 [123.400000]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [1000.000000]</li> <li>decimal2 [1001.230000]</li> <li>string1 [no match]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>decimal1 [123.400000]</li> <li>decimal2 [123.400000]</li> <li>string1 [match]</li> </ul> </li> </ul>

## 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.

EqualsIgnoringCase.ers				
Conditions		0	1	2
a	Entity1.string1.equalsIgnoreCase(Entity1.string2)		T	F
b				
Actions		<		
Post Message(s)				
A	Entity1.boolean1		T	F
B				
Overrides				
Rule Statements				
Ref	ID	Post	Alias	Text
1				boolean1 must be true if string1 and string2 are the same (ignoring case)
2				boolean1 must be false if string1 and string2 are not the same (ignoring case)

## 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.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [McDonnell-Douglas]</li> <li>string2 [McDONNell-DOUGlas]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [LOCKHEED]</li> <li>string2 [lockheed]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [boeing]</li> <li>string2 [boing]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>string1 [McDonnell-Douglas]</li> <li>string2 [McDONNell-DOUGlas]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>string1 [LOCKHEED]</li> <li>string2 [lockheed]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>string1 [boeing]</li> <li>string2 [boing]</li> </ul> </li> </ul>

## 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.

The screenshot shows a Rulesheet editor window titled "EqualsStringsOnly.ers". It contains two main sections: "Conditions" and "Actions", and a "Rule Statements" table below.

Conditions		0	1	2
a	Entity1.string1.equals(Entity1.string2)		T	F
b				

Actions		<		
Post Message(s)				
A	Entity1.boolean1		T	F
B				

Ref	ID	Post	Alias	Text
1				boolean1 must be true if string1 and string2 are the same
2				boolean1 must be false if string1 and string2 are not the same

### 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.

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

**Input:**

- Entity1 [1]
  - string1 [boeing]
  - string2 [boeing]
- Entity1 [2]
  - string1 [Lockheed]
  - string2 [LOCKHEED]
- Entity1 [3]
  - string1 [McDonnell-Douglas]
  - string2 [McDonnell-DOUGlas]

**Output:**

- Entity1 [1]
  - boolean1 [true]
  - string1 [boeing]
  - string2 [boeing]
- Entity1 [2]
  - boolean1 [false]
  - string1 [Lockheed]
  - string2 [LOCKHEED]
- Entity1 [3]
  - boolean1 [false]
  - string1 [McDonnell-Douglas]
  - string2 [McDonnell-DOUGlas]

## 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`.

The screenshot shows a rulesheet editor interface. The top part displays a rule configuration for 'Exists.ers'. It includes a 'Scope' tree with 'Entity1' containing 'decimal1' and 'entity2 (Entity2) [collection1]'. The 'Conditions' section shows 'a collection1 -> exists (string1 = 'New')'. The 'Actions' section shows 'A Entity1.decimal1 = Entity1.decimal1 \* 2'. A table below shows the rule's behavior in two scenarios: '1' (True) and '2' (False). In scenario 1, the condition is true and the action is checked. In scenario 2, the condition is false and the action is unchecked. Below the configuration is a 'Rule Statements' table with columns for Ref, ID, Post, Alias, and Text.

Scope	Conditions	1	2
a	collection1 -> exists (string1 = 'New')	T	F
Filters	Actions	<	
1	Entity1.decimal1 = Entity1.decimal1 * 2	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2	Overrides		

Ref	ID	Post	Alias	Text
A1				If there exists an element of collection1 whose string1 value equals [New], then double the value of decimal1 in Entity1
A2				If there does not exist an element of collection1 whose string1 value equals [New], then take no action

### SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [5.000000]</li> <li>▼ entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>string1 [New York]</li> </ul> </li> <li>▼ entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>string1 [New Jersey]</li> </ul> </li> <li>▼ entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>string1 [Rhode Island]</li> </ul> </li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>decimal1 [7.000000]</li> <li>▼ entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>string1 [New Hampshire]</li> </ul> </li> <li>▼ entity2 (Entity2) [5] <ul style="list-style-type: none"> <li>string1 [New]</li> </ul> </li> <li>▼ entity2 (Entity2) [6] <ul style="list-style-type: none"> <li>string1 [Connecticut]</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [5.000000]</li> <li>▼ entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>string1 [New York]</li> </ul> </li> <li>▼ entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>string1 [New Jersey]</li> </ul> </li> <li>▼ entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>string1 [Rhode Island]</li> </ul> </li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>decimal1 [14.000000]</li> <li>▼ entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>string1 [New Hampshire]</li> </ul> </li> <li>▼ entity2 (Entity2) [5] <ul style="list-style-type: none"> <li>string1 [New]</li> </ul> </li> <li>▼ entity2 (Entity2) [6] <ul style="list-style-type: none"> <li>string1 [Connecticut]</li> </ul> </li> </ul> </li> </ul>

## 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.

Conditions		0
a		
b		
Actions		<
Post Message(s)		
A	Entity1.decimal1 = Entity1.integer1 ** 2	<input checked="" type="checkbox"/>
B	Entity1.decimal2 = Entity1.integer2 ** 0.5	<input checked="" type="checkbox"/>
Overrides		

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.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>integer1 [4]</li> <li>integer2 [2]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>integer1 [5]</li> <li>integer2 [36]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>integer1 [7]</li> <li>integer2 [100]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [16.000000]</li> <li>decimal2 [1.414214]</li> <li>integer1 [4]</li> <li>integer2 [2]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>decimal1 [25.000000]</li> <li>decimal2 [6.000000]</li> <li>integer1 [5]</li> <li>integer2 [36]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>decimal1 [49.000000]</li> <li>decimal2 [10.000000]</li> <li>integer1 [7]</li> <li>integer2 [100]</li> </ul> </li> </ul>

## 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.

The screenshot shows a rulesheet editor for 'False.ers'. It features a 'Scope' panel with 'Entity1' expanded, a 'Filters' panel with a filter 'Entity1.boolean1 = false' selected, and an 'Actions' panel with an action 'Entity1.decimal1 = Entity1.decimal2 + Entity1.integer1' checked. Below the editor is a 'Rule Statements' table:

Ref	ID	Post	Alias	Text
A0				If boolean1 is false, then decimal1 equals the sum of decimal2 and integer1

### 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:

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>decimal2 [10.000000]</li> <li>integer1 [5]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>decimal2 [10.000000]</li> <li>integer1 [5]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>decimal1</li> <li>decimal2 [10.000000]</li> <li>integer1 [5]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li><b>decimal1 [15.000000]</b></li> <li>decimal2 [10.000000]</li> <li>integer1 [5]</li> </ul> </li> </ul>

# Floor

### SYNTAX

<Decimal>.floor

**DESCRIPTION**

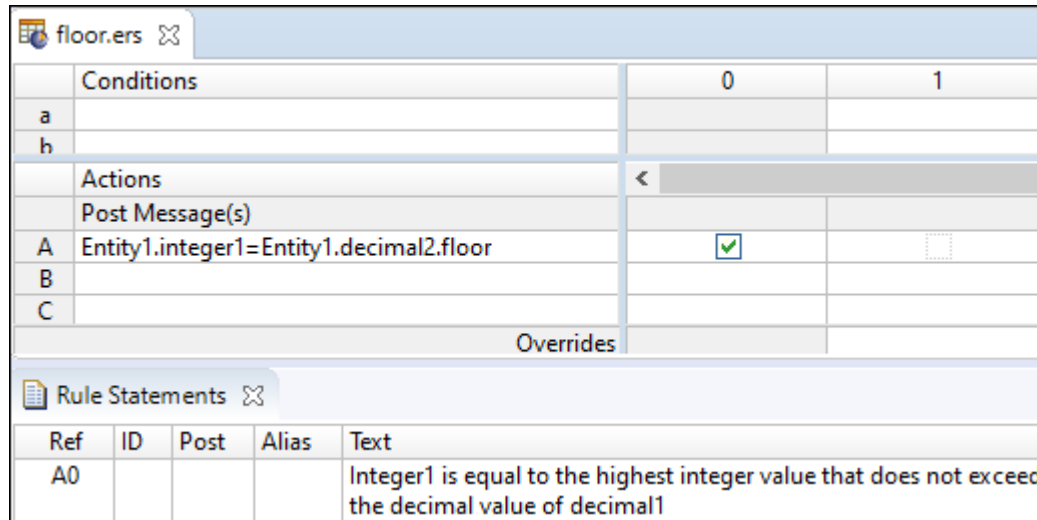
Returns the Decimal closest to zero from <Decimal>. **.floor** may also be thought of as a truncation of <Decimal>.

**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 decimal values to `decimal1` that are closer to zero than the input `decimal2` values.



The screenshot shows a rulesheet editor for 'floor.ers'. It features a table for rule configuration and a section for rule statements.

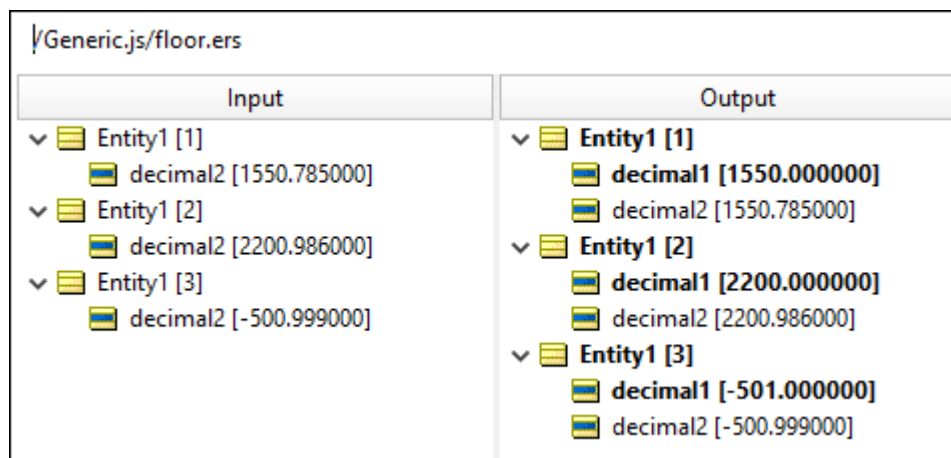
Conditions		0	1
a			
h			
Actions		<	
Post Message(s)			
A	Entity1.integer1=Entity1.decimal2.floor	<input checked="" type="checkbox"/>	<input type="checkbox"/>
B			
C			
Overrides			

Ref	ID	Post	Alias	Text
A0				Integer1 is equal to the highest integer value that does not exceed the decimal value of decimal1

**SAMPLE RULETEST**

A sample Ruletest provides three `decimal2` values. Input and Output panels are shown below:



The screenshot shows a rule test interface for 'Generic.js/floor.ers'. It displays two panels: 'Input' and 'Output'.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>decimal2 [1550.785000]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>decimal2 [2200.986000]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>decimal2 [-500.999000]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [1550.000000]</li> <li>decimal2 [1550.785000]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>decimal1 [2200.000000]</li> <li>decimal2 [2200.986000]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>decimal1 [-501.000000]</li> <li>decimal2 [-500.999000]</li> </ul> </li> </ul>

**Note:** Notice how these results differ from those shown in the Round example.

# Get Milliseconds

## SYNTAX

```
<DateTime>.getMilliseconds
```

## DESCRIPTION

Returns the number of milliseconds elapsed since the epoch: January 1, 1970.

## 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 **.getMilliSeconds** in a Nonconditional rule to evaluate the number of milliseconds between the epoch and `dateTime1`, and return the number as `integer1`.

Conditions		0		
a				
b				
Actions		<		
Post Message(s)				
A	Entity1.integer1=Entity1.dateTime1.getMilliseconds	<input checked="" type="checkbox"/>		
B				
Overrides				
Rule Statements				
Ref	ID	Post	Alias	Text
A0				Set Entity1.integer1 to the number of milliseconds between 1/1/1970 and Entity1.dateTime1

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [5/14/2021 00:00:00]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [1/2/1970 00:00:00]</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>dateTime1 [12/31/2025 11:59:59 PM]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [5/14/2021 00:00:00]</li> <li>integer1 [1620964800000]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [1/2/1970 00:00:00]</li> <li>integer1 [104400000]</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>dateTime1 [12/31/2025 11:59:59 PM]</li> <li>integer1 [1767243599000]</li> </ul> </li> </ul>

## 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 sorting syntax](#) "Advanced collection sorting syntax" in the *Rule Modeling Guide* for more examples of usage.

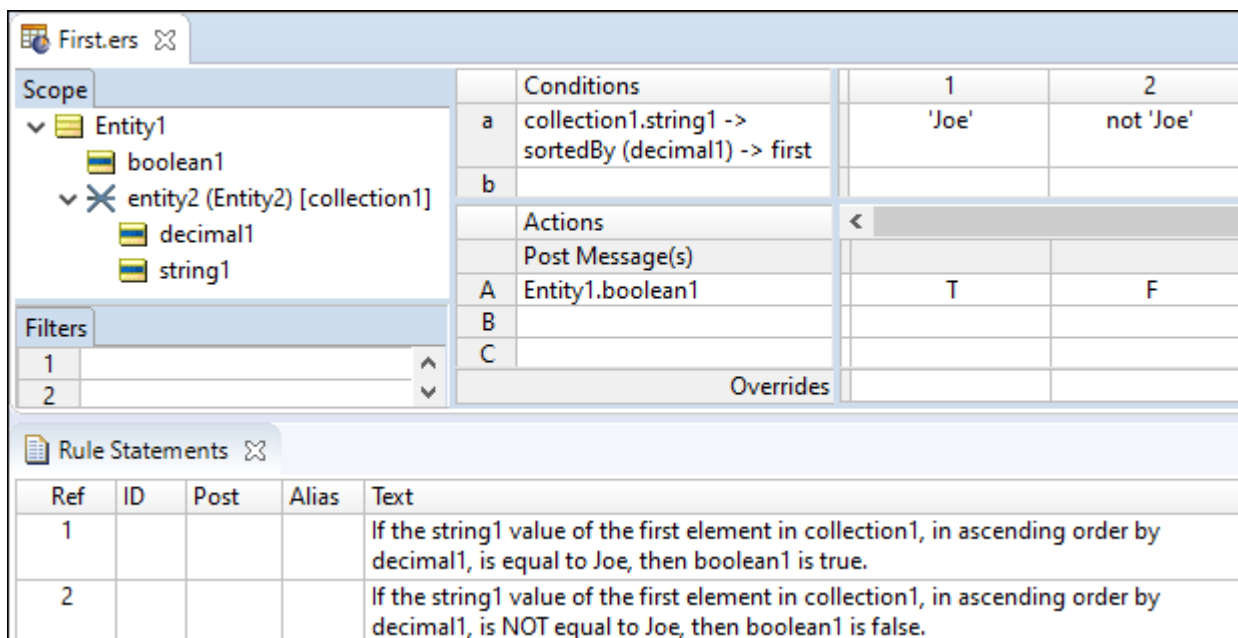
<attribute1> may be of any data type.

### 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`.



Scope		Conditions	1	2
a	collection1.string1 -> sortBy (decimal1) -> first		'Joe'	not 'Joe'
b				
Filters		Actions		
1		Post Message(s)		
2		A Entity1.boolean1	T	F
		B		
		C		
		Overrides		

Ref	ID	Post	Alias	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 first 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.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [2.500000]</li> <li>string1 [Joe]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [5.800000]</li> <li>string1 [Mary]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [3.300000]</li> <li>string1 [Sue]</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [2.500000]</li> <li>string1 [Joe]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [5.800000]</li> <li>string1 [Mary]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [3.300000]</li> <li>string1 [Sue]</li> </ul> </li> </ul> </li> </ul>

## 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 sorting syntax](#) "Advanced collection sorting syntax" in the *Rule Modeling Guide* 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`.

The screenshot shows the 'FirstNumber.ers' rulesheet editor. The 'Scope' panel on the left shows a tree structure: Entity1 (expanded) contains entity2 (Entity2) [collection2] (expanded) which contains decimal1; Entity3 (expanded) contains entity2 (Entity2) [collection3]. The 'Filters' panel shows two filters, 1 and 2. The 'Conditions' panel has three rows labeled a, b, and c. The 'Actions' panel has a 'Post Message(s)' section with two rows: A, containing the rule `collection3 = collection2 -> sortedBy(collection2.decimal1)->first(2)` with a checked checkbox, and B. The 'Rule Statements' panel at the bottom shows a table with columns Ref, ID, Post, Alias, and Text, containing one row: A0, , , , Create collection3 from the first two elements of collection2.

Ref	ID	Post	Alias	Text
A0				Create collection3 from the first two elements of collection2

## 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"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [500.000000]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [800.000000]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [700.000000]</li> </ul> </li> <li>entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>decimal1 [100.000000]</li> </ul> </li> <li>entity2 (Entity2) [5] <ul style="list-style-type: none"> <li>decimal1 [600.000000]</li> </ul> </li> </ul> </li> <li>Entity3 [1]</li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [500.000000]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [800.000000]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [700.000000]</li> </ul> </li> <li>entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>decimal1 [100.000000]</li> </ul> </li> <li>entity2 (Entity2) [5] <ul style="list-style-type: none"> <li>decimal1 [600.000000]</li> </ul> </li> </ul> </li> <li>Entity3 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1]</li> <li>entity2 (Entity2) [4]</li> </ul> </li> </ul>

**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 `->sortByDesc` to order the collection largest-to-smallest and then moves the top 2 entities to the result sequence:

FirstNumber.ers

Scope	Conditions	Value
Entity1	a	
entity2 (Entity2) [collection2]	b	
decimal1	c	0
Entity3		
entity2 (Entity2) [collection3]		
Filters		
1		
2		
Actions		
Post Message(s)		
A	collection3 = collection2 -> sortByDesc(collection2.decimal1)->first(2)	<input checked="" type="checkbox"/>
B		
Overrides		

Ref	ID	Post	Alias	Text
A0				Create collection3 from the first two elements of collection2, sorted in descending order

### 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"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>▼ entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [500.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [800.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [700.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>decimal1 [100.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [5] <ul style="list-style-type: none"> <li>decimal1 [600.000000]</li> </ul> </li> </ul> </li> <li>Entity3 [1]</li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>▼ entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [500.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [800.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [700.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>decimal1 [100.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [5] <ul style="list-style-type: none"> <li>decimal1 [600.000000]</li> </ul> </li> </ul> </li> <li>▼ Entity3 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [2]</li> <li>entity2 (Entity2) [3]</li> </ul> </li> </ul>

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

## 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`.

The screenshot shows the 'ForAll.ers' rule editor. It features a 'Scope' tree on the left with 'Entity1' containing 'decimal1' and 'entity2 (Entity2) [collection1]'. A 'Conditions' table is visible with two columns, '1' and '2'. Condition 'a' is 'collection1 -> forAll(string1 = 'New')', with 'T' in column 1 and 'F' in column 2. Below this is an 'Actions' section with 'Post Message(s)' and two rows, 'A' and 'B'. Row 'A' contains 'Entity1.decimal1 = Entity1.decimal1 \* 2' with a checked checkbox in column 1 and an unchecked checkbox in column 2. Row 'B' is empty. An 'Overrides' section is at the bottom of the actions. Below the actions is a 'Filters' table with two rows. At the bottom is a 'Rule Statements' table with two rows.

Ref	ID	Post	Alias	Text
1				If, within collection1, all string1 values equal [New], then double the value of decimal1 in Entity1
2				If, within collection1, not all string1 values equal [New], then take no action

### SAMPLE RULETEST

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

The screenshot shows two panels: 'Input' and 'Output'. The 'Input' panel shows a tree structure with 'Entity1 [1]' and 'Entity1 [2]'. 'Entity1 [1]' has 'decimal1 [5.000000]' and three 'entity2 (Entity2)' elements with 'string1' values 'New', 'New', and 'Rhode Island'. 'Entity1 [2]' has 'decimal1 [7.000000]' and four 'entity2 (Entity2)' elements with 'string1' values 'New', 'New', 'New', and 'New'. The 'Output' panel shows the result: 'Entity1 [1]' remains the same, but 'Entity1 [2]' now has 'decimal1 [14.000000]' and the same four 'entity2 (Entity2)' elements.

# Greater than

## SYNTAX

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

## DESCRIPTION

DateTime*	Returns a value of true if <DateTime1> is greater than or equal to <DateTime2>. This is equivalent to <DateTime1> occurring "after" <DateTime2>
Number	Returns a value of true if <Number1> is greater than <Number2>. Different numeric data types may be compared in the same expression.
String	Returns a value of true if <String1> is greater than <String2>. Studio uses <a href="#">Character precedence in Unicode and Java Collator</a> on page 217 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.

greaterThan.ers		0	1	2
Conditions				
a	Entity1.string1 > Entity1.string2		T	F
b				
Actions		<		
Post Message(s)				
A	Entity1.dateOnly1 = today	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
B				
Overrides				
Rule Statements				
Ref	ID	Post	Alias	Text
1				If string1 is greater than string2, then assign today's date to dateOnly1
2				If string1 is not greater than string2, then take no action

### SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [9]</li> <li>string2 [1]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [b]</li> <li>string2 [a]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [high-five]</li> <li>string2 [high five]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateOnly1 [08/05/20]</li> <li>string1 [9]</li> <li>string2 [1]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateOnly1 [08/05/20]</li> <li>string1 [b]</li> <li>string2 [a]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>dateOnly1 [08/05/20]</li> <li>string1 [high-five]</li> <li>string2 [high five]</li> </ul> </li> </ul>

## Greater than or equal to

### SYNTAX

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

## DESCRIPTION

DateTime*	Returns a value of true if <DateTime1> is greater than or equal to <DateTime2>. This is equivalent to <DateTime1> occurring on or after <DateTime2>
Number	Returns a value of true if <Number1> is greater than or equal to <Number2>. Different numeric data types may be compared in the same expression.
String	Returns a value of true if <String1> is greater than or equal to <String2>. Corticon Studio uses <a href="#">Character precedence in Unicode and Java Collator</a> on page 217 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.

Conditions		0	1	2
a	Entity1.string1 >= Entity1.string2		T	F
b				
Actions		<		
Post Message(s)				
I	Entity1.dateOnly1 = today	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
J				
Overrides				

Ref	ID	Post	Alias	Text
A1				If string1 is greater than or equal to string2, then assign today's date to dateOnly1
2				If string1 is not greater than or equal to string2, then take no action

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [Fred]</li> <li>string2 [Freddy]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [labour]</li> <li>string2 [labor]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [high-five]</li> <li>string2 [high five]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [Fred]</li> <li>string2 [Freddy]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateOnly1 [08/05/20]</li> <li>string1 [labour]</li> <li>string2 [labor]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>dateOnly1 [08/05/20]</li> <li>string1 [high-five]</li> <li>string2 [high five]</li> </ul> </li> </ul>

## 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`.

hour.ers				
Conditions		0	1	
a				
b				
c				
Actions		<		
Post Message(s)				
A	Entity1.integer1 = Entity1.timeOnly1.hour	<input checked="" type="checkbox"/>		
B				
C				
Overrides				
Rule Statements				
Ref	ID	Post	Alias	Text
A0				integer1 equals the hour value in timeOnly1 (based on a 24-hour clock)

## 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.

Input	Output
<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>timeOnly1 [2:00:00 PM PST]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>timeOnly1 [23:00:00 EST]</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>timeOnly1 [3:00:00 PM]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>integer1 [14]</li> <li>timeOnly1 [2:00:00 PM PST]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>integer1 [23]</li> <li>timeOnly1 [23:00:00 EST]</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>integer1 [15]</li> <li>timeOnly1 [3:00:00 PM]</li> </ul> </li> </ul>

## 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`.

HoursBetween.ers				
Conditions		1	2	
a	Entity1.dateTime1.hoursBetween(Entity1.dateTime2)	<= 24	> 24	
b				
Actions				
Post Message(s)				
A	Entity1.string1	'Not Overdue'	'Overdue'	
B				
Overrides				
Rule Statements				
Ref	ID	Post	Alias	Text
1				If 24 or fewer hours have elapsed between dateTime1 and dateTime2, then Entity1 is not overdue
2				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.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [3/10/2006 4:00:00 PM EST]</li> <li>dateTime2 [3/15/2006 2:30:00 AM EST]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [November 23, 2005 12:30:00 EST]</li> <li>dateTime2 [November 23, 2005 12:45:00 EST]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [3/10/2006 4:00:00 PM EST]</li> <li>dateTime2 [3/15/2006 2:30:00 AM EST]</li> <li>string1 [Overdue]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [November 23, 2005 12:30:00 EST]</li> <li>dateTime2 [November 23, 2005 12:45:00 EST]</li> <li>string1 [Not Overdue]</li> </ul> </li> </ul>

## In LIST

### SYNTAX

Date	<Date1> in {<Date2> , <Date3> , ... }
DateTime	<DateTime1> in {<DateTime2> , <DateTime3> , ... }
Decimal	<Decimal1> in {<Decimal2> , <Decimal3> , ... }
Integer	<Integer1> in {<Integer2> , <Integer3> , ... }

String	<String1> in {<String2>, <String3>, ...}
Time	<Time1> in {<Time2>, <Time3>, ...}

## 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:

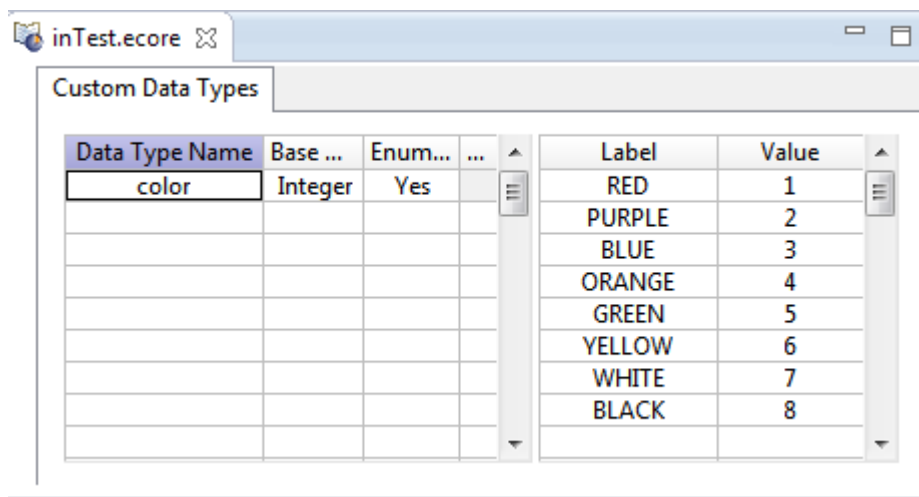
- The labels are listed without delimiters, such as {B, A, C}
- Values and labels can be mixed, such as {A, B, 'C\_value'}.

**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:

Conditions		0
a		
b		
c		
d		

Actions		<
Post Message(s)		
A	Entity1.string1 = 'primary color'	<input checked="" type="checkbox"/>
B		
C		
Overrides		

Ref	ID	Post	Alias	Text
A0				If Entity1.color1 is RED, BLUE, or YELLOW, set the value of Entity1.string1 to 'primary color'

### SAMPLE TEST

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

Input	Output
Entity1 [1] color1 [RED]	Entity1 [1] color1 [RED] string1 [primary color]
Entity1 [2] color1 [PURPLE]	Entity1 [2] color1 [PURPLE]
Entity1 [3] color1 [BLUE]	Entity1 [3] color1 [BLUE] string1 [primary color]
Entity1 [4] color1 [ORANGE]	Entity1 [4] color1 [ORANGE]
Entity1 [5] color1 [GREEN]	Entity1 [5] color1 [GREEN]
Entity1 [6] color1 [YELLOW]	Entity1 [6] color1 [YELLOW] string1 [primary color]
Entity1 [7] color1 [WHITE]	Entity1 [7] color1 [WHITE]
Entity1 [8] color1 [BLACK]	Entity1 [8] color1 [BLACK]

## In RANGE

### SYNTAX

Date	<Date1> in (<earlierDate2>..<laterDate3>)
------	---

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

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 a Rulesheet editor interface. On the left, a tree view shows the scope 'Entity1' with filters 'Entity1.dateOnly1 in ['1/1/62'..'12/31/83'] or Entity1.integer1 in (-40..32) or Entity1.string1 in ('A'..'C')'. Below this, a 'Filters' section shows the same condition. On the right, a table shows 'Conditions' (a, b, c, d, e) and 'Actions' (A: Entity1.alert = 'eligible', B, C, D, E). The 'Alerts' section shows 'Entity1.alert = 'eligible'' with a checked box. At the bottom, a 'Rule Statements' table shows a rule with ID 'A0' and text: 'If Entity1.dateOnly1 falls between 1/1/62 and 12/31/83, Entity1.integer1 has a value between -40 and 32, or Entity1.string1 is between 'A' and 'C', set Entity1.alert as 'eligible''.

## SAMPLE TEST

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

Input	Output
<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>dateOnly1 [2/4/52]</li> <li>integer1 [12]</li> <li>string1 [F]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>string1 [D]</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>dateOnly1 [3/15/77]</li> </ul> </li> <li>▼ Entity1 [4] <ul style="list-style-type: none"> <li>integer1 [32]</li> </ul> </li> <li>▼ Entity1 [5] <ul style="list-style-type: none"> <li>integer1 [-40]</li> <li>string1 [A]</li> </ul> </li> <li>▼ Entity1 [6] <ul style="list-style-type: none"> <li>dateOnly1 [1/1/62]</li> <li>string1 [C]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>alert [eligible]</li> <li>dateOnly1 [2/4/52]</li> <li>integer1 [12]</li> <li>string1 [F]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>string1 [D]</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>alert [eligible]</li> <li>dateOnly1 [3/15/77]</li> </ul> </li> <li>▼ Entity1 [4] <ul style="list-style-type: none"> <li>integer1 [32]</li> </ul> </li> <li>▼ Entity1 [5] <ul style="list-style-type: none"> <li>integer1 [-40]</li> <li>string1 [A]</li> </ul> </li> <li>▼ Entity1 [6] <ul style="list-style-type: none"> <li>alert [eligible]</li> <li>dateOnly1 [1/1/62]</li> <li>string1 [C]</li> </ul> </li> </ul>

# 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.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`.

Conditions		0	1
a	Entity1.boolean1		T
b			
Actions		<	
Post Message(s)			
A	Entity1.integer1 += Entity1.integer2	<input type="checkbox"/>	<input checked="" type="checkbox"/>
B			
Overrides			

Ref	ID	Post	Alias	Text
1				If boolean1 is true then increment integer1 by the value of integer2

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>integer1 [10]</li> <li>integer2 [5]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>integer1 [12]</li> <li>integer2 [4]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>integer1 [25]</li> <li>integer2 [10]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>integer1 [15]</li> <li>integer2 [5]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>integer1 [12]</li> <li>integer2 [4]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>integer1 [35]</li> <li>integer2 [10]</li> </ul> </li> </ul>

## 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.

The screenshot shows a rulesheet editor with the following components:

- Conditions Table:**

	Conditions	0	1
a			
b			
- Actions Table:**

	Actions	<
	Post Message(s)	
A	Entity1.integer1 = Entity1.string1.indexOf('silver')	<input checked="" type="checkbox"/>
B		
	Overrides	
- Rule Statements Table:**

Ref	ID	Post	Alias	Text
A0				integer1 is assigned the value of the starting position of silver inside string1

### 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.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [Hi Ho Silver]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [hi ho silver]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [silver and silver]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>integer1 [0]</li> <li>string1 [Hi Ho Silver]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>integer1 [7]</li> <li>string1 [hi ho silver]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>integer1 [1]</li> <li>string1 [silver and silver]</li> </ul> </li> </ul>

# Is integer

## SYNTAX

```
<String>.isInteger
```

## DESCRIPTION

Returns true if string is an integer

## 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 `isInteger`.

Conditions		0	1	2
a	Entity1.string1.isInteger		T	F
b				
c				
Actions		<		
Post Message(s)				
A	Entity1.string2		'NUMBER'	'NOT NUMBER'
B				
C				
Overrides				

Ref	ID	Post	Alias	Text
1				If Entity1.string1 is an integer, set Entity1.string2 to 'NUMBER'
2				If Entity1.string2 is not an integer, set Entity1.string2 to 'NOT NUMBER'

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [1234]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [-1234]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [1234-]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [1234]</li> <li>string2 [NUMBER]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [-1234]</li> <li>string2 [NUMBER]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [1234-]</li> <li>string2 [NOT NUMBER]</li> </ul> </li> </ul>

# Is empty

## 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 IsEmpty.rulesheet editor interface. The left pane displays the rule configuration for 'Entity1' with a collection 'entity2 (Entity2) [collection1]'. The right pane shows the rule configuration table with conditions and actions.

Scope	Conditions	1	2
a	collection1 -> isEmpty	T	F
b			

Actions	1	2
Post Message(s)	☐	☐
A		
B		
Overrides		

Ref	ID	Post	Alias	Text
1		Warning	Entity1	collection1 is empty, which means that Entity1 has no associated Entity2 elements
2		Info	Entity1	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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]</li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1]</li> <li>entity2 (Entity2) [2]</li> <li>entity2 (Entity2) [3]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]</li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1]</li> <li>entity2 (Entity2) [2]</li> <li>entity2 (Entity2) [3]</li> </ul> </li> </ul>

Rule Messages	
Severity	Message
Warning	collection1 is empty, which means that Entity1 has no associated
Info	collection1 is not empty, which means that Entity1 has at least or

## 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.

Scope		Conditions	1	2
Entity1	entity2 (Entity2) [collection1]	a collection1 -> size > 2	T	F
		b		
Filters		Actions		
1		Post Message(s)		
2		A collection1 -> iterate(string1 = 'test')	<input checked="" type="checkbox"/>	<input type="checkbox"/>
		B		
		Overrides		

Ref	ID	Post	Alias	Text
A1				If there are more than 2 elements in collection1, then assign the value "test" of string1 to every element in the collection

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1]</li> <li>entity2 (Entity2) [2]</li> <li>entity2 (Entity2) [3]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>string1 [test]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>string1 [test]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>string1 [test]</li> </ul> </li> </ul> </li> </ul>

## 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 sorting syntax](#) "Advanced collection sorting syntax" in the *Rule Modeling Guide* 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 'Last.ers' rule configuration in the Corticon Rule Editor. It is divided into several sections:

- Scope:** A tree view showing the rule's structure: Entity1 (boolean1, entity2 (Entity2) [collection1], decimal1, string1).
- Conditions:** A table with two columns (1, 2) and two rows (a, b). Row 'a' contains the condition: `collection1.string1->sortedBy (collection1.decimal1) -> last`. Row 'b' is empty.
- Actions:** A table with two columns (1, 2) and three rows (A, B, C). Row 'A' contains the action: `Entity1.boolean1`. Row 'B' is empty. Row 'C' is empty.
- Filters:** A table with two columns (1, 2) and two rows (1, 2). Both rows are empty.
- Overrides:** A table with two columns (1, 2) and one row (Overrides). The row is empty.
- Rule Statements:** A table with columns: Ref, ID, Post, Alias, Text.
 

Ref	ID	Post	Alias	Text
1		Info	Entity1	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	Entity1	If the string1 value of the last 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' interface for the 'Last.ers' rule. It is divided into two main panels: 'Input' and 'Output'.

- Input:** A tree view showing the input data: Entity1 [1] (entity2 (Entity2) [1] (decimal1 [2.500000], string1 [Mary]), entity2 (Entity2) [2] (decimal1 [5.800000], string1 [Joe]), entity2 (Entity2) [3] (decimal1 [3.300000], string1 [Sue])).
- Output:** A tree view showing the output data: Entity1 [1] (boolean1 [true], entity2 (Entity2) [1] (decimal1 [2.500000], string1 [Mary]), entity2 (Entity2) [2] (decimal1 [5.800000], string1 [Joe]), entity2 (Entity2) [3] (decimal1 [3.300000], string1 [Sue])).
- Rule Messages:** A table with columns: Severity, Message, Entity.
 

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.	Entity1[1]

## 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 sorting syntax](#) "Advanced collection sorting syntax" in the *Rule Modeling Guide* 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`.

The screenshot shows the 'LastNumber.ers' rulesheet editor. The 'Scope' panel on the left shows a tree view with 'Entity1' containing 'entity2 (Entity2) [collection2]' (with a crossed-out icon) and 'decimal1', and 'Entity3' containing 'entity2 (Entity2) [collection3]' (with a crossed-out icon). The 'Filters' panel shows two filters. The 'Conditions' table has three rows labeled 'a', 'b', and 'c'. The 'Actions' table has two rows: 'A' with the text 'collection3 = collection2 -> sortedBy(collection2.decimal1) -> last(2)' and a checked checkbox, and 'B' which is empty. Below the 'Actions' table is an 'Overrides' section. At the bottom, the 'Rule Statements' table is visible.

Ref	ID	Post	Alias	Text
A0				Create collection3 from the last two sorted elements of collection2

## 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"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [500.000000]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [800.000000]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [700.000000]</li> </ul> </li> <li>entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>decimal1 [100.000000]</li> </ul> </li> <li>entity2 (Entity2) [5] <ul style="list-style-type: none"> <li>decimal1 [600.000000]</li> </ul> </li> </ul> </li> <li>Entity3 [1]</li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [500.000000]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [800.000000]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [700.000000]</li> </ul> </li> <li>entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>decimal1 [100.000000]</li> </ul> </li> <li>entity2 (Entity2) [5] <ul style="list-style-type: none"> <li>decimal1 [600.000000]</li> </ul> </li> </ul> </li> <li>Entity3 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [2]</li> <li>entity2 (Entity2) [3]</li> </ul> </li> </ul>

**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	Count
Entity1	a	0
entity2 (Entity2) [collection2]	b	
decimal1		

Filters	Actions	Overrides
1	Post Message(s)	
2	A collection2 = collection2 -> sortBy (collection2.decimal1) -> last(2)	<input checked="" type="checkbox"/>
	Overrides	

Ref	ID	Post	Alias	Text
A0				Set collection2 as its last two items in sorted order

### 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"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>▼ entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [500.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [800.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [700.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>decimal1 [100.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [5] <ul style="list-style-type: none"> <li>decimal1 [600.000000]</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>▼ entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [800.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [700.000000]</li> </ul> </li> <li>▼ Entity2 [4] <ul style="list-style-type: none"> <li>decimal1 [100.000000]</li> </ul> </li> <li>▼ Entity2 [5] <ul style="list-style-type: none"> <li>decimal1 [600.000000]</li> </ul> </li> <li>▼ Entity2 [1] <ul style="list-style-type: none"> <li>decimal1 [500.000000]</li> </ul> </li> </ul> </li> </ul>

**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 <a href="#">Character precedence in Unicode and Java Collator</a> on page 217.

\*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.

The screenshot shows a rulesheet editor for a file named 'lessThan.ers'. It is divided into three main sections: Conditions, Actions, and Rule Statements.

Conditions		0	1	2
a	Entity1.string1 < Entity1.string2		T	F
b				

Actions		<
Post Message(s)		
A	Entity1.dateOnly1 = today	<input checked="" type="checkbox"/>
B		

Ref	ID	Post	Alias	Text
1				If string1 is less than string2, then assign today's date to dateOnly1
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:

The screenshot shows a ruletest interface with two panels: 'Input' and 'Output'. Both panels display a tree view of data for three entities.

Input	Output
Entity1 [1] string1 [apple] string2 [Apple]	Entity1 [1] dateOnly1 [08/05/20] string1 [apple] string2 [Apple]
Entity1 [3] string1 [apple] string2 [apples]	Entity1 [3] dateOnly1 [08/05/20] string1 [apple] string2 [apples]
Entity1 [4] string1 [apple] string2 [apple]	Entity1 [4] string1 [apple] string2 [apple]

# Less than or equal to

## SYNTAX

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

## DESCRIPTION

DateTime*	Returns a value of true if <DateTime1> is less than or equal to <DateTime2>. This is equivalent to <DateTime1> occurring "on or before" <DateTime2>
Number	Returns a value of true if <Number1> is less than or equal to <Number2>. Different numeric data types may be compared in the same expression.
String	Returns a value of true if <String1> is less than or equal to <String2>. Corticon Studio uses <a href="#">Character precedence in Unicode and Java Collator</a> on page 217.

\*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.

lessThanEqual.ers		0	1	2
Conditions				
a	Entity1.string1 <= Entity1.string2		T	F
b				
Actions		<		
Post Message(s)				
A	Entity1.dateOnly1 = today	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
B				
Overrides				

Ref	ID	Post	Alias	Text
1				If string1 is less than or equal to string2, then assign today's date to dateOnly1
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:

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [Apple]</li> <li>string2 [apple]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [apple]</li> <li>string2 [apples]</li> </ul> </li> <li>Entity1 [4] <ul style="list-style-type: none"> <li>string1 [apple]</li> <li>string2 [apple]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [Apple]</li> <li>string2 [apple]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>dateOnly1 [08/05/20]</li> <li>string1 [apple]</li> <li>string2 [apples]</li> </ul> </li> <li>Entity1 [4] <ul style="list-style-type: none"> <li>dateOnly1 [08/05/20]</li> <li>string1 [apple]</li> <li>string2 [apple]</li> </ul> </li> </ul>

# 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`.

log.ers		0		
Conditions				
a				
b				
Actions		<		
Post Message(s)				
A	Entity1.decimal1 = Entity1.integer1.log	<input checked="" type="checkbox"/>		
B				
Overrides				
Rule Statements				
Ref	ID	Post	Alias	Text
A0				decimal1 is equal to the logarithm (base10) of integer1

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]               <ul style="list-style-type: none"> <li>integer1 [10]</li> </ul> </li> <li>Entity1 [2]               <ul style="list-style-type: none"> <li>integer1 [1]</li> </ul> </li> <li>Entity1 [3]               <ul style="list-style-type: none"> <li>integer1 [24]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]               <ul style="list-style-type: none"> <li>decimal1 [1.000000]</li> <li>integer1 [10]</li> </ul> </li> <li>Entity1 [2]               <ul style="list-style-type: none"> <li>decimal1 [0.000000]</li> <li>integer1 [1]</li> </ul> </li> <li>Entity1 [3]               <ul style="list-style-type: none"> <li>decimal1 [1.380211]</li> <li>integer1 [24]</li> </ul> </li> </ul>

**Note:** In a case where the rule encounters  $\log(0)$ , it throws an exception that halts execution. That's because the value of  $\log(0)$  is undefined. If the rule is executing against multiple entities, the arbitrary order of execution might be different on subsequent runs before execution is halted.

# 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`.

The screenshot shows a Rulesheet editor window titled "LogarithmBaseX.ers". It contains a table with columns for "Conditions", "0", and "1". Below the table is an "Actions" section with a "Post Message(s)" field containing the expression `Entity1.decimal1 = Entity1.integer1.log(7.0)`. Below the actions is an "Overrides" section. At the bottom, there is a "Rule Statements" section with a table containing one row: "A0" with the text "decimal1 is equal to the logarithm (base 7) of integer1".

Conditions		0	1
a			
b			

Actions		<
Post Message(s)		
A	<code>Entity1.decimal1 = Entity1.integer1.log(7.0)</code>	<input checked="" type="checkbox"/>
B		

Rule Statements				
Ref	ID	Post	Alias	Text
A0				decimal1 is equal to the logarithm (base 7) of integer1

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>integer1 [10]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>integer1 [173]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>integer1 [24]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [1.183295]</li> <li>integer1 [10]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>decimal1 [2.648268]</li> <li>integer1 [173]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>decimal1 [1.633197]</li> <li>integer1 [24]</li> </ul> </li> </ul>

**Note:** In a case where the rule encounters `log(0)`, it throws an exception that halts execution. That's because the value of `log(0)` is undefined. If the rule is executing against multiple entities, the arbitrary order of execution might be different on subsequent runs before execution is halted.

## 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		1	2
a	Entity1.string1.toLowerCase = Entity1.string2	T	F
b			
Actions		<	
Post Message(s)			
A	Entity1.boolean1	T	F
B			
Overrides			

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:

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [Boeing]</li> <li>string2 [boeing]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [Boeing]</li> <li>string2 [Boeing]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [boeing]</li> <li>string2 [BOEING]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>string1 [Boeing]</li> <li>string2 [boeing]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>string1 [Boeing]</li> <li>string2 [Boeing]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>string1 [boeing]</li> <li>string2 [BOEING]</li> </ul> </li> </ul>

# Matches

## SYNTAX

```
<String>.matches(regularExpression:String)
```

## DESCRIPTION

Returns true if the regular expression matches the String.

## USAGE RESTRICTIONS

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

## RULESHEET EXAMPLES

This sample Rulesheet uses **matches** in non-conditional actions:

The screenshot shows a Rulesheet editor window titled 'matches.ers'. It contains two main sections: 'Conditions' and 'Actions'.

**Conditions:**

Label	Text	Value
	Conditions	0
a		
b		

**Actions:**

Label	Text	Status
	Post Message(s)	<
A	Entity1.boolean1 = Entity1.string1.matches ( '[A-Z,a-z]{5}[0-9]{4}[A-Z,a-z]{1}' )	<input checked="" type="checkbox"/>
B	Entity1.boolean2 = Entity1.string2.matches ( '[a-zA-Z0-9_+-.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+\$' )	<input checked="" type="checkbox"/>
Overrides		

**Rule Statements:**

Ref	ID	Post	Alias	Text
A0				Entity1.boolean1 is true if string1 is a valid identifier, and false otherwise
B0				Entity1.boolean2 is true if string2 is a valid email address, and false otherwise

**Action A: Determine whether a String is a valid identifier** - A String must contain an item identification with the following pattern:

1. Characters 1-5: alphabetic.
2. Characters 6-10: numeric.
3. Character 11: alphabetic.

**Action B: Check whether an email address is valid** - An email address must have alphanumeric characters and certain special characters before and after an @ and a dot.

## SAMPLE RULETEST

A sample Ruletest provides various valid and invalid Strings that are evaluated by the two regular expression examples.

Input	Output
<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>string1 [ABCDE1234x]</li> <li>string2 [ProgressSupport@progress.com]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>string1 [ABCDEFGHlx]</li> <li>string2 [???@progress.com]</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>string1 [ABCDE-1234-x]</li> <li>string2 [ProgressSupport @ progress.com]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>boolean2 [true]</li> <li>string1 [ABCDE1234x]</li> <li>string2 [ProgressSupport@progress.com]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>boolean2 [false]</li> <li>string1 [ABCDEFGHlx]</li> <li>string2 [???@progress.com]</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>boolean2 [false]</li> <li>string1 [ABCDE-1234-x]</li> <li>string2 [ProgressSupport @ progress.com]</li> </ul> </li> </ul>

## 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.

MaximumValue.ers				
Conditions		0	1	2
a	Entity1.decimal1.max(Entity1.decimal2) > 5.0		T	-
b	Entity1.integer1.max(Entity1.integer1) > 8		-	T
Actions		<		
Post Message(s)			☑	☑
A				
B				
Overrides				
Rule Statements				
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.

Input	Output	
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [4.900000]</li> <li>decimal2 [5.100000]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>decimal1 [5.000000]</li> <li>decimal2 [4.300000]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>integer1 [5]</li> <li>integer2 [14]</li> </ul> </li> <li>Entity1 [4] <ul style="list-style-type: none"> <li>integer1 [7]</li> <li>integer2 [1]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [4.900000]</li> <li>decimal2 [5.100000]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>decimal1 [5.000000]</li> <li>decimal2 [4.300000]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>integer1 [5]</li> <li>integer2 [14]</li> </ul> </li> <li>Entity1 [4] <ul style="list-style-type: none"> <li>integer1 [7]</li> <li>integer2 [1]</li> </ul> </li> </ul>	
Rule 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`.

The screenshot shows the 'MaximumValueCollection.ers' rulesheet editor. It features a 'Scope' tree on the left with 'Entity1' containing 'decimal1' and 'entity2 (Entity2) [collection1]'. Below the scope is a 'Filters' section with two filter slots. The main area is divided into 'Conditions' (with slots 'a' and 'b'), 'Actions' (with a 'Post Message(s)' section containing rule 'A: Entity1.decimal1 = collection1.decimal1->max' and rule 'R'), and an 'Overrides' section. At the bottom, the 'Rule Statements' table is visible:

Ref	ID	Post	Alias	Text
A0				Assign the highest value of decimal1 in collection1 to Entity1.decimal1

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [1.100000]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [3.100000]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [2.700000]</li> </ul> </li> <li>entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>decimal1 [7.900000]</li> </ul> </li> <li>entity2 (Entity2) [5] <ul style="list-style-type: none"> <li>decimal1 [4.600000]</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li><b>decimal1 [7.900000]</b></li> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [1.100000]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [3.100000]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [2.700000]</li> </ul> </li> <li>entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>decimal1 [7.900000]</li> </ul> </li> <li>entity2 (Entity2) [5] <ul style="list-style-type: none"> <li>decimal1 [4.600000]</li> </ul> </li> </ul> </li> </ul>

# 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.

MinimumValue.ers				
Conditions		0	1	2
a	Entity1.decimal1.min(Entity1.decimal2) > 5.0		T	-
b	Entity1.integer1.min(Entity1.integer2) > 8		-	T
c				
Actions		<		
Post Message(s)			✉	✉
A				
B				
Overrides				
Rule Statements				
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.

Input		Output	
Entity1 [1]	decimal1 [4.900000] decimal2 [5.100000]	Entity1 [1]	decimal1 [4.900000] decimal2 [5.100000]
Entity1 [2]	decimal1 [5.100000] decimal2 [594.300000]	Entity1 [2]	decimal1 [5.100000] decimal2 [594.300000]
Entity1 [3]	integer1 [1500] integer2 [245]	Entity1 [3]	integer1 [1500] integer2 [245]
Entity1 [4]	integer1 [350] integer2 [1]	Entity1 [4]	integer1 [350] integer2 [1]

Rule Statements		Rule Messages
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

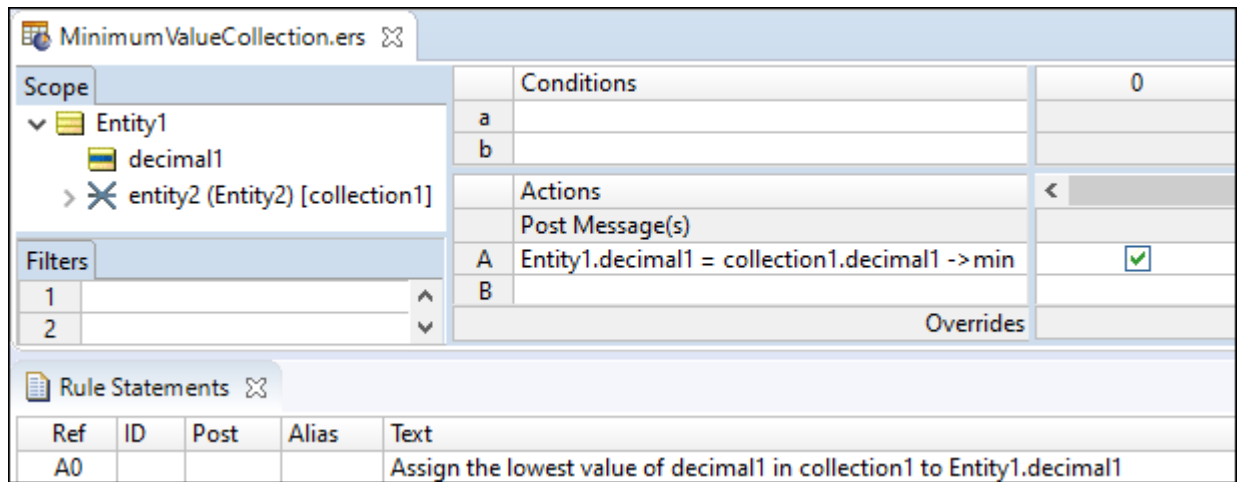
Returns the lowest value of <attribute> for all elements in <Collection>. <attribute> must be a numeric data type. <Collection> must be expressed as a unique alias.

## 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.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [1.100000]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [3.100000]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [2.700000]</li> </ul> </li> <li>entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>decimal1 [7.900000]</li> </ul> </li> <li>entity2 (Entity2) [5] <ul style="list-style-type: none"> <li>decimal1 [4.600000]</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li><b>decimal1 [1.100000]</b></li> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [1.100000]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [3.100000]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [2.700000]</li> </ul> </li> <li>entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>decimal1 [7.900000]</li> </ul> </li> <li>entity2 (Entity2) [5] <ul style="list-style-type: none"> <li>decimal1 [4.600000]</li> </ul> </li> </ul> </li> </ul>

## 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`.

Conditions		0
a		
b		
Actions		<
Post Message(s)		
A	Entity1.integer1 = Entity1.timeOnly1.min	<input checked="" type="checkbox"/>
B		
Overrides		

Ref	ID	Post	Alias	Text
A0				integer1 equals the minute value in timeOnly1

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>timeOnly1 [00:00:00]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>timeOnly1 [11:12:35 PM]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>timeOnly1 [23:24:00]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>integer1 [0]</li> <li>timeOnly1 [00:00:00]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>integer1 [12]</li> <li>timeOnly1 [11:12:35 PM]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>integer1 [24]</li> <li>timeOnly1 [23:24:00]</li> </ul> </li> </ul>

# 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		1	2
a	Entity1.dateTime1.minsBetween(Entity1.dateTime2)	<= 30	> 30
b			
Actions		<	
Post Message(s)			
A	Entity1.string1	'Not Overdue'	'Overdue'
B			
Overrides			

Ref	ID	Post	Alias	Text
1				If 30 or fewer minutes have elapsed between <code>dateTime1</code> and <code>dateTime2</code> , then <code>Entity1</code> is not overdue
2				If more than 30 minutes have elapsed between <code>dateTime1</code> and <code>dateTime2</code> , then <code>Entity2</code> 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
<ul style="list-style-type: none"> <li>Entity1 [1]           <ul style="list-style-type: none"> <li>dateTime1 [3/14/2026 4:00:00 PM EST]</li> <li>dateTime2 [3/15/2026 2:30:00 AM EST]</li> </ul> </li> <li>Entity1 [2]           <ul style="list-style-type: none"> <li>dateTime1 [November 23, 2021 12:30:00 EST]</li> <li>dateTime2 [November 23, 2021 12:10:00 EST]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]           <ul style="list-style-type: none"> <li>dateTime1 [3/14/2026 4:00:00 PM EST]</li> <li>dateTime2 [3/15/2026 2:30:00 AM EST]</li> <li>string1 [Overdue]</li> </ul> </li> <li>Entity1 [2]           <ul style="list-style-type: none"> <li>dateTime1 [November 23, 2021 12:30:00 EST]</li> <li>dateTime2 [November 23, 2021 12:10:00 EST]</li> <li>string1 [Not Overdue]</li> </ul> </li> </ul>

# 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`.

The screenshot shows the Mod.ers rule editor interface. It has two tabs: 'Mod.ers' and 'Rule Statements'. The 'Mod.ers' tab is active and contains a table with 'Conditions' and 'Actions' sections. The 'Conditions' section has columns for '0' and '1'. The 'Actions' section has a 'Post Message(s)' column and a 'Check' column. The 'Rule Statements' tab is also visible and contains a table with columns for 'Ref', 'ID', 'Post', 'Alias', and 'Text'.

Conditions		0	1
a			
b			

Actions		<	Check
Post Message(s)			
A	Entity1.integer1 = Entity1.integer2.mod(3)		<input checked="" type="checkbox"/>
B			

Ref	ID	Post	Alias	Text
A0				Integer1 equals the whole number remainder of integer2, divided by 3

## SAMPLE RULETEST

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

The screenshot shows the Ruletest interface with two panels: 'Input' and 'Output'. Both panels show a tree view of entities and their values.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>integer2 [675]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>integer2 [781]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>integer2 [1022]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>integer1 [0]</li> <li>integer2 [675]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>integer1 [1]</li> <li>integer2 [781]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>integer1 [2]</li> <li>integer2 [1022]</li> </ul> </li> </ul>

# 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.

The screenshot shows a rulesheet editor for 'month.ers'. It is divided into two main sections: 'Conditions' and 'Rule Statements'.

**Conditions Section:**

Conditions	Count
a	0
b	

**Actions Section:**

Actions	Count
Post Message(s)	<
A Entity1.integer1=Entity1.dateTime1.month	✓
B Entity1.integer2=Entity1.dateOnly1.month	✓

**Rule Statements Section:**

Ref	ID	Post	Alias	Text
A0				integer1 equals the month value in dateTime1
B0				integer2 equals the month value in dateOnly1

## 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.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [5/5/1955 00:00:00]</li> <li>dateTime2 [9/5/2015 00:00:00]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [11/5/1955 00:00:00]</li> <li>dateTime2 [9/5/2015 00:00:00]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>dateTime1 [12/5/2025 00:00:00]</li> <li>dateTime2 [3/5/2015 00:00:00]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [5/5/1955 00:00:00]</li> <li>dateTime2 [9/5/2015 00:00:00]</li> <li>integer1 [5]</li> <li>integer2 [9]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [11/5/1955 00:00:00]</li> <li>dateTime2 [9/5/2015 00:00:00]</li> <li>integer1 [11]</li> <li>integer2 [9]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>dateTime1 [12/5/2025 00:00:00]</li> <li>dateTime2 [3/5/2015 00:00:00]</li> <li>integer1 [12]</li> <li>integer2 [3]</li> </ul> </li> </ul>

## 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`.

Conditions		1	2
a	Entity1.dateTime1.monthsBetween(Entity1.dateTime2)	<= 6	> 6
b			
Actions		<	
Post Message(s)			
A	Entity1.string1	'Not Overdue'	'Overdue'
B			
Overrides			

Ref	ID	Post	Alias	Text
1				If 6 or fewer months have elapsed between date1 and date2, then Entity1 is not overdue
2				If more than 6 months have elapsed between date1 and date2, then Entity1 is overdue

### 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).

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [12/4/2021 12:00:00 PM EST]</li> <li>dateTime2 [March 11, 2022 17:00:00 EST]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [7/4/2025 12:00:00 PM EST]</li> <li>dateTime2 [March 11, 2026 17:00:00 EST]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [12/4/2021 12:00:00 PM EST]</li> <li>dateTime2 [March 11, 2022 17:00:00 EST]</li> <li>string1 [Not Overdue]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [7/4/2025 12:00:00 PM EST]</li> <li>dateTime2 [March 11, 2026 17:00:00 EST]</li> <li>string1 [Overdue]</li> </ul> </li> </ul>

## 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

Conditions		1	2
a	Entity1.integer1 * Entity1.integer2	< 100	>= 100
b			
Actions		<	
Post Message(s)			
A	Entity1.boolean1	T	F
B			
Overrides			

Ref	ID	Post	Alias	Text
1				If integer1 multiplied by integer2 is less than 100, then boolean1 is true
2				If integer1 multiplied by integer2 is greater than or equal to 100, then boolean1 is false

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>integer1 [9]</li> <li>integer2 [10]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>integer1 [500]</li> <li>integer2 [2]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>integer1 [25]</li> <li>integer2 [5]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>integer1 [9]</li> <li>integer2 [10]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>integer1 [500]</li> <li>integer2 [2]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>integer1 [25]</li> <li>integer2 [5]</li> </ul> </li> </ul>

# 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`.

The screenshot shows a rulesheet editor with two tabs: 'NaturalLog.ers' and 'Rule Statements'. The 'NaturalLog.ers' tab is active and displays a table with 'Conditions' and 'Actions' sections. The 'Conditions' section has a value of '0'. The 'Actions' section has a 'Post Message(s)' field with the text 'Entity1.decimal1 = Entity1.decimal2.ln' and a checked checkbox. Below the 'Actions' section is an 'Overrides' section. The 'Rule Statements' tab is also visible, showing a table with columns 'Ref', 'ID', 'Post', 'Alias', and 'Text'. The table contains one row with 'Ref' 'A0' and 'Text' 'decimal1 is equal to the natural logarithm (base e) of decimal2'.

Conditions		0
a		
b		
Actions		<
Post Message(s)		
A	Entity1.decimal1 = Entity1.decimal2.ln	<input checked="" type="checkbox"/>
B		
Overrides		

Ref	ID	Post	Alias	Text
A0				decimal1 is equal to the natural logarithm (base e) of decimal2

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>decimal2 [2.719000]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>decimal2 [125.733000]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>decimal2 [24.300000]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [1.000264]</li> <li>decimal2 [2.719000]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>decimal1 [4.834161]</li> <li>decimal2 [125.733000]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>decimal1 [3.190476]</li> <li>decimal2 [24.300000]</li> </ul> </li> </ul>

**Note:** In a case where the rule encounters `0.ln`, it throws an exception that halts execution. That's because the value of `0.ln` is undefined. If the rule is executing against multiple entities, the arbitrary order of execution might be different on subsequent runs before execution is halted.

# New

## SYNTAX

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

## DESCRIPTION

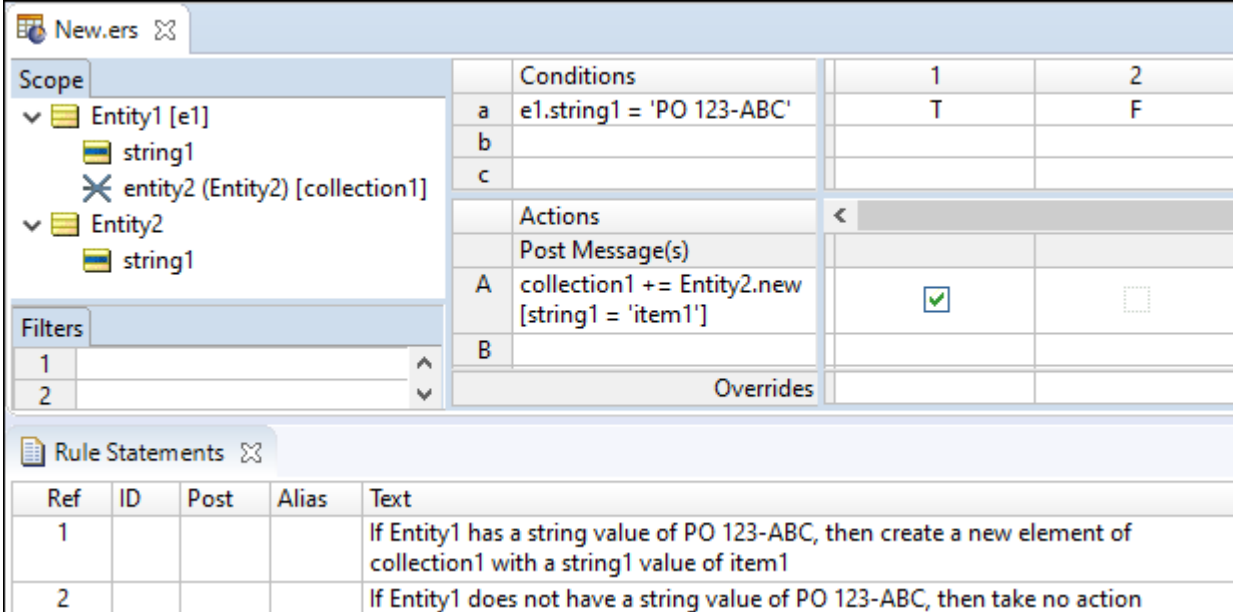
creates a new <Entity> with attribute values defined by optional <Expression>. Expressions (when present) should be written as assignments in the form: *attribute = value*. The attribute used in <Expression> (when present) must be an attribute of <Entity>.

## 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.



The screenshot shows a Rulesheet editor window titled "New.ers". The interface is divided into several sections:

- Scope:** A tree view showing the hierarchy of entities and collections. Under "Entity1 [e1]", there is a "string1" attribute and an "entity2 (Entity2) [collection1]" collection. Under "Entity2", there is a "string1" attribute.
- Filters:** A table with two rows, numbered 1 and 2.
- Conditions:** A table with three rows (a, b, c) and two columns (1, 2). Row 'a' contains the condition "e1.string1 = 'PO 123-ABC'".
- Actions:** A table with three rows (A, B) and two columns (1, 2). Row 'A' contains the action "collection1 += Entity2.new [string1 = 'item1']".
- Overrides:** A table with two columns (1, 2).
- Rule Statements:** A table with columns "Ref", "ID", "Post", "Alias", and "Text". It contains two statements:
 

Ref	ID	Post	Alias	Text
1				If Entity1 has a string value of PO 123-ABC, then create a new element of collection1 with a string1 value of item1
2				If Entity1 does not have a string value of PO 123-ABC, then take no action

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>string1 [PO 123-ABC]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>string1 [PO 987-XYZ]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>string1 [PO 123-ABC]</li> <li>▼ entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>string1 [item1]</li> </ul> </li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>string1 [PO 987-XYZ]</li> </ul> </li> </ul>

## 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:

Scope	Conditions
<ul style="list-style-type: none"> <li>ManyToOne</li> <li>OneToOne</li> <li>Root               <ul style="list-style-type: none"> <li>✕ manyToMany (ManyToMany)</li> <li>➤ manyToOne (ManyToOne)</li> <li>← oneToMany (OneToMany)</li> <li>— oneToOne (OneToOne)</li> </ul> </li> </ul>	a b c d e f g h i j k
Filters	Actions
1	Post Message(s)
2	A Root.newUnique[oneToOne = OneToOne]
3	B Root.newUnique[manyToOne = ManyToOne]
4	C Root.newUnique[oneToMany = OneToMany]
	D Root.newUnique[manyToMany = ManyToMany]

## 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.

The screenshot shows the Corticon Rule Editor for a rule named 'NewUnique.ers'. The interface is divided into several sections:

- Scope:** A tree view showing the rule's structure. It includes 'Entity1 [e1]' with a 'string1' property and a collection 'entity2 (Entity2) [collection1]'. 'Entity2' also has a 'string1' property.
- Conditions:** A table with columns for condition labels (a, b, c) and two columns for rule versions (1 and 2).
 

	1	2
a	e1.string1 = 'PO 123-ABC'	T
b		
c		
- Actions:** A table with columns for action labels (A, B) and two columns for rule versions (1 and 2).
 

	1	2
A	collection1 += Entity2.newUnique [string1 = 'item1']	<input checked="" type="checkbox"/>
B		<input type="checkbox"/>
- Filters:** A simple list with two entries, '1' and '2'.
- Rule Statements:** A table summarizing the rule logic.
 

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:

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>string1 [PO 123-ABC]</li> </ul> </li> <li>Entity2 [1]                             <ul style="list-style-type: none"> <li>string1 [item1]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>string1 [PO 123-ABC]</li> </ul> </li> <li>Entity2 [1]                             <ul style="list-style-type: none"> <li>string1 [item1]</li> </ul> </li> </ul>

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>string1 [PO 123-ABC]</li> </ul> </li> <li>Entity2 [1]                             <ul style="list-style-type: none"> <li>string1 []</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li><b>Entity1 [1]</b> <ul style="list-style-type: none"> <li>string1 [PO 123-ABC]</li> </ul> </li> <li><b>entity2 (Entity2) [2]</b> <ul style="list-style-type: none"> <li><b>string1 [item1]</b></li> </ul> </li> <li>Entity2 [1]                             <ul style="list-style-type: none"> <li>string1 []</li> </ul> </li> </ul>

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>string1 [PO 987-XYZ]</li> </ul> </li> <li>Entity1 [2]                             <ul style="list-style-type: none"> <li>string1</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>string1 [PO 987-XYZ]</li> </ul> </li> <li>Entity1 [2]                             <ul style="list-style-type: none"> <li>string1</li> </ul> </li> </ul>

# 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**).

Not.ers				
Conditions		0	1	2
a	Entity1.string1		'A'	not 'A'
b				
Actions		<		
Post Message(s)				
A	Entity1.boolean1		T	F
B				
Overrides				
Rule Statements				
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:

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]               <ul style="list-style-type: none"> <li>string1 [A]</li> </ul> </li> <li>Entity1 [2]               <ul style="list-style-type: none"> <li>string1 [123]</li> </ul> </li> <li>Entity1 [3]               <ul style="list-style-type: none"> <li>string1 [a]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]               <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>string1 [A]</li> </ul> </li> <li>Entity1 [2]               <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>string1 [123]</li> </ul> </li> <li>Entity1 [3]               <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>string1 [a]</li> </ul> </li> </ul>

## Limitations to using NOT in a Conditional cell

When you use **not** in a Conditional cell with an attribute name, the form is `not valueSet` which evaluates as `true` when the condition is not a member of an entry in the `valueSet`. Such entries in the `valueSet` must be literals (or partial expressions containing only literals); no variables or attributes may be included. Inclusion of an attribute reference in the `valueSet` is not valid.

Although `not attribute` is unsupported, it is not determined that it is invalid until it does not process. Then, it indicates that it is invalid.

Consider the following examples:

**Table 1: Valid usage**

Condition	Cell value
<code>foo.color</code>	<code>not 'red'</code>
<code>foo.color</code>	<code>&lt;&gt; 'red'</code>
<code>foo.color</code>	<code>&lt;&gt; bar.color</code>

**Table 2: Invalid usage**

Condition	Cell value
<code>foo.color</code>	<code>not bar.color</code>

## 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`.

The screenshot shows the NotEmpty.ers rule editor interface. It includes a Scope tree on the left with 'Entity1' and 'entity2 (Entity2) [collection1]'. The main area is divided into 'Conditions' and 'Actions' tables. The 'Conditions' table has rows 'a' through 'c', with row 'a' containing 'collection1 -> notEmpty'. The 'Actions' table has rows 'A' and 'B', with row 'A' containing 'Post Message(s)' and checkboxes for columns 1 and 2. Below these is an 'Overrides' section. At the bottom, the 'Rule Statements' table is visible, containing two entries with Ref, ID, Post, Alias, and Text columns.

Scope		Conditions	1	2
Entity1	entity2 (Entity2) [collection1]	a	collection1 -> notEmpty	T
		b		F
		c		
Actions				
		A	Post Message(s)	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
		B		
		Overrides		

Ref	ID	Post	Alias	Text
1		Warning	Entity1	collection1 is not empty, which means that Entity1 has at least one associated Entity2 element
2		Info	Entity1	collection1 is empty, which means that Entity1 has no associated Entity2 elements

### SAMPLE RULETEST

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

The screenshot shows the Rule Messages panel. It has two tabs: 'Rule Statements' and 'Rule Messages'. The 'Rule Messages' tab is active, showing a table with 'Severity' and 'Message' columns. The 'Input' and 'Output' panels above show tree views of entity collections. The 'Input' panel shows 'Entity1 [1]' and 'Entity1 [2]' with two 'entity2 (Entity2)' elements. The 'Output' panel shows the same structure. The 'Rule Messages' table has two rows: a 'Warning' message and an 'Info' message, both related to the 'collection1' not being empty or empty.

Severity	Message
Warning	collection1 is not empty, which means that Entity1 has at least one associated Entity2 element
Info	collection1 is empty, which means that Entity1 has no associated Entity2 elements

## Not equal to

### SYNTAX

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

## 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>.

## USAGE RESTRICTIONS

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

**Note: Use of < > when using custom data types** - If your Vocabulary uses custom data types, there are limits to the validity of < > in cells. In the following illustration, the `not` operator will validly work against a custom data type label, a value where a label is in use, and the value of a value-only definition. However, only the value where a label is in use is valid when < > is used.

The screenshot displays the Corticon Rule Editor interface. The top window, titled '\*NOT.ers', shows a rule table with the following content:

Conditions	1	2	3	4
a e1.LV	L1	not L1	<> L1	
b e1.LV	'LV1'	not 'LV1'	<> 'LV1'	
c e1.V	'V1'	not 'V1'	<> 'V1'	
d				
e e2.String1	'S1'	not 'S1'	<> 'S1'	
f e2.String2	'S2'	not 'S2'	<> 'S2'	
g				
h				

The bottom window, titled 'Simple.ecore', shows a tree view of the vocabulary structure and a 'Custom Data Types' table:

Data Type N...	Label	Value
LV	L1	'LV1'
V	L2	'LV2'

## 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.

NotEqualTo.ers				
Conditions		1	2	
a	Entity1.decimal1 <> Entity1.decimal2	T	F	
b				
Actions				
Post Message(s)				
A	Entity1.string1	'no match'	'match'	
B				
Overrides				
Rule Statements				
Ref	ID	Post	Alias	Text
1				If decimal1 does not equal decimal2, then assign a value of [no match] to string1
2				If decimal1 equals decimal2, then assign a value of [match] to string1

## SAMPLE RULETEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>decimal1 [1000.000000]</li> <li>decimal2 [1000.000000]</li> </ul> </li> <li>Entity1 [2]                             <ul style="list-style-type: none"> <li>decimal1 [123.400000]</li> <li>decimal2 [231.500000]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>decimal1 [1000.000000]</li> <li>decimal2 [1000.000000]</li> <li>string1 [match]</li> </ul> </li> <li>Entity1 [2]                             <ul style="list-style-type: none"> <li>decimal1 [123.400000]</li> <li>decimal2 [231.500000]</li> <li>string1 [no match]</li> </ul> </li> </ul>

# Now

## SYNTAX

now

## DESCRIPTION

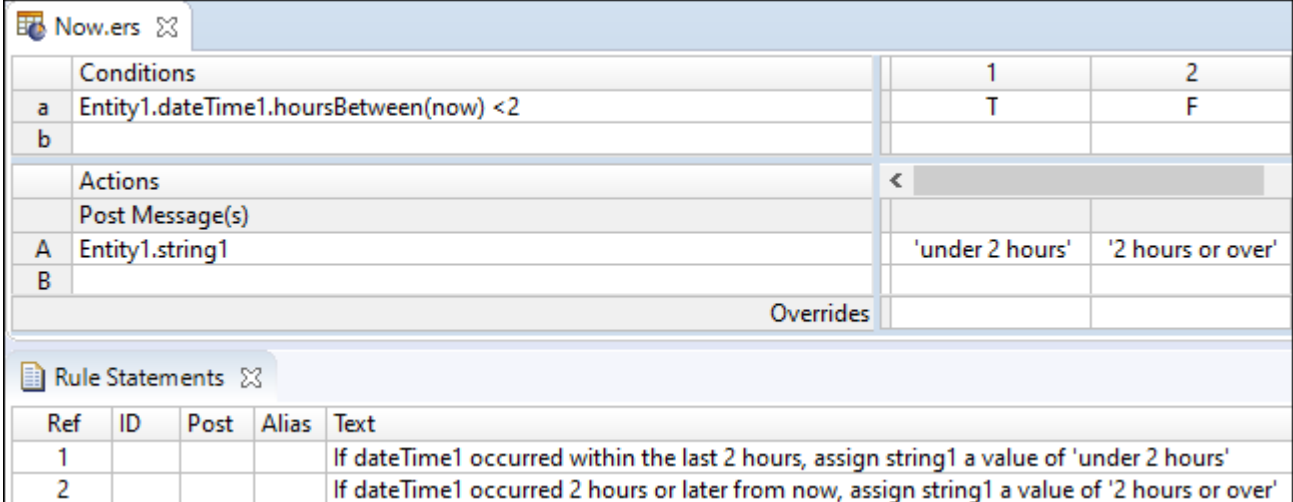
Returns the current system date and time when the rule is executed. This DateTime value is assigned the first time **now** is used in a Decision Service, then remains constant until the Decision Service finishes execution, regardless of how many additional times it is used. This means that every rule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



The screenshot shows a rulesheet editor for a rule named "Now.ers". It is divided into two main sections: "Conditions" and "Actions".

**Conditions:**

	1	2
a	T	F
b		

**Actions:**

	1	2
A	'under 2 hours'	'2 hours or over'
B		

**Rule Statements:**

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

## SAMPLE RULETEST

A sample Ruletest provides two examples of `dateTime1`. Assume **now** is equal to March 1, 2018 14:20:00 EST. Note that a future date in example 2 results in a negative value and therefore is under 2 hours. Input and Output panels are shown below. Notice the variation in DateTime masks (formats).

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [3/1/2018 16:30:00 EST]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [November 21, 2024 4:00:00 PM PST]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [3/1/2018 16:30:00 EST]</li> <li>string1 [2 hours or over]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [November 21, 2024 4:00:00 PM PST]</li> <li>string1 [under 2 hours]</li> </ul> </li> </ul>

# 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.

Conditions		0	1	2
a	Entity1.decimal1		null	other
b				
Actions		<		
Post Message(s)			✉	✉
A	Entity1.boolean1		T	F
B				
-				
Overrides				

Rule Statements				
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.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [4.000000]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>decimal1</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>decimal1 [0.000000]</li> </ul> </li> <li>Entity1 [4] <ul style="list-style-type: none"> <li>decimal1 [-13.000000]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>decimal1 [4.000000]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>decimal1</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>decimal1 [0.000000]</li> </ul> </li> <li>Entity1 [4] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>decimal1 [-13.000000]</li> </ul> </li> </ul>

Severity	Message
Warning	If decimal1 has the value of null, then assign boolean1 the value of true
Info	If decimal1 has any value other than null (any real number), then assign
Info	If decimal1 has any value other than null (any real number), then assign
Info	If decimal1 has any value other than null (any real number), then assign

## 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`.

Other.ers		0	1	2
Conditions				
a	Entity1.decimal1		null	other
b				
Actions		<		
Post Message(s)			✉	✉
A	Entity1.boolean1		T	F
B				
Overrides				

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 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
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [0.000000]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>decimal1</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>decimal1 [3.450000]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>decimal1 [0.000000]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>decimal1</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>decimal1 [3.450000]</li> </ul> </li> </ul>

Severity	Message
Warning	If decimal1 has the value of null, then assign boolean1 the value of true
Info	If decimal1 has any value other than null (any number), then assign boolean1
Info	If decimal1 has any value other than null (any number), then assign boolean1

Or

## SYNTAX

<Expression1> or <Expression2> or ....

### 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.

### RULESHEET EXAMPLE

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

Scope		Conditions		1	2
Entity1		a	Entity1.string1	'Jack'	'Jill'
		b			
Filters		Actions		<	
1	Entity1.integer1 < 10 or Entity1.boolean1	Post Message(s)			
2		A	Entity1.boolean2	F	T
3		B			
		Overrides			
Rule Statements					
Ref	ID	Post	Alias	Text	
1				If integer1 is less than 10, or boolean1 is true and string1 equals Jack, then boolean2 is false	
2				If integer1 is less than 10, or boolean1 is true and string1 equals Jill, then boolean2 is true	

### SAMPLE TEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>integer1 [5]</li> <li>string1 [Jack]</li> </ul> </li> <li>Entity1 [2]                             <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>integer1 [12]</li> <li>string1 [Jill]</li> </ul> </li> <li>Entity1 [3]                             <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>integer1 [45]</li> <li>string1 [Jack]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>boolean2 [false]</li> <li>integer1 [5]</li> <li>string1 [Jack]</li> </ul> </li> <li>Entity1 [2]                             <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>boolean2 [true]</li> <li>integer1 [12]</li> <li>string1 [Jill]</li> </ul> </li> <li>Entity1 [3]                             <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>integer1 [45]</li> <li>string1 [Jack]</li> </ul> </li> </ul>

# Random

## SYNTAX

```
<IntegerAttribute>.random(minRange,maxRange)
```

```
<DecimalAttribute>.random(minRange,maxRange)
```

## DESCRIPTION

Returns a random value between `minRange` and `maxRange`. Either range can be a numeric value of the same datatype, or numeric attributes of the same type; in which case, the attributes can have arithmetic operators, `absoluteValue`, and unary negative applied.

## USAGE RESTRICTIONS

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

## RULESHEET EXAMPLES

This sample Rulesheet uses **random** in non-conditional actions:

Conditions		0
a		
Actions		< [ ]
Post Message(s)		
A	Entity2.decimal1=Entity1.decimal1.random(100.0,200.0)	✓
B	Entity2.decimal2=Entity1.decimal1.random(0.0,10000.0)	✓
C	Entity2.decimal3=Entity1.decimal1.random(-1000.0,-100.0)	✓
D	Entity2.decimal4=Entity1.decimal1.random(-1000.0,Entity2.decimal2)	✓
E	Entity2.decimal5=Entity1.decimal1.random(0.0,Entity2.decimal3.absVal)	✓
F	Entity2.decimal6=Entity1.decimal1.random(Entity2.decimal1, 1000000.0)	✓
G	Entity2.decimal7=Entity1.decimal1.random(Entity2.decimal3,Entity2.decimal2)	✓
H	Entity2.decimal8=Entity1.decimal1.random(Entity2.decimal3*10.0,Entity2.decimal2*10.0)	✓
I	Entity2.decimal9=Entity1.decimal1.random(20.0,10.0)	✓
J		
K	Entity2.integer1=Entity1.integer1.random(100,200)	✓
L	Entity2.integer2=Entity1.integer1.random(0,100)	✓
M	Entity2.integer3=Entity1.integer1.random(-105,-100)	✓
N	Entity2.integer4=Entity1.integer1.random(-1000,Entity2.integer2)	✓
O	Entity2.integer5=Entity1.integer1.random(0,Entity2.integer3.absVal)	✓
P	Entity2.integer6=Entity1.integer1.random(Entity2.integer1, 1000000)	✓
Q	Entity2.integer7=Entity1.integer1.random(Entity2.integer3,Entity2.integer2)	✓
R	Entity2.integer8=Entity1.integer1.random(Entity2.integer3*10,Entity2.integer2*10)	✓
S	Entity2.integer9=Entity1.integer1.random(20,10)	✓
T		
Overrides		

## SAMPLE RULETEST

A sample Ruletest requires values for Entity1 although they have no impact on the output. As the result is random, there cannot be an expected value.

Input	Output
<div style="margin-left: 20px;"> <div style="margin-left: 20px;">Entity1 [1]               <ul style="list-style-type: none"> <li>decimal1 [0.000000]</li> <li>integer1 [0]</li> </ul> </div> <div style="margin-left: 20px;">Entity2 [1]               <ul style="list-style-type: none"> <li>decimal1</li> <li>decimal2</li> <li>decimal3</li> <li>decimal4</li> <li>decimal5</li> <li>decimal6</li> <li>decimal7</li> <li>decimal8</li> <li>decimal9</li> <li>integer1</li> <li>integer2</li> <li>integer3</li> <li>integer4</li> <li>integer5</li> <li>integer6</li> <li>integer7</li> <li>integer8</li> <li>integer9</li> </ul> </div> </div>	<div style="margin-left: 20px;"> <div style="margin-left: 20px;">Entity1 [1]               <ul style="list-style-type: none"> <li>decimal1 [0.000000]</li> <li>integer1 [0]</li> </ul> </div> <div style="margin-left: 20px;">Entity2 [1]               <ul style="list-style-type: none"> <li>decimal1 [155.863422]</li> <li>decimal2 [3715.800005]</li> <li>decimal3 [-815.275958]</li> <li>decimal4 [-206.761569]</li> <li>decimal5 [294.585655]</li> <li>decimal6 [141411.435602]</li> <li>decimal7 [1757.651933]</li> <li>decimal8 [4063.641385]</li> <li>decimal9 [11.963497]</li> <li>integer1 [148]</li> <li>integer2 [42]</li> <li>integer3 [-103]</li> <li>integer4 [-872]</li> <li>integer5 [81]</li> <li>integer6 [958580]</li> <li>integer7 [-12]</li> <li>integer8 [52]</li> <li>integer9 [18]</li> </ul> </div> </div>

## 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.

The `.remove` operator's impact on elements of a collection can be controlled:

- When the operator is written as `.remove`, `.remove()`, or `.remove(true)`, any lower-level associated entities are also removed. For an example of this behavior, see [example 2](#) below.
- When the operator is written as `.remove(false)`, lower-level associated entities are promoted to root level. For an example of this behavior, see [example 3](#) 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](#)).

### EXAMPLE 1: Remove an element from a collection

#### RULESHEET 1

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`.

The screenshot shows the 'Remove.ers' rulesheet editor. The 'Scope' panel on the left shows a tree structure: Entity1 (expanded) contains entity2 (Entity2) [collection1] (expanded), which contains decimal1. The 'Conditions' table has two rows: 'a' with the condition 'collection1.decimal1 > 5.0' and 'b' which is empty. The 'Actions' table has two rows: 'A' with the action 'collection1.remove' and a checked checkbox, and 'B' which is empty. The 'Rule Statements' table at the bottom has one row with Ref '1', ID, Post, Alias, and Text 'Remove an element from collection1 whose decimal1 value is greater than 5'.

Scope		Conditions	1	2
Entity1	a	collection1.decimal1 > 5.0	T	F
entity2 (Entity2) [collection1]	b			
decimal1				

Filters		Actions	1	2
1		Post Message(s)		
2		A collection1.remove	<input checked="" type="checkbox"/>	<input type="checkbox"/>
		B		
		Overrides		

Ref	ID	Post	Alias	Text
1				Remove an element from collection1 whose decimal1 value is greater than 5

#### RULETEST 1

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [4.500000]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [5.001000]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [3.200000]</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [4.500000]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [3.200000]</li> </ul> </li> </ul> </li> </ul>

### EXAMPLE 2: Remove an entity and its children

#### RULESHEET 2

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

Scope		Conditions		1	2
Entity1	a	Entity1.entity2.decimal1 > 5		T	F
entity2 (Entity2)	b				
decimal1					
Filters		Actions			
1	A	Entity1.entity2.remove		<input checked="" type="checkbox"/>	<input type="checkbox"/>
2	B				
			Overrides		

Ref	ID	Post	Alias	Text
1				Remove any element from the collection whose decimal1 value is greater than 5.

### 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.

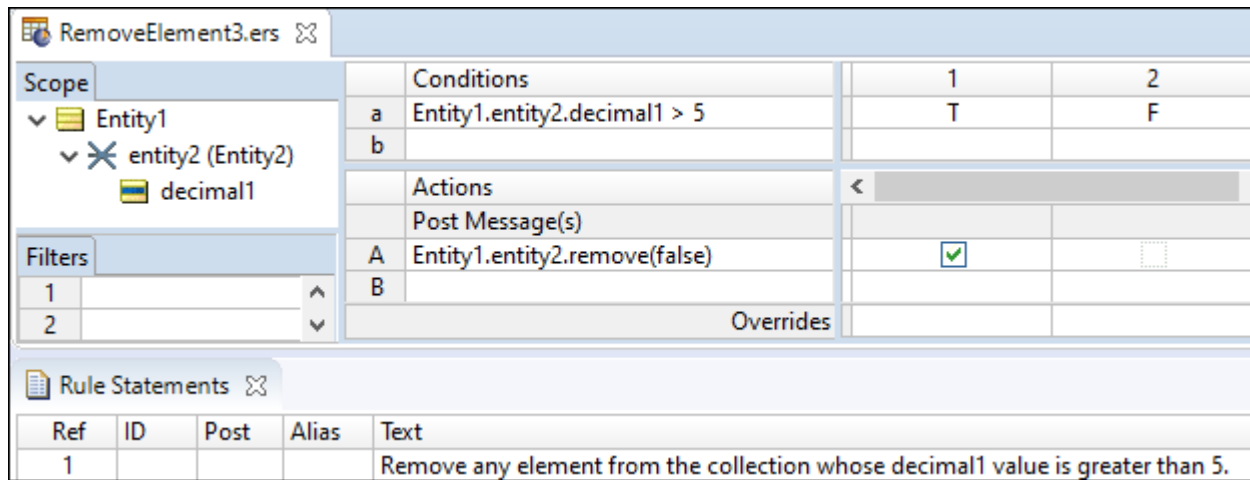
Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [2.000000]</li> <li>entity3 (Entity3) [1] <ul style="list-style-type: none"> <li>string1 [A]</li> </ul> </li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [6.000000]</li> <li>entity3 (Entity3) [2] <ul style="list-style-type: none"> <li>string1 [B]</li> </ul> </li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [2.000000]</li> <li>entity3 (Entity3) [1] <ul style="list-style-type: none"> <li>string1 [A]</li> </ul> </li> </ul> </li> </ul> </li> </ul>

**Note:** Removing an entity and its children removes child entities from the work document only, not from working memory. If rules are written so as to access the child entities directly, they will still execute after the parent has been removed.

### EXAMPLE 3: Remove an entity then promote its children

#### RULESHEET 3

This Rulesheet uses the operator with its (false) parameter to remove only the specified 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.



Scope		Conditions		1	2
Entity1	a	Entity1.entity2.decimal1 > 5		T	F
entity2 (Entity2)	b				
decimal1	Actions				
Post Message(s)					
A Entity1.entity2.remove(false)			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
B					
Overrides					

Ref	ID	Post	Alias	Text
1				Remove any element from the collection whose decimal1 value is greater than 5.

### RULETEST 3

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 promoted to root level.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [2.000000]</li> <li>entity3 (Entity3) [1] <ul style="list-style-type: none"> <li>string1 [A]</li> </ul> </li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [6.000000]</li> <li>entity3 (Entity3) [2] <ul style="list-style-type: none"> <li>string1 [B]</li> </ul> </li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [2.000000]</li> <li>entity3 (Entity3) [1] <ul style="list-style-type: none"> <li>string1 [A]</li> </ul> </li> </ul> </li> <li>Entity3 [2] <ul style="list-style-type: none"> <li>string1 [B]</li> </ul> </li> </ul> </li> </ul>

## Replace elements

### 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. The association between Entity2 and Entity3 has a cardinality of “one-to-one”. If multiple Entity3 are present, only one will be added to collection1.

The screenshot shows the 'ReplaceElement.ers' rule editor. It features a tree view on the left under 'Scope' showing 'Entity2' containing 'entity3 (Entity3) [collection1]' and 'Entity3'. Below the tree is a 'Filters' section with two filter slots. The main area is divided into 'Conditions' and 'Actions' sections. The 'Conditions' table has three rows (a, b, c) and two columns (1, 2). Row 'a' has 'Entity3.boolean1' in column 1 and 'T' in column 2. Row 'b' is empty. Row 'c' is empty. The 'Actions' section has a 'Post Message(s)' section with two rows (A, B) and two columns (1, 2). Row 'A' has 'collection1 = Entity3' in column 1 and a checked checkbox in column 2. Row 'B' is empty. Below the actions is an 'Overrides' section. At the bottom is a 'Rule Statements' table with columns 'Ref', 'ID', 'Post', 'Alias', and 'Text'. Row 1 has '1' in Ref, 'ID' in ID, 'Post' in Post, 'Alias' in Alias, and 'If boolean1 value of Entity3 is true, then add the element to collection1' in Text. Row 2 has '2' in Ref, 'ID' in ID, 'Post' in Post, 'Alias' in Alias, and 'If boolean1 value of Entity3 is false, then take no action' in Text.

Conditions		1	2
a	Entity3.boolean1	T	F
b			
c			

Actions		1	2
Post Message(s)			
A	collection1 = Entity3	<input checked="" type="checkbox"/>	<input type="checkbox"/>
B			

Ref	ID	Post	Alias	Text
1				If boolean1 value of Entity3 is true, then add the element to collection1
2				If boolean1 value of Entity3 is false, then take no action

### SAMPLE TEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity2 [1]                             <ul style="list-style-type: none"> <li>entity3 (Entity3) [1]                                     <ul style="list-style-type: none"> <li>boolean1 [false]</li> </ul> </li> </ul> </li> <li>Entity3 [2]                             <ul style="list-style-type: none"> <li>boolean1 [true]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity2 [1]                             <ul style="list-style-type: none"> <li>entity3 (Entity3) [2]                                     <ul style="list-style-type: none"> <li>boolean1 [true]</li> </ul> </li> </ul> </li> <li>Entity3 [1]                             <ul style="list-style-type: none"> <li>boolean1 [false]</li> </ul> </li> </ul>

Input	Output
<ul style="list-style-type: none"> <li>Entity2 [1]</li> <li>Entity3 [1]                             <ul style="list-style-type: none"> <li>boolean1 [true]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity2 [1]                             <ul style="list-style-type: none"> <li>entity3 (Entity3) [1]                                     <ul style="list-style-type: none"> <li>boolean1 [true]</li> </ul> </li> </ul> </li> </ul>

Input	Output
<ul style="list-style-type: none"> <li>Entity2 [1]</li> <li>Entity3 [1]                             <ul style="list-style-type: none"> <li>boolean1 [false]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity2 [1]</li> <li>Entity3 [1]                             <ul style="list-style-type: none"> <li>boolean1 [false]</li> </ul> </li> </ul>

# Replace String

## SYNTAX

```
<String>.replaceString(stringToBeReplaced,replacementString)
```

## DESCRIPTION

Returns a new string where the instances of the String to be replaced are replaced by the value of the replacement String.

## 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 **replaceString** in non-conditional actions.

replaceString.ers		0	1
Conditions			
a			
b			
c			
Actions		<	
Post Message(s)			
A	Entity1.string2 = Entity1.string1.replaceString('red', 'blue')	<input checked="" type="checkbox"/>	
B	Entity2.string2 = Entity3.string1.replaceString('red', 'dark blue')	<input checked="" type="checkbox"/>	
C	Entity3.string2 = Entity3.string1.replaceString('dark blue', 'red')	<input checked="" type="checkbox"/>	
Overrides			
Rule Statements			
Ref	ID	Post	Text
A0			Instances of 'red' in string2 of Entity1 are replaced with 'blue'
B0			Instances of 'red' in string2 of Entity2 are replaced with 'dark blue'
C0			Instances of 'dark blue' in string2 of Entity3 are replaced with 'red'

## SAMPLE RULETEST

A sample Ruletest shows the `replaceString` effect in output.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [a red balloon]</li> </ul> </li> <li>Entity2 [1] <ul style="list-style-type: none"> <li>string1 [a red balloon]</li> </ul> </li> <li>Entity3 [1] <ul style="list-style-type: none"> <li>string1 [a dark blue balloon]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [a red balloon]</li> <li>string2 [a blue balloon]</li> </ul> </li> <li>Entity2 [1] <ul style="list-style-type: none"> <li>string1 [a red balloon]</li> <li>string2 [a dark blue balloon]</li> </ul> </li> <li>Entity3 [1] <ul style="list-style-type: none"> <li>string1 [a dark blue balloon]</li> <li>string2 [a red balloon]</li> </ul> </li> </ul>

## Regular expression to replace String

### SYNTAX

```
<String>.regexReplaceString(regularExpression, replacementString)
```

### DESCRIPTION

Returns a new String where the strings matching the regular expression are replaced by the replacement string.

---

**Note:** Regular expressions are a well-established technique that uses a sequence of characters to define a search pattern. For more information, see Wikipedia, as well one of the many sites that provide examples, such as [regular-expressions.info](http://regular-expressions.info), and others that analyze the expressions you create.

---

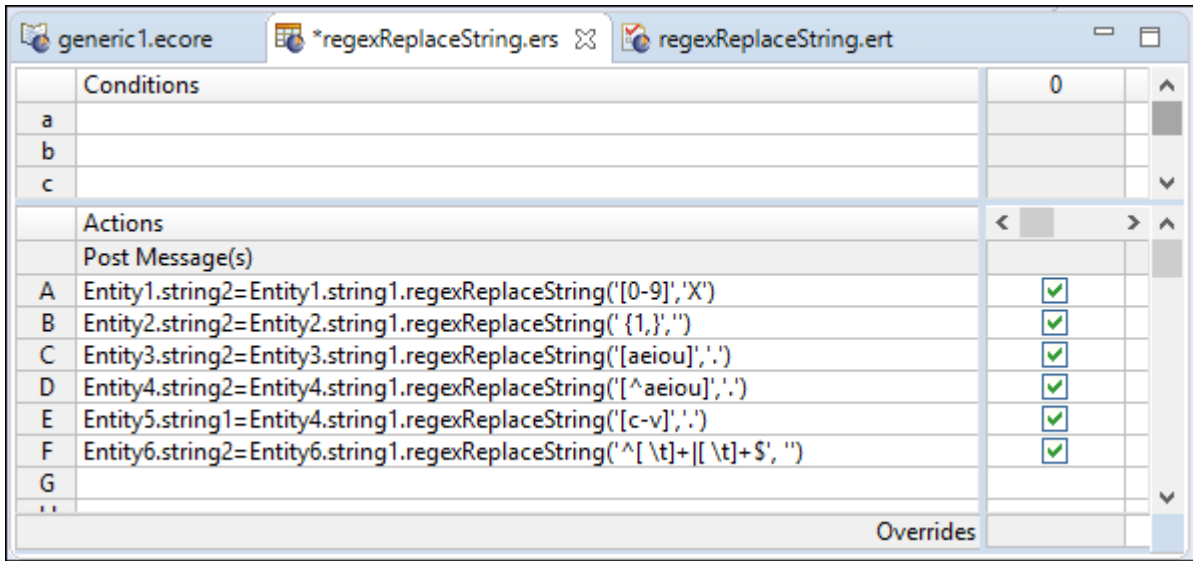
### USAGE RESTRICTIONS

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

### RULESHEET EXAMPLE

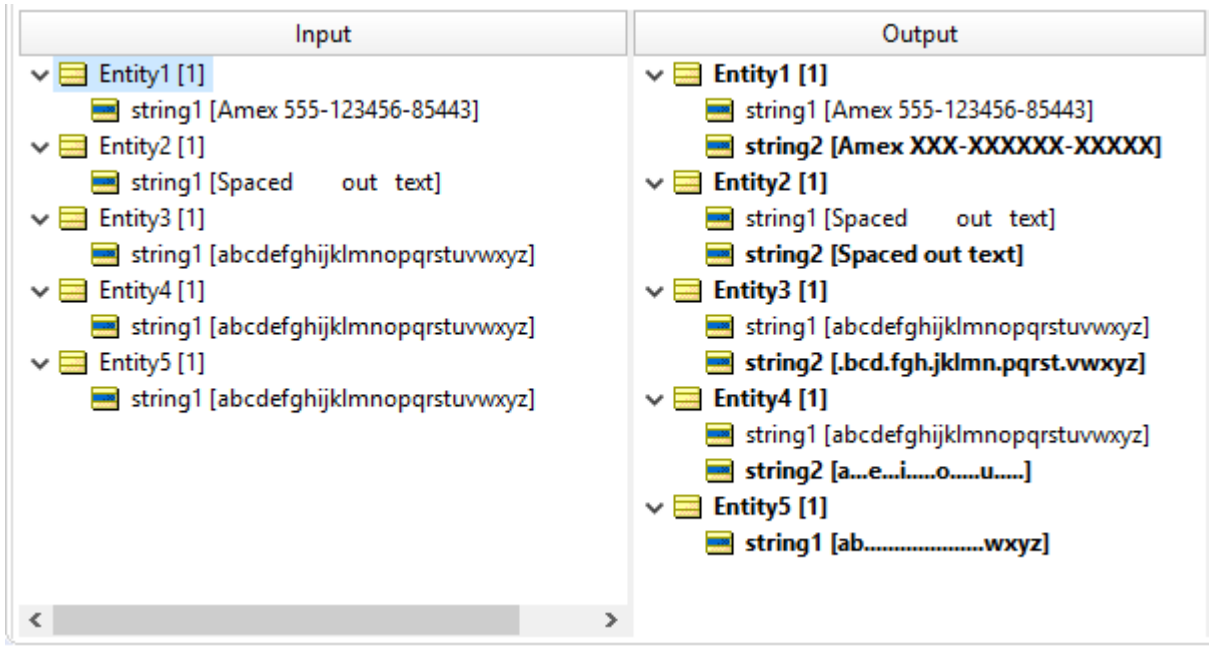
This sample Rulesheet uses **regexReplaceString** in non-conditional actions as follows:

- `regexReplaceString("[0-9]", "X")`: Replace all instances of digits with the character X
- `regexReplaceString(" {2,}", " ")` - Replace all instances of multiple spaces with a single space
- `regexReplaceString("[aeiou]", ".")` - Replace all vowels with a dot .
- `regexReplaceString("[^aeiou]", ".")` - Replace all non-vowel characters with a dot.
- `regexReplaceString("[c-v]", ".")` - Replace each character in the range from c to v with a dot.
- `regexReplaceString('^[ \t]+|[ \t]+$')` - Strip off leading and trailing spaces.



### SAMPLE RULETEST

A sample Ruletest shows the regexReplaceString effect in output.



## 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 2<sup>nd</sup> decimal place, and assigns it to `decimal1`.

The screenshot shows a rulesheet editor with the following components:

- Conditions:** A table with columns 'a' and 'b', currently empty.
- Actions:** A table with columns 'A' and 'B'. Action A contains the expression `Entity1.decimal1 = Entity1.decimal2.round(2)` and has a checkmark in the 'B' column.
- Rule Statements:** A table with columns 'Ref', 'ID', 'Post', 'Alias', and 'Text'. Row A0 contains the text: `decimal1 is assigned the value of decimal2 rounded to two decimal places`.

## SAMPLE TEST

A sample Ruletest provides results for five examples of `decimal2`.

Input	Output
Entity1 [1] decimal2 [1550.785000]	Entity1 [1] <b>decimal1 [1550.790000]</b> decimal2 [1550.785000]
Entity1 [2] decimal2 [2200.986000]	Entity1 [2] <b>decimal1 [2200.990000]</b> decimal2 [2200.986000]
Entity1 [3] decimal2 [-500.990000]	Entity1 [3] <b>decimal1 [-500.990000]</b> decimal2 [-500.990000]
Entity1 [4] decimal2 [-5.123000]	Entity1 [4] <b>decimal1 [-5.120000]</b> decimal2 [-5.123000]
Entity1 [5] decimal2 [12.345600]	Entity1 [5] <b>decimal1 [12.350000]</b> decimal2 [12.345600]

# 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.

The screenshot shows a rulesheet editor window titled "Second.ers". It contains a table with "Conditions" and "Actions" sections. The "Conditions" section has two rows labeled "a" and "b". The "Actions" section has two rows labeled "A" and "B". Row "A" contains the action "Entity1.integer1 = Entity1.dateTime1.sec" with a green checkmark. Below the actions is an "Overrides" section. At the bottom, there is a "Rule Statements" section with a table:

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.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [March 12, 2022 17:00:23 EST]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [12/2/2024 2:29:45 PM PST]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [March 12, 2022 17:00:23 EST]</li> <li>integer1 [23]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [12/2/2024 2:29:45 PM PST]</li> <li>integer1 [45]</li> </ul> </li> </ul>

# 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`.

SecondsBetween.ers				
Conditions		1	2	
a	Entity1.dateTime1.secsBetween(Entity1.dateTime2)	<= 60	> 60	
b				
Actions		<		
Post Message(s)				
A	Entity1.string1	'Not Overdue'	'Overdue'	
B				
Overrides				
Rule Statements				
Ref	ID	Post	Alias	Text
1				If 60 or fewer seconds have elapsed between <code>dateTime1</code> and <code>dateTime2</code> , then Entity1 is Not Overdue
2				If more than 60 seconds have elapsed between <code>dateTime1</code> and <code>dateTime2</code> , then Entity1 is Overdue

## SAMPLE TEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [3/12/2018 17:00:23]</li> <li>dateTime2 [4/24/2019 10:00:00]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [12/2/2023 2:29:45 PM PST]</li> <li>dateTime2 [12/2/2023 2:30:00 PM PST]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [3/12/2018 17:00:23]</li> <li>dateTime2 [4/24/2019 10:00:00]</li> <li>string1 [Overdue]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [12/2/2023 2:29:45 PM PST]</li> <li>dateTime2 [12/2/2023 2:30:00 PM PST]</li> <li>string1 [Not Overdue]</li> </ul> </li> </ul>

## 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`

SizeOfString.ers				
Conditions				0
a				
b				
Actions				<
Post Message(s)				
A		Entity1.integer1 = Entity1.string1.size		<input checked="" type="checkbox"/>
B				
Overrides				
Rule Statements				
Ref	ID	Post	Alias	Text
A0				integer1 equals the number of characters in string1

### SAMPLE TEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [goodbye]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [hello!]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [next week]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>integer1 [7]</li> <li>string1 [goodbye]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>integer1 [6]</li> <li>string1 [hello!]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>integer1 [9]</li> <li>string1 [next week]</li> </ul> </li> </ul>

## 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	1	2
a	collection1 -> size	< 12	>= 12
b			
Filters	Actions		
A	Entity1.boolean2	F	T
B			
	Overrides		

Ref	ID	Post	Alias	Text
1				If there are fewer 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.

Input	Output
<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>◇ entity2 (Entity2) [1]</li> <li>◇ entity2 (Entity2) [2]</li> <li>◇ entity2 (Entity2) [3]</li> <li>◇ entity2 (Entity2) [4]</li> <li>◇ entity2 (Entity2) [5]</li> <li>◇ entity2 (Entity2) [6]</li> <li>◇ entity2 (Entity2) [7]</li> <li>◇ entity2 (Entity2) [8]</li> <li>◇ entity2 (Entity2) [9]</li> <li>◇ entity2 (Entity2) [10]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>◇ entity2 (Entity2) [11]</li> <li>◇ entity2 (Entity2) [12]</li> <li>◇ entity2 (Entity2) [13]</li> <li>◇ entity2 (Entity2) [14]</li> <li>◇ entity2 (Entity2) [15]</li> <li>◇ entity2 (Entity2) [16]</li> <li>◇ entity2 (Entity2) [17]</li> <li>◇ entity2 (Entity2) [18]</li> <li>◇ entity2 (Entity2) [19]</li> <li>◇ entity2 (Entity2) [20]</li> <li>◇ entity2 (Entity2) [21]</li> <li>◇ entity2 (Entity2) [22]</li> <li>◇ entity2 (Entity2) [23]</li> <li>◇ entity2 (Entity2) [24]</li> <li>◇ entity2 (Entity2) [25]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>boolean2 [false]</li> <li>◇ entity2 (Entity2) [1]</li> <li>◇ entity2 (Entity2) [2]</li> <li>◇ entity2 (Entity2) [3]</li> <li>◇ entity2 (Entity2) [4]</li> <li>◇ entity2 (Entity2) [5]</li> <li>◇ entity2 (Entity2) [6]</li> <li>◇ entity2 (Entity2) [7]</li> <li>◇ entity2 (Entity2) [8]</li> <li>◇ entity2 (Entity2) [9]</li> <li>◇ entity2 (Entity2) [10]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>boolean2 [true]</li> <li>◇ entity2 (Entity2) [11]</li> <li>◇ entity2 (Entity2) [12]</li> <li>◇ entity2 (Entity2) [13]</li> <li>◇ entity2 (Entity2) [14]</li> <li>◇ entity2 (Entity2) [15]</li> <li>◇ entity2 (Entity2) [16]</li> <li>◇ entity2 (Entity2) [17]</li> <li>◇ entity2 (Entity2) [18]</li> <li>◇ entity2 (Entity2) [19]</li> <li>◇ entity2 (Entity2) [20]</li> <li>◇ entity2 (Entity2) [21]</li> <li>◇ entity2 (Entity2) [22]</li> <li>◇ entity2 (Entity2) [23]</li> <li>◇ entity2 (Entity2) [24]</li> <li>◇ entity2 (Entity2) [25]</li> </ul> </li> </ul>

## 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 in Unicode and Java Collator](#) on page 217. <Collection> must be expressed as a unique alias.

See [Advanced collection sorting syntax](#) "Advanced collection sorting syntax" and [Statement blocks](#) "Statement blocks" in the *Rule Modeling Guide* 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`.

Scope		Conditions	1	2
a	collection1 -> sortedBy (decimal1) -> first.string1		'Joe'	not 'Joe'
b				
Filters		Actions		
1		Entity1.boolean1	T	F
2				
		Overrides		

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

## 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.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [2.500000]</li> <li>string1 [Joe]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [5.800000]</li> <li>string1 [Mary]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [3.300000]</li> <li>string1 [Sue]</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [true] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [2.500000]</li> <li>string1 [Joe]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [5.800000]</li> <li>string1 [Mary]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [3.300000]</li> <li>string1 [Sue]</li> </ul> </li> </ul> </li> </ul> </li></ul>

## 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`.

Scope		Conditions		1	2
Entity1		a	Entity1.boolean1	T	F
boolean1		b			
entity2 (Entity2) [collection1]		Actions			
string1		Post Message(s)			
		A	collection1 -> sortedBy (decimal1) -> first.string1	'Joe'	'Mary'
		B			
		Overrides			
Filters					
1					
2					
Rule Statements					
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.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [8.500000]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [5.800000]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [3.300000]</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [8.500000]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [5.800000]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [3.300000]</li> <li>string1 [Mary]</li> </ul> </li> </ul> </li> </ul>

## 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 in Unicode and Java Collator](#) on page 217. `<Collection>` must be expressed as a unique alias.

See [Advanced collection sorting syntax](#) "Advanced collection sorting syntax" and [Statement blocks](#) "Statement blocks" in the *Rule Modeling Guide* 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`.

Scope		Conditions	1	2
Entity1	a	collection1 -> sortedByDesc (decimal1) -> first.string1	'Joe'	not 'Joe'
boolean1	b			
entity2 (Entity2) [collection1]	Actions		<	
decimal1	A	Entity1.boolean1	T	F
string1	B			
	C			
	Overrides			

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.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [2.500000]</li> <li>string1 [Joe]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [5.800000]</li> <li>string1 [Mary]</li> </ul> </li> <li>entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>decimal1 [3.300000]</li> <li>string1 [Sue]</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [2.500000]</li> <li>string1 [Joe]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [5.800000]</li> <li>string1 [Mary]</li> </ul> </li> <li>entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>decimal1 [3.300000]</li> <li>string1 [Sue]</li> </ul> </li> </ul> </li> </ul>

## 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`.

Conditions		1	2
a	Entity1.boolean1	T	F
b			
Actions		<	
Post Message(s)			
A	collection1 -> sortedByDesc(decimal1) -> first.string1	'Joe'	'Mary'
B			
Overrides			

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 2

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [2.500000]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [5.800000]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [3.300000]</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [2.500000]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [5.800000]</li> <li>string1 [Joe]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [3.300000]</li> </ul> </li> </ul> </li> </ul>

## 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.

Conditions		0	1	2
a	Entity1.string1.startsWith(string2)		T	F
b				
Actions		<		
Post Message(s)				
A	Entity1.boolean1		T	F
B				
Overrides				

Ref	ID	Post	Alias	Text
1				If string1 starts with string2, then boolean1 is true
2				If string1 does not start with string2, then boolean1 is false

## SAMPLE TEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [Strongsville]</li> <li>string2 [happy]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [New York]</li> <li>string2 [New]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [Amityville]</li> <li>string2 [A]</li> </ul> </li> <li>Entity1 [4] <ul style="list-style-type: none"> <li>string1 [Amityville]</li> <li>string2 [ami]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>string1 [Strongsville]</li> <li>string2 [happy]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>string1 [New York]</li> <li>string2 [New]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>string1 [Amityville]</li> <li>string2 [A]</li> </ul> </li> <li>Entity1 [4] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>string1 [Amityville]</li> <li>string2 [ami]</li> </ul> </li> </ul>

# 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 sorting syntax](#) "Advanced collection sorting syntax" in the *Rule Modeling Guide* 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.

The screenshot shows a Rulesheet editor window titled "SubSequence.ers". The interface is divided into several sections:

- Scope:** A tree view showing a hierarchy of entities. "Entity1" is expanded to show "entity2 (Entity2) [collection2]" and "decimal1". "Entity3" is also expanded to show "entity2 (Entity2) [collection3]".
- Filters:** A table with two rows, numbered 1 and 2, with empty input fields.
- Conditions:** A table with three rows labeled 'a', 'b', and 'c', all of which are empty.
- Actions:** A table with two rows labeled 'A' and 'B'. Row 'A' contains the text "collection3 = collection2 -> sortedBy (collection2.decimal1) -> subSequence(3,4)" and has a green checkmark in the rightmost column. Row 'B' is empty.
- Overrides:** A table with one empty row.
- Rule Statements:** A table with columns "Ref", "ID", "Post", "Alias", and "Text". The first row has "A0" in the "Ref" column and "Assign the 3rd and 4th elements of collection2, after sorting, to collection3" in the "Text" column.

## 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"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>▼ entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [500.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [800.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [600.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>decimal1 [700.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [5] <ul style="list-style-type: none"> <li>decimal1 [100.000000]</li> </ul> </li> </ul> </li> <li>Entity3 [1]</li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>▼ entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [500.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [800.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [600.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>decimal1 [700.000000]</li> </ul> </li> <li>▼ entity2 (Entity2) [5] <ul style="list-style-type: none"> <li>decimal1 [100.000000]</li> </ul> </li> </ul> </li> <li>▼ Entity3 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [3]</li> <li>entity2 (Entity2) [4]</li> </ul> </li> </ul>

**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`.

Conditions		0
a		
b		
Actions		<
Post Message(s)		
A	Entity1.string2 = Entity1.string1.substring(4,7)	<input checked="" type="checkbox"/>
B		
Overrides		

Ref	ID	Post	Alias	Text
A0				string2 equals the portion of string1 delimited by the 4th and 7th character positions

## SAMPLE RULETEST

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

Input	Output	Expected
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [howitzer]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [superSize]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [piglets]</li> </ul> </li> <li>Entity1 [4] <ul style="list-style-type: none"> <li>string1 [cowardice]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [howitzer]</li> <li>string2 [itze]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [superSize]</li> <li>string2 [erSi]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [piglets]</li> <li>string2 [lets]</li> </ul> </li> <li>Entity1 [4] <ul style="list-style-type: none"> <li>string1 [cowardice]</li> <li>string2 [ardi]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [howitzer]</li> <li>string2 [itze]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [superSize]</li> <li>string2 [erSi]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [piglets]</li> <li>string2 [lets]</li> </ul> </li> <li>Entity1 [4] <ul style="list-style-type: none"> <li>string1 [cowardice]</li> <li>string2 [ardi]</li> </ul> </li> </ul>

# 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`

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

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.

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [23.000000]</li> <li>decimal2 [7.200000]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>decimal1 [10.300000]</li> <li>decimal2 [41.670000]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>decimal1 [-4.560000]</li> <li>decimal2 [-8.120000]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>decimal1 [23.000000]</li> <li>decimal2 [7.200000]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>decimal1 [10.300000]</li> <li>decimal2 [41.670000]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>decimal1 [-4.560000]</li> <li>decimal2 [-8.120000]</li> </ul> </li> </ul>

# 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`

The screenshot shows the 'Sum.ers' interface. On the left, a tree view shows the scope: Entity1, entity2 (Entity2) [collection1], and decimal1. Below this are filters and a table of rule statements. On the right, a configuration table shows conditions and actions for two rule instances.

Scope		Conditions	1	2
a	collection1.decimal1 -> sum	< 9	>= 9	
b				
Filters		Actions		
1		Post Message(s)	✉	✉
2		A		
		B		
		Overrides		

Ref	ID	Post	Alias	Text
1		Info	Entity1	If the sum of decimal1 in collection1 is less than 9, then post an info message
2		Warning	Entity1	If the sum of decimal1 in collection1 is greater than or equal to 9, then post a warning message

## SAMPLE TEST

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

The screenshot shows the 'Ruletest' interface. It has two main panels: 'Input' and 'Output'. Below these is a 'Rule Messages' panel showing a table of messages.

Input		Output	
Entity1 [1]	entity2 (Entity2) [1]	Entity1 [1]	entity2 (Entity2) [1]
	decimal1 [1.200000]		decimal1 [1.200000]
entity2 (Entity2) [2]	entity2 (Entity2) [2]	entity2 (Entity2) [2]	entity2 (Entity2) [2]
	decimal1 [2.700000]		decimal1 [2.700000]
entity2 (Entity2) [3]	entity2 (Entity2) [3]	entity2 (Entity2) [3]	entity2 (Entity2) [3]
	decimal1 [3.500000]		decimal1 [3.500000]

Severity	Message	Entity
Info	If the sum of decimal1 in collection1 is less than 9, then post an info message	Entity1[1]

# 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.

today.ers		0	1	2	3
Conditions					
a	Entity1.dateOnly1.daysBetween(today)<5		T	F	
b					
c					
Actions		<			
Post Message(s)					
A	Entity1.string1		'under 5 days'	'5 days or more'	
B					
C					
Overrides					

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

## SAMPLE TEST

A sample Ruletest provides three examples of `dateOnly1`. Assume **today** is equal to August 9, 2020. Input and Output panels are shown below:

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateOnly1 [8/5/2020]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateOnly1 [7/15/2020]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>dateOnly1 [8/8/20]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateOnly1 [8/5/2020]</li> <li>string1 [under 5 days]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateOnly1 [7/15/2020]</li> <li>string1 [5 days or more]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>dateOnly1 [8/8/20]</li> <li>string1 [under 5 days]</li> </ul> </li> </ul>

## 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.

Casting DateTime to DateOnly.ers				
Conditions				0
a				
b				
Actions				<
Post Message(s)				
A	Entity1.dateOnly1 = Entity1.dateTime1.toDate			<input checked="" type="checkbox"/>
B	Entity1.dateOnly2 = Entity1.dateTime2.toDate			<input checked="" type="checkbox"/>
Overrides				
Rule Statements				
Ref	ID	Post	Alias	Text
A0				dateOnly1 is equal to the date value of dateTime1, if it has one
B0				dateOnly2 is equal to the date value of dateTime2

## SAMPLE TEST

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateOnly1</li> <li>dateOnly2</li> <li>dateTime1 [1/1/2022 3:45:00 AM EST]</li> <li>dateTime2 [April 10, 2024 2:29:00 AM EDT]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateOnly1</li> <li>dateOnly2</li> <li>dateTime1 [4/10/2024 3:45:00 AM EST]</li> <li>dateTime2 [4/10/2024 20:00:00 PST]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateOnly1 [1/1/2022]</li> <li>dateOnly2 [April 10, 2024]</li> <li>dateTime1 [1/1/2022 3:45:00 AM EST]</li> <li>dateTime2 [April 10, 2024 2:29:00 AM EDT]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateOnly1 [4/10/2024]</li> <li>dateOnly2 [4/10/2024]</li> <li>dateTime1 [4/10/2024 3:45:00 AM EST]</li> <li>dateTime2 [4/10/2024 20:00:00 PST]</li> </ul> </li> </ul>

## 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`.

Conditions		0	1
a			
b			
Actions		<	
Post Message(s)			
A	Entity1.dateTime1 = Entity1.string1.toDateTime	<input checked="" type="checkbox"/>	
B			
Overrides			
Rule Statements			
Ref	ID	Post	Text
A0			dateTime1 is equal to string1 converted to a dateTime data type

## SAMPLE TEST

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [12/31/2021]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [January 29, 2022]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [Thursday, June 2, 2022 1:00:00 PM PDT]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [12/31/2021 12:00:00 AM]</li> <li>string1 [12/31/2021]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [January 29, 2022 12:00:00 AM]</li> <li>string1 [January 29, 2022]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>dateTime1 [Thursday, June 2, 2022 1:00:00 PM PDT]</li> <li>string1 [Thursday, June 2, 2022 1:00:00 PM PDT]</li> </ul> </li> </ul>

## To dateTime Casting a date to a dateTime

### 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.

Conditions		0	1
a			
b			
Actions		<	
Post Message(s)			
A	Entity1.dateTime1 = Entity1.dateOnly1.toDateTime	<input checked="" type="checkbox"/>	
B			
Overrides			
Rule Statements			
Ref	ID	Post	Text
A0			dateTime1 is equal to the date portion of dateOnly1 plus 12 AM

## SAMPLE TEST

Input	Output
<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>dateOnly1 [April 10, 2016]</li> <li>dateTime1</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>dateOnly1 [2/3/2020]</li> <li>dateTime1</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>dateOnly1 [November 20, 1980]</li> <li>dateTime1</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>dateOnly1 [April 10, 2016]</li> <li>dateTime1 [April 10, 2016 12:00:00 AM]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>dateOnly1 [2/3/2020]</li> <li>dateTime1 [2/3/2020 12:00:00 AM]</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>dateOnly1 [November 20, 1980]</li> <li>dateTime1 [November 20, 1980 12:00:00 AM]</li> </ul> </li> </ul>

## 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`.

Conditions		0	1	
a				
b				
Actions		<		
Post Message(s)				
A	Entity1.dateTime1 = Entity1.timeOnly1.toDateTime	<input checked="" type="checkbox"/>		
B				
Overrides				
Rule Statements				
Ref	ID	Post	Alias	Text
A0				dateTime1 is equal to the time only portion of timeOnly1 plus the epoch date

## SAMPLE TEST

Input	Output
<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1</li> <li>timeOnly1 [2:00:00 PM EST]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1</li> <li>timeOnly1 [23:59:59 GMT]</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>dateTime1</li> <li>timeOnly1 [1:15:15 PM PST]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>dateTime1 [01/01/70 2:00:00 PM EST]</li> <li>timeOnly1 [2:00:00 PM EST]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>dateTime1 [01/01/70 23:59:59 GMT]</li> <li>timeOnly1 [23:59:59 GMT]</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>dateTime1 [01/01/70 1:15:15 PM PST]</li> <li>timeOnly1 [1:15:15 PM PST]</li> </ul> </li> </ul>

## 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, signed 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 `-01:45`.

Conditions		0
a		
b		

Actions		<
Post Message(s)		
A	Entity1.dateTime1 = Entity1.dateOnly1.toDateTime('-01:45')	<input checked="" type="checkbox"/>
B		
C		
D		

Ref	ID	Post	Alias	Text
A0				dateTime1 is the date converted to GMT using the timezone offset

### SAMPLE TEST

Input	Output
Entity1 [1] dateOnly1 [1/1/2020]	Entity1 [1] dateOnly1 [1/1/2020] dateTime1 [1/1/2020 3:15:00 AM]
Entity1 [2] dateOnly1 [12/31/2025]	Entity1 [2] dateOnly1 [12/31/2025] dateTime1 [12/31/2025 3:15:00 AM]
Entity1 [3] dateOnly1 [6/19/2035]	Entity1 [3] dateOnly1 [6/19/2035] dateTime1 [6/19/2035 2:15:00 AM]

## 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 'ToDecimal.ers' rulesheet editor. It features a 'Conditions' table with columns 'a' and 'b', and a '0' column. Below it is an 'Actions' table with columns 'A' and 'B', and a '0' column. The 'Post Message(s)' section contains two actions: 'A Entity1.decimal1 = Entity1.integer1.toDecimal' and 'B Entity1.decimal2 = Entity1.string1.toDecimal'. An 'Overrides' section is also present. Below the editor is a 'Rule Statements' table with columns 'Ref', 'ID', 'Post', 'Alias', and 'Text'.

Conditions		0	1
a			
b			

Actions		0
Post Message(s)		
A	Entity1.decimal1 = Entity1.integer1.toDecimal	<input checked="" type="checkbox"/>
B	Entity1.decimal2 = Entity1.string1.toDecimal	<input checked="" type="checkbox"/>

Ref	ID	Post	Alias	Text
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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>integer1 [1]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>integer1 [25]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [1]</li> </ul> </li> <li>Entity1 [4] <ul style="list-style-type: none"> <li>string1 [5.345678]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [1.000000]</li> <li>integer1 [1]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>decimal1 [25.000000]</li> <li>integer1 [25]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>decimal2 [1.000000]</li> <li>string1 [1]</li> </ul> </li> <li>Entity1 [4] <ul style="list-style-type: none"> <li>decimal2 [5.345678]</li> <li>string1 [5.345678]</li> </ul> </li> </ul>

# 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 that 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.

Conditions		0	1	
a				
b				
Actions		<		
Post Message(s)				
A	Entity1.integer1 = Entity1.decimal1.toInteger	<input checked="" type="checkbox"/>		
B	Entity1.integer2 = Entity1.string1.toInteger	<input checked="" type="checkbox"/>		
Overrides				
Rule Statements				
Ref	ID	Post	Alias	Text
A0				integer1 is equal to the value of decimal1 converted into an integer data type
B0				integer1 is equal to the value of string1 converted into a string data type

## SAMPLE TEST

Input	Output
<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [7.234000]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>decimal1 [3.999000]</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>string1 [-6]</li> </ul> </li> <li>▼ Entity1 [4] <ul style="list-style-type: none"> <li>string1 [5.0]</li> </ul> </li> <li>▼ Entity1 [5] <ul style="list-style-type: none"> <li>string1 [123A]</li> </ul> </li> <li>▼ Entity1 [6] <ul style="list-style-type: none"> <li>string1 [7]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [7.234000]</li> <li>integer1 [7]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>decimal1 [3.999000]</li> <li>integer1 [3]</li> </ul> </li> <li>▼ Entity1 [3] <ul style="list-style-type: none"> <li>integer2 [-6]</li> <li>string1 [-6]</li> </ul> </li> <li>▼ Entity1 [4] <ul style="list-style-type: none"> <li>string1 [5.0]</li> </ul> </li> <li>▼ Entity1 [5] <ul style="list-style-type: none"> <li>string1 [123A]</li> </ul> </li> <li>▼ Entity1 [6] <ul style="list-style-type: none"> <li>integer2 [7]</li> <li>string1 [7]</li> </ul> </li> </ul>

### Cases when the toInteger operator accepts null and empty values for string attributes

There are two factors:

1. Prior to evaluating a rule, Corticon checks if any attribute values used in the expressions in the rule are null and, if so, does not execute the rule.
2. During expression evaluation, Corticon protects against null pointer exceptions. The expression "test.string.toInteger" will return null if the string is not an integer. However, the expression "test.string.toInteger + 3" will return "3" if the string is not a number – the value 0 being used as the result of the toInteger.

Consider the action expression:

```
test.integer =test.string.toInteger
```

Here is the Ruletest output for three tests:

Input	Output
test [1] integer [5] string [] test [2] integer [5] string test [3] integer [5] string [null]	test [1] integer string [] test [2] integer [5] string test [3] integer string [null]

How this Ruletest was processed:

- In test 1, the string is empty but not a null value so the expression evaluates and assigns null to integer.
- In test 2, the string is null so the pre-check for null values does not pass and the expression is not evaluated and the value of integer is unchanged
- In test 3, the string is the string "null" but not a null value so the expression evaluates and assigns null to integer. (Note the value "null" here is a string, it could have just as well been "foo").

Now change the action expression to:

```
test.integer =test.string.toInteger + 3
```

Here is the Ruletest output now:

Input	Output
test [1] integer [5] string [] test [2] integer [5] string test [3] integer [5] string [null]	test [1] integer [3] string [] test [2] integer [5] string test [3] integer [3] string [null]

How this Ruletest was processed now:

- In test 1, the string is empty but not a null value so the expression evaluates. To prevent a NPE during evaluation, the value 0 is used as the result of the toInteger resulting in the expression being "0 + 3" so integer is assigned a value of 3.
- In test 2, the string is null so the pre-check for null values does not pass and the expression is not evaluated and the value of integer is unchanged.

- In test 3, the string is the string “null” but not a null value so the expression evaluates in the same fashion as 1, that is, “0 + 3” and assigns a value of 3.

You might argue that you cannot assume a value of 0 when doing toString on a non-number string. However, to protect a business user against runtime exceptions, Corticon makes logical substitutions during rule evaluation to protect against null values.

## 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.

To string.ers				
Conditions				0
a				
b				
Actions				<
Post Message(s)				
A			Entity1.string1 = Entity1.decimal1.toString	<input checked="" type="checkbox"/>
B			Entity1.string2 = Entity1.integer1.toString	<input checked="" type="checkbox"/>
C			Entity2.string1 = Entity2.dateTime1.toString + ' AD'	<input checked="" type="checkbox"/>
Overrides				
Rule Statements				
Ref	ID	Post	Alias	Text
A0				Entity1.string1 is equal to the value of decimal1 converted into a string data type
B0				Entity1.string2 is equal to the value of integer1 converted into a string data type
C0				Entity2.string1 is equal to the value of dateTime1 converted into a string data type and appended with AD.

## SAMPLE TEST

Input	Output
<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [3.456700]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>integer1 [5]</li> </ul> </li> <li>▼ Entity2 [1] <ul style="list-style-type: none"> <li>dateTime1 [3/16/2026 2:00:00 PM EST]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1] <ul style="list-style-type: none"> <li>decimal1 [3.456700]</li> <li>string1 [3.456700]</li> </ul> </li> <li>▼ Entity1 [2] <ul style="list-style-type: none"> <li>integer1 [5]</li> <li>string2 [5]</li> </ul> </li> <li>▼ Entity2 [1] <ul style="list-style-type: none"> <li>dateTime1 [3/16/2026 2:00:00 PM EST]</li> <li>string1 [3/16/2026 2:00:00 PM EST AD]</li> </ul> </li> </ul>

## 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.

### 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`.

Conditions		0	1	
a				
b				
Actions		<		
Post Message(s)				
A	Entity1.timeOnly1 = Entity1.dateTime1.toTime	<input checked="" type="checkbox"/>		
B				
Overrides				
Rule Statements				
Ref	ID	Post	Alias	Text
A0				timeOnly1 is equal to the time portion of dateTime1

## SAMPLE TEST

Input	Output
<ul style="list-style-type: none"> <li>▼ Entity1 [1]           <ul style="list-style-type: none"> <li>dateTime1 [2/2/2027 3:10:12 AM EST]</li> <li>timeOnly1</li> </ul> </li> <li>▼ Entity1 [2]           <ul style="list-style-type: none"> <li>dateTime1 [April 10, 2024 2:00:00 PM EST]</li> <li>timeOnly1</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▼ Entity1 [1]           <ul style="list-style-type: none"> <li>dateTime1 [2/2/2027 3:10:12 AM EST]</li> <li><b>timeOnly1 [3:10:12 AM EST]</b></li> </ul> </li> <li>▼ Entity1 [2]           <ul style="list-style-type: none"> <li>dateTime1 [April 10, 2024 2:00:00 PM EST]</li> <li><b>timeOnly1 [2:00:00 PM EST]</b></li> </ul> </li> </ul>

## 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`.

Ref	ID	Post	Alias	Text
A0				The value of boolean1 in collection1 with the lowest decimal1 must be true

## SAMPLE TEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [2.500000]</li> <li>integer1 [9]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [5.800000]</li> <li>integer1 [5]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [3.300000]</li> <li>integer1 [7]</li> </ul> </li> <li>entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>decimal1 [7.400000]</li> <li>integer1 [2]</li> </ul> </li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>entity2 (Entity2) [5] <ul style="list-style-type: none"> <li>decimal1 [3.000000]</li> <li>integer1 [4]</li> </ul> </li> <li>entity2 (Entity2) [6] <ul style="list-style-type: none"> <li>decimal1 [2.000000]</li> <li>integer1 [4]</li> </ul> </li> <li>entity2 (Entity2) [7] <ul style="list-style-type: none"> <li>decimal1 [4.000000]</li> <li>integer1 [4]</li> </ul> </li> <li>entity2 (Entity2) [8] <ul style="list-style-type: none"> <li>decimal1 [8.000000]</li> <li>integer1 [4]</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [DECR]</li> <li>entity2 (Entity2) [1] <ul style="list-style-type: none"> <li>decimal1 [2.500000]</li> <li>integer1 [9]</li> </ul> </li> <li>entity2 (Entity2) [2] <ul style="list-style-type: none"> <li>decimal1 [5.800000]</li> <li>integer1 [5]</li> </ul> </li> <li>entity2 (Entity2) [3] <ul style="list-style-type: none"> <li>decimal1 [3.300000]</li> <li>integer1 [7]</li> </ul> </li> <li>entity2 (Entity2) [4] <ul style="list-style-type: none"> <li>decimal1 [7.400000]</li> <li>integer1 [2]</li> </ul> </li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [CNST]</li> <li>entity2 (Entity2) [5] <ul style="list-style-type: none"> <li>decimal1 [3.000000]</li> <li>integer1 [4]</li> </ul> </li> <li>entity2 (Entity2) [6] <ul style="list-style-type: none"> <li>decimal1 [2.000000]</li> <li>integer1 [4]</li> </ul> </li> <li>entity2 (Entity2) [7] <ul style="list-style-type: none"> <li>decimal1 [4.000000]</li> <li>integer1 [4]</li> </ul> </li> <li>entity2 (Entity2) [8] <ul style="list-style-type: none"> <li>decimal1 [8.000000]</li> <li>integer1 [4]</li> </ul> </li> </ul> </li> </ul>

**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.

## Trim spaces

### SYNTAX

<String>.trimSpaces

### DESCRIPTION

Returns <String>.

### 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 `trimSpaces`.

Ref	ID	Post	Alias	Text
A0				Entity1.string1 is set to the value of Entity1.string2 without extra spaces

## SAMPLE RULETEST

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

**Note:** As the Studio Tester trims spaces in the input area, you cannot really test this operation here!

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [test]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [test test]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [test test]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [test]</li> <li>string2 [test]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [test test]</li> <li>string2 [test test]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [test test]</li> <li>string2 [test test]</li> </ul> </li> </ul>

# 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.

The screenshot shows the True.ers rulesheet editor. The left pane shows the Scope tree with Entity1 containing Filters (boolean1, decimal1, decimal2, integer1). The right pane shows the Conditions table with rows a, b, and c, and the Actions table with a Post Message(s) section containing Action A: Entity1.decimal1 = Entity1.decimal2 + Entity1.integer1. The Filters section shows a filter 1: Entity1.boolean1 = true. The Rule Statements table at the bottom shows a single rule A0 with the text: If boolean1 is true, then decimal1 equals the sum of decimal2 plus integer1.

Ref	ID	Post	Alias	Text
A0				If boolean1 is true, then decimal1 equals the sum of decimal2 plus integer1

## 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:

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li>decimal2 [10.000000]</li> <li>integer1 [5]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>decimal2 [10.000000]</li> <li>integer1 [5]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>boolean1 [true]</li> <li><b>decimal1 [15.000000]</b></li> <li>decimal2 [10.000000]</li> <li>integer1 [5]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>boolean1 [false]</li> <li>decimal2 [10.000000]</li> <li>integer1 [5]</li> </ul> </li> </ul>

# Uppercase

## SYNTAX

<String>.toUpper

## 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 **.toUpper** to convert *string2* to uppercase and assign it to *string1*.

The screenshot shows the 'Uppercase.ers' rulesheet editor. It features a table for conditions and actions, and a 'Rule Statements' section below.

Conditions		0	1
a			
b			

Actions		<
Post Message(s)		
A	Entity1.string1 = Entity1.string2.toUpper	<input checked="" type="checkbox"/>
B		

Rule Statements				
Ref	ID	Post	Alias	Text
A0				string1 equals string2 converted to uppercase

## SAMPLE TEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string2 [uppercase]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string2 [CaliForniA]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string2 [TNT]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1] <ul style="list-style-type: none"> <li>string1 [UPPERCASE]</li> <li>string2 [uppercase]</li> </ul> </li> <li>Entity1 [2] <ul style="list-style-type: none"> <li>string1 [CALIFORNIA]</li> <li>string2 [CaliForniA]</li> </ul> </li> <li>Entity1 [3] <ul style="list-style-type: none"> <li>string1 [TNT]</li> <li>string2 [TNT]</li> </ul> </li> </ul>

# 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`.

The screenshot shows a Rulesheet editor window titled "WeekofMonth.ers". It is divided into two main sections: "Conditions" and "Rule Statements".

**Conditions Section:**

Conditions		0
a		
b		

**Actions Section:**

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

**Rule Statements Section:**

Ref	ID	Post	Alias	Text
A0				integer1 is equal to the integer number of the week of the month in dateTime1

## SAMPLE TEST

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]               <ul style="list-style-type: none"> <li>dateTime1 [2/1/2024 12:00:00 PM]</li> </ul> </li> <li>Entity1 [2]               <ul style="list-style-type: none"> <li>dateTime1 [4/30/2024 1:30:00 PM]</li> </ul> </li> <li>Entity1 [3]               <ul style="list-style-type: none"> <li>dateTime1 [9/30/2026 4:00:00 AM]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]               <ul style="list-style-type: none"> <li>dateTime1 [2/1/2024 12:00:00 PM]</li> <li>integer1 [1]</li> </ul> </li> <li>Entity1 [2]               <ul style="list-style-type: none"> <li>dateTime1 [4/30/2024 1:30:00 PM]</li> <li>integer1 [5]</li> </ul> </li> <li>Entity1 [3]               <ul style="list-style-type: none"> <li>dateTime1 [9/30/2026 4:00:00 AM]</li> <li>integer1 [5]</li> </ul> </li> </ul>

# 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`.

The screenshot shows a rulesheet editor with the following sections:

- Conditions:** A table with columns 'a' and 'b'. The value '0' is entered in the 'b' column.
- Actions:** A table with columns 'A' and 'B'. The action 'Entity1.integer1 = Entity1.dateTime1.weekOfYear' is entered in the 'A' column, with a green checkmark in the 'B' column.
- Rule Statements:** A table with columns 'Ref', 'ID', 'Post', 'Alias', and 'Text'. The statement 'integer1 is equal to the integer number of the week of the year in dateTime1' is listed with 'A0' in the 'Ref' column.

## SAMPLE TEST

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>dateTime1 [12/30/2023 2:00:00 PM]</li> </ul> </li> <li>Entity1 [2]                             <ul style="list-style-type: none"> <li>dateTime1 [8/25/2024 11:45:00 AM]</li> </ul> </li> <li>Entity1 [3]                             <ul style="list-style-type: none"> <li>dateTime1 [3/16/2026 10:30:00 PM]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>dateTime1 [12/30/2023 2:00:00 PM]</li> <li>integer1 [52]</li> </ul> </li> <li>Entity1 [2]                             <ul style="list-style-type: none"> <li>dateTime1 [8/25/2024 11:45:00 AM]</li> <li>integer1 [35]</li> </ul> </li> <li>Entity1 [3]                             <ul style="list-style-type: none"> <li>dateTime1 [3/16/2026 10:30:00 PM]</li> <li>integer1 [12]</li> </ul> </li> </ul>

# 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.

The screenshot shows a rulesheet editor for 'year.ers'. It is divided into two main sections: 'Conditions' and 'Actions'.

**Conditions:**

	0	1
a		
b		

**Actions:**

Post Message(s)	0	1
A Entity1.integer1 = Entity1.dateTime1.year	<input checked="" type="checkbox"/>	
B Entity1.integer2 = Entity1.dateOnly1.year	<input checked="" type="checkbox"/>	

**Rule Statements:**

Ref	ID	Post	Alias	Text
A0				integer1 equals the year value in dateTime1
B0				integer2 equals the year value in dateOnly1

### SAMPLE TEST

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

Input	Output
<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>dateOnly1 [2/1/2020]</li> <li>dateTime1 [3/16/2026 3:00:00 PM EST]</li> </ul> </li> <li>Entity1 [2]                             <ul style="list-style-type: none"> <li>dateOnly1 [May 14, 2019]</li> <li>dateTime1 [June 20, 2006 2:00:00 AM PST]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Entity1 [1]                             <ul style="list-style-type: none"> <li>dateOnly1 [2/1/2020]</li> <li>dateTime1 [3/16/2026 3:00:00 PM EST]</li> <li>integer1 [2026]</li> <li>integer2 [2020]</li> </ul> </li> <li>Entity1 [2]                             <ul style="list-style-type: none"> <li>dateOnly1 [May 14, 2019]</li> <li>dateTime1 [June 20, 2006 2:00:00 AM PST]</li> <li>integer1 [2006]</li> <li>integer2 [2019]</li> </ul> </li> </ul>

# 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`.

YearsBetween.ers				
Conditions		1	2	
a	Entity1.dateTime1.yearsBetween(Entity1.dateTime2)	<= 3	> 3	
b				
Actions		<		
Post Message(s)				
A	Entity1.string1	'Not Overdue'	'Overdue'	
Overrides				
Rule Statements				
Ref	ID	Post	Alias	Text
1				If 3 or fewer years have elapsed between <code>dateTime1</code> and <code>dateTime2</code> , then Entity1 is not overdue
2				If more than 3 years have elapsed between <code>dateTime1</code> and <code>dateTime2</code> , then Entity1 is overdue

## SAMPLE TEST

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

Input	Output
<ul style="list-style-type: none"> <li> <span style="font-size: 1em;">▼</span> <span style="font-size: 1em;">☰</span> Entity1 [1]           <ul style="list-style-type: none"> <li><span style="font-size: 0.8em;">☰</span> dateTime1 [May 9, 2019 2:30:00 PM EST]</li> <li><span style="font-size: 0.8em;">☰</span> dateTime2 [February 5, 2017 5:30:00 PM EST]</li> </ul> </li> <li> <span style="font-size: 1em;">▼</span> <span style="font-size: 1em;">☰</span> Entity1 [2]           <ul style="list-style-type: none"> <li><span style="font-size: 0.8em;">☰</span> dateTime1 [3/10/1992 2:00:00 PM PST]</li> <li><span style="font-size: 0.8em;">☰</span> dateTime2 [7/1/2025 11:30:00 AM PST]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li> <span style="font-size: 1em;">▼</span> <span style="font-size: 1em;">☰</span> <b>Entity1 [1]</b> <ul style="list-style-type: none"> <li><span style="font-size: 0.8em;">☰</span> dateTime1 [May 9, 2019 2:30:00 PM EST]</li> <li><span style="font-size: 0.8em;">☰</span> dateTime2 [February 5, 2017 5:30:00 PM EST]</li> <li><span style="font-size: 0.8em;">☰</span> <b>string1 [Not Overdue]</b></li> </ul> </li> <li> <span style="font-size: 1em;">▼</span> <span style="font-size: 1em;">☰</span> <b>Entity1 [2]</b> <ul style="list-style-type: none"> <li><span style="font-size: 0.8em;">☰</span> dateTime1 [3/10/1992 2:00:00 PM PST]</li> <li><span style="font-size: 0.8em;">☰</span> dateTime2 [7/1/2025 11:30:00 AM PST]</li> <li><span style="font-size: 0.8em;">☰</span> <b>string1 [Overdue]</b></li> </ul> </li> </ul>

---

# A

---

## Standard Boolean constructions

---

The topics in this section presents several standard truth tables (AND, NAND, OR, XOR, NOR, and XNOR) with examples of usage in a Rulesheet.

For details, see the following topics:

- [Boolean AND](#)
- [Boolean NAND](#)
- [Boolean OR](#)
- [Boolean XOR](#)
- [Boolean NOR](#)
- [Boolean XNOR](#)

### Boolean AND

In a decision table, a rule with **AND**'ed Conditions is expressed as a single column, with values for each Condition aligned vertically in that column. For example:

1. If a person is 45 or older and smokes, then classify the person as high risk

Conditions		0	1
a	Applicant.age >= 45		T
b	Applicant.smoker		T
Actions		<	
Post Message(s)			
A	Applicant.riskRating		'high'
B			
C			
Overrides			

Ref	ID	Post	Alias	Text
1				If an applicant is 45 or older and smokes, then classify the applicant as high risk

In this scenario, each Condition has a set of 2 possible values:

person is 45 or older: {true, false}

person is a smoker: {true, false}

and the outcome may also have two possible values:

person's risk rating: {low, high}

These Conditions and Actions yield the following truth table:

age >= 45	smoker	risk rating
true	true	high
true	false	
false	true	
false	false	

Note that we have only filled in a single value of risk rating, because the business rule above only covers a single scenario: where `age >= 45` and `smoker = true`. Running [The completeness checker](#) *The completeness checker* as described in the *Rule Modeling* section quickly identifies the remaining three scenarios:

PolicyApplicantBoolean.ers				
Conditions		1	2	3
a	Applicant.age >= 45	T	T	F
b	Applicant.smoker	T	{F, null}	-
c				
Actions		<		
Post Message(s)				
A	Applicant.riskRating	'high'		
B				
Overrides				
Rule Statements				
Ref	ID	Post	Alias	Text
1				If an applicant is 45 or older and smokes, then classify the applicant as high risk

Completing the truth table and the Rulesheet requires the definition of 2 additional business rules:

PolicyApplicantBoolean.ers				
Conditions		1	2	3
a	Applicant.age >= 45	T	T	F
b	Applicant.smoker	T	{F, null}	-
Actions		<		
Post Message(s)				
A	Applicant.riskRating	'high'	'low'	'low'
B				
C				
Overrides				
Rule Statements				
Ref	ID	Post	Alias	Text
1				If an applicant is 45 or older and smokes, then classify the applicant as high risk
2				If an applicant is 45 or older and does NOT smoke, then classify the applicant as low risk
3				If an applicant is NOT 45 or older, then ignore whether or not the applicant smokes and classify them as low risk

and updating the truth table, we recognize the classic **AND** Boolean function.

age >= 45	smoker	risk rating
true	true	high
true	false	low
false	true	low
false	false	low

Once the basic truth table framework has been established in the Rulesheet by the Completeness Checker – in other words, all logical combinations of Conditions have been explicitly entered as separate columns in the Rulesheet – we can alter the outcomes to implement other standard Boolean constructions. For example, the **NAND** construction has the following truth table:

## Boolean NAND

age >= 45	smoker	risk rating
true	true	low
true	false	high
false	true	high
false	false	high

Also known as “Not And”, this construction is shown in the following Rulesheet:

Conditions		0	1	2	3	4
a	Applicant.age >= 45		T	T	F	F
b	Applicant.smoker		T	F	T	F
Actions		<				
Post Message(s)						
A	Applicant.riskRating		'low'	'high'	'high'	'high'
B						
C						
D						
Overrides						

Ref	ID	Post	Alias	Text
1				If an applicant is 45 or older AND smokes, then classify the applicant as low risk
2				If an applicant is 45 or older AND does NOT smoke, then classify the applicant as high risk
3				If an applicant is younger than 45 AND smokes, then classify the applicant as high risk
4				If an applicant is younger than 45 AND does NOT smoke, then classify the applicant as high risk

## Boolean OR

age >= 45	smoker	risk rating
true	true	high
true	false	high
false	true	high
false	false	low

PolicyApplicantBooleanOr.ers					
Conditions	0	1	2	3	4
a Applicant.age >= 45		T	T	F	F
b Applicant.smoker		T	F	T	F
c					
Actions	<				
Post Message(s)					
A Applicant.riskRating		'high'	'high'	'high'	'low'
B					
Overrides					

Rule Statements				
Ref	ID	Post	Alias	Text
1				If an applicant is 45 or older and smokes, then classify the applicant as high risk
2				If an applicant is 45 or older and does NOT smoke, then classify the applicant as high risk
3				If an applicant is younger than 45 and smokes, then classify the applicant as high risk
4				If an applicant is younger than 45 and does NOT smoke, then classify the applicant as low risk

## Boolean XOR

Using “Exclusive Or” logic, `riskRating` is high whenever the age or smoker test, but not both, is satisfied. This construction is shown in the following Rulesheet:

age >= 45	smoker	risk rating
true	true	low
true	false	high
false	true	high
false	false	low

XOR.ers					
Conditions	0	1	2	3	4
a Applicant.age >= 45		T	T	F	F
b Applicant.smoker		T	F	T	F
c					
Actions	<				
Post Message(s)					
A Applicant.riskRating		'low'	'high'	'high'	'low'
B					
Overrides					
Rule Statements					
Ref	ID	Post	Alias	Text	
1				If an applicant is 45 or older AND smokes, then classify the applicant as low risk	
2				If an applicant is 45 or older AND does NOT smoke, then classify the applicant as high risk	
3				If an applicant is younger than 45 AND smokes, then classify the applicant as high risk	
4				If an applicant is younger than 45 AND does NOT smoke, then classify the applicant as low risk	

## Boolean NOR

Also known as “Not Or”, this construction is shown in the following Rulesheet:

age >= 45	smoker	risk rating
true	true	low
true	false	low
false	true	low
false	false	high

PolicyApplicantBooleanNOR.ers

Conditions	0	1	2	3	4
a Applicant.age >= 45		T	T	F	F
b Applicant.smoker		T	F	T	F
c					
Actions	<				
Post Message(s)					
A Applicant.riskRating		'low'	'low'	'low'	'high'
B					
Overrides					

Rule Statements

Ref	ID	Post	Alias	Text
1				If an applicant is 45 or older and smokes, then classify the applicant as low risk
2				If an applicant is younger than 45 AND does NOT, then classify the applicant as low risk
3				If an applicant is younger than 45 AND smokes, then classify the applicant as low risk
4				If an applicant is younger than 45 AND does NOT smoke, then classify the applicant as high risk

## Boolean XNOR

Also known as “Exclusive NOR”, this construction is shown in the following Rulesheet:

age >= 45	smoker	risk rating
true	true	high
true	false	low
false	true	low
false	false	high

PolicyApplicantBooleanXNOR.ers						
Conditions		0	1	2	3	4
a	Applicant.age <= 45		F	T	F	T
b	Applicant.smoker		F	T	T	F
c						
Actions		<				
Post Message(s)						
A	Applicant.riskRating		'high'	'high'	'low'	'low'
B						
C						
Overrides						
Rule Statements						
Ref	ID	Post	Alias	Text		
1				If an applicant is 45 or older AND does NOT smoke, then classify the applicant as high risk		
2				If an applicant is younger than 45 AND smokes, then classify the applicant as high risk		
3				If an applicant is 45 or older AND smokes, then classify the applicant as low risk		
4				If an applicant is younger than 45 AND does NOT smoke, then classify the applicant as low risk		

---

# B

## Character precedence in 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 `false`.
- `'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

character	name	precedence	Unicode 5.0 code
-	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
#	number sign or hash sign	27	0023

<b>character</b>	<b>name</b>	<b>precedence</b>	<b>Unicode 5.0 code</b>
%	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
s, S	letter s, small and capital	62	0073, 0053

<b>character</b>	<b>name</b>	<b>precedence</b>	<b>Unicode 5.0 code</b>
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

---

**C**

# 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	( <i>expression</i> , <i>expression</i> )	Logical: AND	(>5.5,<10)
	( <i>expression</i> or <i>expression</i> )	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.

---



---

## Formats for Date Time and DateTime properties

---

DateTime information may take many different formats. Corticon uses 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. `'2019-04-12'` conforms to this mask; `'April 12th, 2019'` 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 `brms.properties` file) are also present in the Corticon Server's `brms.properties` file.

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.

## Setting and modifying masks

Date/time masks are stored as a set of defaults that can be replaced by listing preferred values in the `brms.properties` file located at your work directory root – or, in Studio, the preferred location specified in **Preferences**. Corticon Studio's `DateTime` datatype uses both date and time data. The `Date` datatype handles only date information, and the `Time` datatype handles only time information.

The Corticon XML Translator will maintain the consistency of `DateTime`, `Date`, and `Time` values from input to output documents as long as the masks that are used are contained in the lists.

---

**Note:** Property settings you list in your `brms.properties` do not *append* to an existing list, they *replace* the default values. For example, if you want to add a new `DateTime` mask to the built-in list, be sure to include all the masks you intend to use, not just the new one. If your `brms.properties` file contains only the new mask, then it will be the only mask Corticon uses.

---

There is only one `Date` datatype. It handles dates, times, and date/times. A `Date` attribute is designated as date, time, or date/time depending on which of the masks below are matched. This designation changes the behavior of `Date` comparison operators.

The `dateformat`, `timeformat`, and `datetimeformat`, `Date` masks process incoming date/times on request XML payloads, insert date/times into output response XML payloads, parse entries made in the Studio Rulesheets, Vocabulary, and Testsheets, and to display any date/time in Studio.

The first entry for each `dateformat`, `datetimeformat`, and `timeformat` is the default mask. For example, the built-in operator `today` always returns the current date in the default `dateformat` mask.

The function `now` returns the current date in the default `datetimeformat`. The entries can be altered but must conform to the patterns/masks supported by the Java class `SimpleDateFormat` in the `java.text` package.

```
com.corticon.crml.OclDate.dateformat=
MM/dd/yy
MM/dd/yyyy
M/d/yy
M/d/yyyy
yyyy/MM/dd
yyyy-MM-dd
yyyy/M/d
yy/MM/dd
yy/M/d
MMM d, yyyy
MMMMM d, yyyy
```

```
com.corticon.crml.OclDate.datetimeformat=
MM/dd/yy h:mm:ss a
MM/dd/yyyy h:mm:ss a
M/d/yy h:mm:ss a
M/d/yyyy h:mm:ss a
yyyy/MM/dd h:mm:ss a
yyyy/M/d h:mm:ss a
yy/MM/dd h:mm:ss a
yy/M/d h:mm:ss a
MMM d, yyyy h:mm:ss a
MMMMM d, yyyy h:mm:ss a
MM/dd/yy H:mm:ss
MM/dd/yyyy H:mm:ss
M/d/yy H:mm:ss
M/d/yyyy H:mm:ss
yyyy/MM/dd H:mm:ss
yyyy/M/d H:mm:ss
yy/MM/dd H:mm:ss
yy/M/d H:mm:ss
MMM d, yyyy H:mm:ss
MMMMM d, yyyy H:mm:ss
```

```

MM/dd/yy hh:mm:ss a
MM/dd/yyyy hh:mm:ss a
M/d/yy hh:mm:ss a
M/d/yyyy hh:mm:ss a
yyyy/MM/dd hh:mm:ss a
yyyy/M/d hh:mm:ss a
yy/MM/dd hh:mm:ss a
yy/M/d hh:mm:ss a
MMM d, yyyy hh:mm:ss a
MMMMM d, yyyy hh:mm:ss a
MM/dd/yy HH:mm:ss
MM/dd/yyyy HH:mm:ss
M/d/yy HH:mm:ss
M/d/yyyy HH:mm:ss
yyyy/MM/dd HH:mm:ss
yyyy/M/d HH:mm:ss
yy/MM/dd HH:mm:ss
yy/M/d HH:mm:ss
MMM d, yyyy HH:mm:ss
MMMMM d, yyyy HH:mm:ss
MM/dd/yy h:mm:ss a z
MM/dd/yyyy h:mm:ss a z
M/d/yy h:mm:ss a z
M/d/yyyy h:mm:ss a z
yyyy/MM/dd h:mm:ss a z
yyyy/M/d h:mm:ss a z
yy/MM/dd h:mm:ss a z
yy/M/d h:mm:ss a z
MMM d, yyyy h:mm:ss a z
MMMMM d, yyyy h:mm:ss a z
MM/dd/yy H:mm:ss z
MM/dd/yyyy H:mm:ss z
M/d/yy H:mm:ss z
M/d/yyyy H:mm:ss z
yyyy/MM/dd H:mm:ss z
yyyy/M/d H:mm:ss z
yy/MM/dd H:mm:ss z
yy/M/d H:mm:ss z
MMM d, yyyy H:mm:ss z
MMMMM d, yyyy H:mm:ss z
MM/dd/yy hh:mm:ss a z
MM/dd/yyyy hh:mm:ss a z
M/d/yy hh:mm:ss a z
M/d/yyyy hh:mm:ss a z
yyyy/MM/dd hh:mm:ss a z
yyyy/M/d hh:mm:ss a z
yy/MM/dd hh:mm:ss a z
yy/M/d hh:mm:ss a z
MMM d, yyyy hh:mm:ss a z
MMMMM d, yyyy hh:mm:ss a z
MM/dd/yy HH:mm:ss z
MM/dd/yyyy HH:mm:ss z
M/d/yy HH:mm:ss z
M/d/yyyy HH:mm:ss z
yyyy/MM/dd HH:mm:ss z
yyyy/M/d HH:mm:ss z
yy/MM/dd HH:mm:ss z
yy/M/d HH:mm:ss z
MMM d, yyyy HH:mm:ss z
MMMMM d, yyyy HH:mm:ss z

```

```

com.corticon.crml.OclDate.timeformat=
h:mm:ss a
h:mm:ss a z
H:mm:ss
H:mm:ss z
hh:mm:ss a
hh:mm:ss a z

```

```
HH:mm:ss
HH:mm:ss z
```

-----

When `com.corticon.crml.OclDate.locale=true`, it will override the default datetime mask and use the locale mask as the date style type defined by `com.corticon.crml.OclDate.datetype` and the time style type defined by `com.corticon.crml.OclDate.timetypevalue` for `datetype` and `timetype` are defined as values of `java.text.DateFormat` enums: FULL = 0, LONG = 1, MEDIUM = 2, SHORT = 3 .

```
com.corticon.crml.OclDate.locale=false
com.corticon.crml.OclDate.datetype=3
com.corticon.crml.OclDate.timetype=2
```

-----

If `permissive` is true (default), then the Corticon date/time parser will be lenient when handling incoming or entered date/times, trying to find a match even if the pattern is not contained in the mask lists. If false, then any incoming or entered date/time must strictly adhere to the patterns defined by `dateformat`, `datetimeformat`, `timeformat`.

Default patterns are for United States and other countries that follow the US conventions on date/times.

```
com.corticon.crml.OclDate.permissive =true
```

-----

By default, when the value of `now` is pinned, the milliseconds are set to zero. This property can specify how to deal with the nano seconds (which can affect the milliseconds).

- Value of `ZERO_MILLIS` sets the nanos to 0 (which also sets milliseconds to zero)
- Value of `ZERO_NANOS` sets only the last 3 digits of the nanos to zero (which does not modify millis)
- Value of `NO_ZERO` does not modify the nanos (This has shown some rare side effects where datetime appears equal however the hidden nanos values cause comparison to be not equal)

Default value is `ZERO_MILLIS`

```
com.corticon.crml.OclDate.nanos=ZERO_MILLIS
```

-----

If `maskliterals` is true (default), the system will parse strings and dates more quickly by checking for the presence of mask literals (for example, “/”, “-”, “:” or “.”) before consulting the date masks (an expensive process). If a string does not contain any of the mask literal characters, it can be immediately deemed a string (as opposed to a date).

```
com.corticon.crml.OclDate.maskliterals =true
```

-----

When a Date literal contains time zone information, it may cause a shift in the actual day because internally Corticon expects Date literals to have a time component of 00:00:00 GMT. By setting `com.corticon.crml.OclDate.ignoreTimeZoneOnDate=true`, the time component is ignored when converting a string into a Date literal. The result is an internal value that has a time component of 00:00:00 GMT which normalizes the Date literal. Default value is `false`.

```
com.corticon.crml.OclDate.ignoreTimeZoneOnDate =true
```

## Mask patterns

To take advantage of this feature, all user-specified date masks must contain at least one literal character. If any user-specified masks contain exclusively date pattern characters (for example, 'MMddYY'), *maskliterals* must be set to false in order to prevent the system from misinterpreting date literals (for example, '123199') as simple strings.

These properties deal with the way Corticon Studio and Corticon Server handle date/time formats. Preset formats -- referred to as *masks* - are used to:

- Process incoming date/times on request XML payloads.
- Insert date/times into output response XML payloads.
- Parse entries made in the Corticon Studio Rulesheets, Vocabulary, and Tests.
- To display any date/time in Corticon Studio.

Masks are divided into 3 categories: *dateformat*, *datetimeformat*, *timeformat*.

Use the following chart to decode the date mask formats:

The following symbols are used in date/time masks:

Symbol	Meaning	Presentation	Patterns
G	Era designator	Text	G = {AD, BC}
Y	Year	Number	yy = {00..99} yyyy = {0000..9999}
Y	Week year	Number	YY = {00..99} YYYY = {0000..9999}
M	Month in year	Text or Number	M = {1..12} MM = {01..12} MMM = {Jan..Dec} MMMM = {January..December}
w	Week in year	Number	w = {1..53} ww = {01..53}
W	Week in month	Number	W = {1..6}
D	Day in year	Number	D = {0..366} DDD = {000..366}
d	Day in month	Number	d = {1..31} dd = {01..31}
F	Day of week in month	Number	F = {0..6}

Symbol	Meaning	Presentation	Patterns
E	Day name in week	Text	E, EE, or EEE = {Sun..Sat} EEEE = {Sunday..Saturday}
u	Day number of week (1 = Monday, ..., 7 = Sunday)	Number	u = {1..7}
a	AM/PM marker	Text	a = {AM, PM}
H	Hour in 24-hour format (0-23)	Number	H = {0..23} HH = {00..23}
k	Hour in day (1-24)	Number	k = {1..24} kk = {01..24}
K	Hour in AM/PM (0-11)	Number	K = {1..12} KK = {01..12}
h	Hour in AM or PM	Number	h = {1..12} hh = {01..12}
m	Minute in hour	Number	m = {0..59} mm = {00..59}
s	Second in minute	Number	s = {0..59} ss = {00..59}
S	Millisecond in minute	Number	S = {0..999} SSS = {000..999}
z	General time zone	Text	z, zz, or zzz = abbreviated time zone zzzz = full time zone
Z	RFC 822 time zone	Text	Z,ZZ, or ZZZ = abbreviated time zone ZZZZ = full time zone
X	ISO 8601 time zone	Text	X, XX, or XXX = abbreviated time zone XXXX = full time zone
`	escape character used to insert text	Delimiter	
'	single quote	Literal	'

---

Any characters in the pattern that are not in the ranges of [a..z] and [A..Z] will be treated as quoted text. For instance, characters like {:, ., <space>, #, @} will appear in the resulting time text even they are not embraced within single quotes. A pattern containing any invalid pattern letter will result in a thrown exception during formatting or parsing.

Examples:

Sample Pattern	Resulting Formatted Date
yyyy.MM.dd G 'at' hh:mm:ss z	2013.07.10 AD at 15:08:56 PDT
EEE, MMM d, ''yy	Wed, Jul 10, '13
h:mm a	12:08 PM
hh 'o''clock' a, zzzz	12 o'clock PM, Pacific Daylight Time
K:mm a, z	0:00 PM, PST
yyyy.MMMM.dd G h:mm a	2013.July.10 AD 12:08 PM

---

**Note:** See [SimpleDateFormat Javadocs](#) for more detailed information.

---

