

# webMethods Business Rules Reference

Version 10.1

October 2017

This document applies to webMethods Business Rules Version 10.1 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2015-2017 Software AG, Darmstadt, Germany and/or Software AG USA Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://softwareag.com/licenses>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

# Table of Contents

<b>About this Guide.....</b>	<b>7</b>
Document Conventions.....	7
Online Information.....	8
<b>Functions Overview.....</b>	<b>11</b>
Summary of Conversion Functions.....	15
toBoolean().....	16
toDate().....	16
toDate().....	17
toDate(String dateFormat).....	17
toDouble().....	18
toDouble().....	18
toLong().....	18
toLong().....	19
toString().....	19
toString().....	19
toString().....	20
toString().....	20
toString(String dateFormat).....	21
Summary of Date Functions.....	21
int century().....	23
int compareDates(Date date1, Date date2).....	24
Date date().....	24
Date dateMinusDays(int days).....	25
Date datePlusDays(int days).....	25
long dateToInt().....	25
int dayOfMonth().....	26
int dayOfWeek().....	26
int dayOfYear().....	26
int daysInMonth(int month, int year).....	27
long diffInDays(Date anotherDate).....	27
long diffInMonths(Date anotherDate).....	28
long diffInYears(Date anotherDate).....	28
Date intToDate(long millis).....	28
isLeapYear(int year).....	29
int month().....	29
String time().....	30
boolean verifyDate(int year, int month, int day).....	30
boolean verifyIntDate(long millis).....	30
boolean verifyMonth(int month).....	31
boolean verifyYear(int year).....	31

int year()	32
Date ymdToDate(int year, int month, int day)	32
Summary of List and Range Functions	32
boolean inList(String[] list)	35
boolean inList(String[] list, boolean ignoreCase)	36
boolean inRange(Date lowerBound, Date upperBound)	36
boolean inRange(Date lowerBound, Date upperBound, Date[] exclusions)	37
boolean inRange(Date[][] listOfRanges, Date[] exclusions)	37
boolean inRange(Date[][] listOfRanges, boolean inclusiveLower, boolean inclusiveUpper, Date[] exclusions)	38
boolean inRange(Double lowerBound, Double upperBound)	39
boolean inRange(Double lowerBound, Double upperBound, Double[] exclusions)	39
boolean inRange(Double[][] listOfRanges, Double[] exclusions)	40
boolean inRange(Double[][] listOfRanges, boolean inclusiveLower, boolean inclusiveUpper, Double[] exclusions)	41
boolean inRange(Long lowerBound, Long upperBound)	42
boolean inRange(Long lowerBound, Long upperBound, Long[] exclusions)	43
boolean inRange(Long[][] listOfRanges, Long[] exclusions)	43
boolean inRange(Long[][] listOfRanges, boolean inclusiveLower, boolean inclusiveUpper, Long[] exclusions)	44
boolean inRange(String lowerBound, String upperBound)	45
boolean inRange(String lowerBound, String upperBound, String[] exclusions)	46
boolean inRange(String[][] listOfRanges, String[] exclusions)	46
boolean inRange(String[][] listOfRanges, boolean ignoreCase, boolean inclusiveLower, boolean inclusiveUpper, String[] exclusions)	47
object list createList(Object[] list)	48
object list appendToList(Object[] list, Object item)	48
object list insertIntoList(Object[] list, Object item, int position)	49
object list removeFromList(Object[] list, int position)	49
Summary of Math Functions	50
long abs(long value)	53
double abs(long double)	53
double acos(double val)	53
double asin(double val)	54
double atan(double val)	54
double ceil(double val)	54
double cos(double val)	55
double cosh(double val)	55
double degreesToRadians(double angdeg)	55
double exp(double val)	56
double floor(double val)	56
double log(double val)	57
long max(long val1, long val2)	57
double max(double val1, double val2)	57
long min(long val1, long val2)	58

double min(double val1, double val2).....	58
long mod(long val1, long val2).....	59
double mod(double val1, double val2).....	59
double pi().....	59
double pow(double base, double exponent).....	60
double radiansToDegrees(double angrad).....	60
long round(double val).....	60
double round(double val, int scale).....	61
double round(double val, int scale, int roundingMethod).....	61
double sin(double val).....	62
double sinh(double val).....	62
double tan(double val).....	63
double tanh(double val).....	63
Summary of String Functions.....	64
concat(String str): String.....	65
contains(String str): boolean.....	66
endsWith(String suffix): boolean.....	66
equals(String anotherString): boolean.....	66
equalsIgnoreCase(String anotherString): boolean.....	67
matches(String regex): boolean.....	67
regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len): boolean.....	68
replaceAll(String regex, String replacement): String.....	68
replaceFirst(String regex, String replacement): String.....	69
startsWith(String prefix): boolean.....	69
substring(int beginIndex): String.....	70
substring(int beginIndex, int endIndex): String.....	70
toLowerCase(): String.....	70
toUpperCase(): String.....	71
trim(): String.....	71
<b>Rules-Related Event Types Overview.....</b>	<b>73</b>
Summary of Rules-Related Event Types.....	74
Decision Entity Changed.....	75
Hot Deployment Started.....	75
Project Deleted.....	76
Project Imported.....	76
Project Exported.....	77
Project Deployed.....	77
Project Undeployed.....	78
<b>Services Overview.....</b>	<b>79</b>
Summary of WmBusinessRules Built-in Services.....	81
pub.businessrules.client.invoke.....	81
Summary of REST Services.....	83
GET <base URL>/projects.....	86
GET <base URL>/project/%ruleProjectName%.....	86

PUT <base URL>/project/%ruleProjectName%/lock.....	87
PUT <base URL>/project/%ruleProjectName%/unlock.....	87
PUT <base URL>/project/%ruleProjectName%/unlockall.....	87
PUT <base URL>/project/%ruleProjectName%/decisiontable/%decisionTableName%.....	88
PUT <base URL>/project/%ruleProjectName%/decisiontable/%decisionTableName%/lock.....	88
PUT <base URL>/project/%ruleProjectName%/decisiontable/%decisionTableName%/unlock.....	89
PUT <base URL>/project/%ruleProjectName%/deploy.....	89
GET <base URL>/project/%ruleProjectName%.....	89
PUT <base URL>/project/%ruleProjectName%.....	90
DELETE <base URL>/project/%ruleProjectName%.....	90
GET <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%.....	91
PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%.....	91
GET <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/condition_nodes/%conditionNodeId%.....	92
PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/condition_nodes/%conditionNodeId%.....	92
GET <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/lines/%conditionLinkId%.....	93
PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/lines/%conditionLinkId%.....	94
GET <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/result_nodes/%resultNodeId%.....	95
PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/result_nodes/%resultNodeId%.....	95
GET <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/results/%resultId%.....	96
PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/results/%resultId%.....	97
PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/lock.....	97
PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/unlock.....	98
GET <base URL>/projects/%ruleProjectName%/decision_tables/%decisionTableName%.....	99
<b>Technical Details for Preconfigured Verification Services.....</b>	<b>101</b>
<b>Technical Details for Preconfigured Data Provider Services.....</b>	<b>103</b>
<b>Administering Business Rules with Command Central.....</b>	<b>105</b>
Commands that Business Rules Supports.....	106
Configuration Types that Business Rules Supports.....	108
Run-Time Monitoring Statuses for Business Rules.....	109

---

## About this Guide

---

*webMethods Business Rules Reference* describes the structure of rule functions, rules-related event types and wMBusinessRules built-in services that are part of the Rules Development feature of Software AG Designer. It also provides information about preconfigured REST services you can use with Business Rules and about how to administer Business Rules with Command Central.

*webMethods Business Rules Reference* contains supporting documentation on the following main topics:

- ["Functions Overview" on page 11.](#)
- ["Rules-Related Event Types Overview" on page 73.](#)
- ["Services Overview" on page 79.](#)
- ["Technical Details for Preconfigured Verification Services" on page 101.](#)
- ["Technical Details for Preconfigured Data Provider Services" on page 103.](#)
- ["Administering Business Rules with Command Central" on page 105.](#)

With respect to processing of personal data according to the EU General Data Protection Regulation (GDPR), appropriate steps are documented in *webMethods BPM Rules Development Help, Processing Personal Data*.

## Document Conventions

---

Convention	Description
<b>Bold</b>	Identifies elements on a screen.
Narrowfont	Identifies storage locations for services on webMethods Integration Server, using the convention <i>folder.subfolder:service</i> .
UPPERCASE	Identifies keyboard keys. Keys you must press simultaneously are joined with a plus sign (+).
<i>Italic</i>	Identifies variables for which you must supply values specific to your own situation or environment. Identifies new terms the first time they occur in the text.

---

---

Convention	Description
Monospace font	Identifies text you must type or messages displayed by the system.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the   symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

## Online Information

---

### Software AG Documentation Website

You can find documentation on the Software AG Documentation website at <http://documentation.softwareag.com>. The site requires Empower credentials. If you do not have Empower credentials, you must use the TECHcommunity website.

### Software AG Empower Product Support Website

You can find product information on the Software AG Empower Product Support website at <https://empower.softwareag.com>.

To submit feature/enhancement requests, get information about product availability, and download products, go to [Products](#).

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the [Knowledge Center](#).

### Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at <http://techcommunity.softwareag.com>. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.



- 
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
  - Link to external websites that discuss open standards and web technology.



# 1 Functions Overview

- Summary of Conversion Functions ..... 15
- toBoolean() ..... 16
- toDate() ..... 16
- toDate() ..... 17
- toDate(String dateFormat) ..... 17
- toDouble() ..... 18
- toDouble() ..... 18
- toLong() ..... 18
- toLong() ..... 19
- toString() ..... 19
- toString() ..... 19
- toString() ..... 20
- toString() ..... 20
- toString(String dateFormat) ..... 21
- Summary of Date Functions ..... 21
- int century() ..... 23
- int compareDates(Date date1, Date date2) ..... 24
- Date date() ..... 24
- Date dateMinusDays(int days) ..... 25
- Date datePlusDays(int days) ..... 25
- long dateToInt() ..... 25
- int dayOfMonth() ..... 26
- int dayOfWeek() ..... 26
- int dayOfYear() ..... 26
- int daysInMonth(int month, int year) ..... 27
- long diffInDays(Date anotherDate) ..... 27

■ long diffInMonths(Date anotherDate) .....	28
■ long diffInYears(Date anotherDate) .....	28
■ Date intToDate(long millis) .....	28
■ isLeapYear(int year) .....	29
■ int month() .....	29
■ String time() .....	30
■ boolean verifyDate(int year, int month, int day) .....	30
■ boolean verifyIntDate(long millis) .....	30
■ boolean verifyMonth(int month) .....	31
■ boolean verifyYear(int year) .....	31
■ int year() .....	32
■ Date ymdToDate(int year, int month, int day) .....	32
■ Summary of List and Range Functions .....	32
■ boolean inList(String[] list) .....	35
■ boolean inList(String[] list, boolean ignoreCase) .....	36
■ boolean inRange(Date lowerBound, Date upperBound) .....	36
■ boolean inRange(Date lowerBound, Date upperBound, Date[] exclusions) .....	37
■ boolean inRange(Date[][] listOfRanges, Date[] exclusions) .....	37
■ boolean inRange(Date[][] listOfRanges, boolean inclusiveLower, boolean inclusiveUpper, Date[] exclusions) .....	38
■ boolean inRange(Double lowerBound, Double upperBound) .....	39
■ boolean inRange(Double lowerBound, Double upperBound, Double[] exclusions) .....	39
■ boolean inRange(Double[][] listOfRanges, Double[] exclusions) .....	40
■ boolean inRange(Double[][] listOfRanges, boolean inclusiveLower, boolean inclusiveUpper, Double[] exclusions) .....	41
■ boolean inRange(Long lowerBound, Long upperBound) .....	42
■ boolean inRange(Long lowerBound, Long upperBound, Long[] exclusions) .....	43
■ boolean inRange(Long[][] listOfRanges, Long[] exclusions) .....	43
■ boolean inRange(Long[][] listOfRanges, boolean inclusiveLower, boolean inclusiveUpper, Long[] exclusions) .....	44
■ boolean inRange(String lowerBound, String upperBound) .....	45

■ boolean inRange(String lowerBound, String upperBound, String[] exclusions) .....	46
■ boolean inRange(String[][] listOfRanges, String[] exclusions) .....	46
■ boolean inRange(String[][] listOfRanges, boolean ignoreCase, boolean inclusiveLower, boolean inclusiveUpper, String[] exclusions) .....	47
■ object list createList(Object[] list) .....	48
■ object list appendToList(Object[] list, Object item) .....	48
■ object list insertIntoList(Object[] list, Object item, int position) .....	49
■ object list removeFromList(Object[] list, int position) .....	49
■ Summary of Math Functions .....	50
■ long abs(long value) .....	53
■ double abs(long double) .....	53
■ double acos(double val) .....	53
■ double asin(double val) .....	54
■ double atan(double val) .....	54
■ double ceil(double val) .....	54
■ double cos(double val) .....	55
■ double cosh(double val) .....	55
■ double degreesToRadians(double angdeg) .....	55
■ double exp(double val) .....	56
■ double floor(double val) .....	56
■ double log(double val) .....	57
■ long max(long val1, long val2) .....	57
■ double max(double val1, double val2) .....	57
■ long min(long val1, long val2) .....	58
■ double min(double val1, double val2) .....	58
■ long mod(long val1, long val2) .....	59
■ double mod(double val1, double val2) .....	59
■ double pi() .....	59
■ double pow(double base, double exponent) .....	60
■ double radiansToDegrees(double angrad) .....	60

■ long round(double val) .....	60
■ double round(double val, int scale) .....	61
■ double round(double val, int scale, int roundingMethod) .....	61
■ double sin(double val) .....	62
■ double sinh(double val) .....	62
■ double tan(double val) .....	63
■ double tanh(double val) .....	63
■ Summary of String Functions .....	64
■ concat(String str): String .....	65
■ contains(String str): boolean .....	66
■ endsWith(String suffix): boolean .....	66
■ equals(String anotherString): boolean .....	66
■ equalsIgnoreCase(String anotherString): boolean .....	67
■ matches(String regex): boolean .....	67
■ regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len): boolean .....	68
■ replaceAll(String regex, String replacement): String .....	68
■ replaceFirst(String regex, String replacement): String .....	69
■ startsWith(String prefix): boolean .....	69
■ substring(int beginIndex): String .....	70
■ substring(int beginIndex, int endIndex): String .....	70
■ toLowerCase(): String .....	70
■ toUpperCase(): String .....	71
■ trim(): String .....	71

webMethods Rules Development provides a set of predefined functions. For detailed information about how to work with functions, see *webMethods BPM Rules Development Help*.

Five categories of functions exist:

- ["Summary of Conversion Functions" on page 15.](#)
- ["Summary of Date Functions" on page 21.](#)
- ["Summary of List and Range Functions" on page 32.](#)
- ["Summary of Math Functions" on page 50.](#)
- ["Summary of String Functions" on page 64.](#)

## Summary of Conversion Functions

webMethods Rules Development provides the following predefined conversion functions:

Function	Returns	Description
<a href="#">toBoolean()</a>	Boolean	Returns a boolean with a value represented by this string.
<a href="#">toDate()</a>	Date	Returns a date that this string represents.
<a href="#">toDate()</a>	Date Object	Allocates a date object for this long value.
<a href="#">toDate(String dateFormat)</a>	Date	Returns a date that represents this string based on the specified format.
<a href="#">toDouble()</a>	Double	Returns a double value holding the value of this string.
<a href="#">toDouble()</a>	Double	Returns a double value represented by this long value.
<a href="#">toLong()</a>	Long	Returns a long value holding the value of this string.

Function	Returns	Description
<code>toLong()</code>	Long	Returns a long value represented by this double value.
<code>toString()</code>	String	Returns a string representing the value of this long value.
<code>toString()</code>	String	Returns a string representing the value of this double value.
<code>toString()</code>	String Object	Returns a string object representing the value of this boolean.
<code>toString()</code>	String	Returns a string representing the value of this date.
<code>toString(String dateFormat)</code>	String	Returns a string representing the value of this date in the specified format.

## toBoolean()

Returns a boolean with a value represented by this string. The boolean returned represents a true value if the string argument is not null and is equal, ignoring case, to the string `true`.

### Input Parameters

None.

### Return Value

**Boolean** The boolean value represented by this string.

## toDate()

Returns a date that this string represents.



**Note:** You should only use this function if the format of the date is unknown. If the date format is known, you should use `toDate(String dateFormat)` which is more efficient.

---

### Input Parameters

None.

---

### Return Value

**Date** A date if it can be determined, `null` otherwise.

---

## toDate()

Allocates a date object and initializes it to represent the number of milliseconds since the standard base time known as "the epoch" (namely January 1, 1970, 00:00:00 GMT) for this long value.

---

### Input Parameters

None.

---

### Return Value

**Date Object** A date object.

---

## toDate(String dateFormat)

Returns a date that represents this string based on the specified format.

**Note:** If you do not know the format of the date, you can use `toDate()` which attempts to determine the format, but is less efficient.

---

### Input Parameters

*dateFormat*

**String** A pattern that defines the format of this date. The pattern is based on Java date and time patterns.

**Return Value**

---

**Date** A date that represents this string based on the specified format.

---

**toDouble()**

Returns a double value holding the value of this string.

**Input Parameters**

---

None.

**Return Value**

---

**Double** A double value holding the value of this string.

---

**toDouble()**

Returns a double value represented by this long value.

**Input Parameters**

---

None.

**Return Value**

---

**Double** A double value represented by this long value.

---

**toLong()**

Returns a long value holding the value of this string.

**Input Parameters**

---

None.

**Return Value**

---

**Long** A long value holding the value of this string.

---

**toLong()**

Returns a long value represented by this double value.

**Note:** Truncation toward zero will occur.

**Input Parameters**

---

None.

**Return Value**

---

**Long** A long value holding the value (truncated toward zero) of this double value.

---

**toString()**

Returns a string representing the value of this long value.

**Input Parameters**

---

None.

**Return Value**

---

**String** A string representation of the value of this object in base 10.

---

**toString()**

Returns a string representing the value of this double value.

**Input Parameters**

---

None.

**Return Value**

---

**String** A string representation of the value of this object in base 10.

---

**toString()**

Returns a string object representing the value of this boolean. If this object represents the value `true`, a string equal to `true` is returned. Otherwise, a string equal to `false` is returned.

**Input Parameters**

---

None.

**Return Value**

---

**String** A string representation of the value of this object.

---

**toString()**

Returns a string representing the value of this date based on the locale that the application is running under.

**Input Parameters**

---

None.

**Return Value**

---

**String** A string representing the value of this date based on the locale that the application is running under.

---

## toString(String dateFormat)

Returns a string representing the value of this date in the specified format.

### Input Parameters

*DateFormat*                      **String** A pattern that defines the format of this date. The pattern is based on Java date and time patterns.

### Return Value

**String** A string representing the value of this date in the specified format.

---

## Summary of Date Functions

webMethods Rules Development provides the following predefined date functions:

**Note:**     Dates are based on the ISO 8601 standard which is based on the proleptic Gregorian calendar.

Function	Returns	Description
<code>int century()</code>	Integer	Returns the century for this date.
<code>int compareDates(Date date1, Date date2)</code>	Integer	Compares one date against another date.
<code>Date date()</code>	Date	Returns the current date from the server where the decision entity is invoked.
<code>Date dateMinusDays(int days)</code>	Date	Returns a copy of this date minus the specified number of days.
<code>Date datePlusDays(int days)</code>	Date	Returns a copy of this date plus the specified number of days.

Function	Returns	Description
<code>long dateToInt()</code>	Integer	Returns the number of milliseconds since the standard base time.
<code>int dayOfMonth()</code>	Integer	Returns the day of the month for this date.
<code>int dayOfWeek()</code>	Integer	Returns the day of the week for this date.
<code>int dayOfYear()</code>	Integer	Returns the day of the year for this date.
<code>int daysInMonth(int month, int year)</code>	Integer	Returns the number of days in the specified month in the given year.
<code>long diffInDays(Date anotherDate)</code>	Integer	Returns the mathematical difference in days between this date and the specified date ignoring the times of both dates.
<code>long diffInMonths(Date anotherDate)</code>	Integer	Returns the mathematical difference in months between this date and the specified date ignoring the times of both dates.
<code>long diffInYears(Date anotherDate)</code>	Integer	Returns the mathematical difference in years between this date and the specified date ignoring the times of both dates.
<code>Date intToDate(long millis)</code>	Date	Returns a date that represents the specified number of milliseconds since the standard base time.
<code>isLeapYear(int year)</code>	Boolean	Indicates whether or not the specified year is a leap year.

Function	Returns	Description
<code>int month()</code>	Integer	Returns the numeric month for this date.
<code>String time()</code>	String	Returns the string representation of the time this date as Hours:Minutes:Seconds.Milliseconds.
<code>boolean verifyDate(int year, int month, int day)</code>	Boolean	Verifies that the date represented by the specified year, month and day is valid.
<code>boolean verifyIntDate(long millis)</code>	Boolean	Verifies that the specified number of milliseconds is a valid date representation.
<code>boolean verifyMonth(int month)</code>	Boolean	Verifies that the specified numeric month is valid.
<code>boolean verifyYear(int year)</code>	Boolean	Verifies that the specified numeric year is valid.
<code>int year()</code>	Integer	Returns the numeric 4-digit year for this date.
<code>Date ymdToDate(int year, int month, int day)</code>	Date	Creates and returns a date from three integers (year, month, day).

## `int century()`

Returns the century for this date. The century is based on the era, where era is expressed as a constant, zero for BC/BCE, one for AD/CE.

### Input Parameters

None.

**Return Value**

---

**Integer** The century for this date.

---

**int compareDates(Date date1, Date date2)**

Compares one date against another date and returns +1 if the first date is greater than the second date, -1 if the first date is less than the second date, and 0 if the dates are equal.

**Input Parameters**

---

*date1*                                    **Date** The first date to compare.

*date2*                                    **Date** The second date to compare against the first date.

**Return Value**

---

**Integer** The value 0 if dates are equal. The value -1 if the first date is before the second date. The value +1 if the first date is after the second date.

---

**Date date()**

Returns the current date from the server where the decision entity is invoked, measured to the nearest millisecond.

**Input Parameters**

---

None.

**Return Value**

---

**Date** The current date from the server where the decision entity is invoked, measured to the nearest millisecond.





**Return Value**

---

**Integer** The number of milliseconds since January 1, 1970, 00:00:00 GMT represented by this date.

---

**int dayOfMonth()**

Returns the day of the month for this date (1 - 31 depending on the month).

**Input Parameters**

---

None.

**Return Value**

---

**Integer** The day of month for this date.

---

**int dayOfWeek()**

Returns the day of the week represented by this date. The returned value (1 = Sunday, 2 = Monday, 3 = Tuesday, 4 = Wednesday, 5 = Thursday, 6 = Friday, 7 = Saturday) represents the day of the week that contains or begins with the instant in time represented by this date, as interpreted in the local time zone.

**Input Parameters**

---

None.

**Return Value**

---

**Integer** The day of week for this date.

---

**int dayOfYear()**

Returns the day of the year for this date (1 - 366 depending on the year).

**Input Parameters**

---

None.

**Return Value**

---

**Integer** The day of the year for this date.

---

**int daysInMonth(int month, int year)**

Returns the number of days in the specified month in the given year.

**Input Parameters**

---

*month* **Integer** The numeric month (1 = January, 2 = February, ..., 12 = December).

*year* **Integer** The numeric 4-digit year.

**Return Value**

---

**Integer** The number of days in the specified month in the given year.

---

**long diffInDays(Date anotherDate)**

Returns the mathematical difference in days between this date and the specified date ignoring the times of both dates.

**Input Parameters**

---

*anotherDate* **Date** The date to be used to compute the mathematical difference in days from this date.

**Return Value**

---

**Integer** The mathematical difference in days between this date and the specified date ignoring the times of both dates.

---

## long diffInMonths(Date anotherDate)

Returns the mathematical difference in months between this date and the specified date ignoring the times of both dates.

### Input Parameters

---

*anotherDate*                      **Date** The date to be used to compute the mathematical difference in months from this date.

### Return Value

---

**Integer** The mathematical difference in months between this date and the specified date ignoring the times of both dates.

---

## long diffInYears(Date anotherDate)

Returns the mathematical difference in years between this date and the specified date ignoring the times of both dates.

### Input Parameters

---

*anotherDate*                      **Date** The date to be used to compute the mathematical difference in years from this date.

### Return Value

---

**Integer** The mathematical difference in years between this date and the specified date ignoring the times of both dates.

---

## Date intToDate(long millis)

Returns a date that represents the specified number of milliseconds since the standard base time known as "the epoch", namely January 1, 1970, 00:00:00 GMT.



---

## String time()

Returns the string representation of the time of this date using the default formatting style of the default locale.

### Input Parameters

---

None.

### Return Value

---

**String** The time for this date.

---

## boolean verifyDate(int year, int month, int day)

Verifies that the date represented by the specified year, month and day is valid.

### Input Parameters

---

<i>year</i>	<b>Integer</b> The 4-digit year of the date.
<i>month</i>	<b>Integer</b> The numeric month within the specified year (1 = January, 2 = February, ..., 12 = December).
<i>day</i>	<b>Integer</b> The numeric day within the specified month.

### Return Value

---

**Boolean** Returns `true` if the date represented by the specified year, month and day is valid, `false` otherwise.

---

## boolean verifyIntDate(long millis)

Verifies that the specified number of milliseconds is a valid date representation. (Should be the number of milliseconds since the standard base time known as "the epoch", namely January 1, 1970, 00:00:00 GMT.)



---

## int year()

Returns the numeric 4-digit year for this date.

### Input Parameters

---

None.

### Return Value

---

**Integer** The numeric 4-digit year for this date.

---

## Date ymdToDate(int year, int month, int day)

Creates and returns a date from three integers (year, month, day).

### Input Parameters

---

<i>year</i>	<b>Integer</b> The 4-digit year of the date.
<i>month</i>	<b>Integer</b> The numeric month within the specified year (1 = January, 2 = February, ..., 12 = December).
<i>day</i>	<b>Integer</b> The numeric day within the specified month.

### Return Value

---

**Date** A date for the specified year, month and day.

---

## Summary of List and Range Functions

webMethods Rules Development provides the following predefined list and range functions:



Function	Returns	Description
<code>boolean inList(String[] list)</code>	Boolean	Indicates whether or not this string is in the specified list.
<code>boolean inList(String[] list, boolean ignoreCase)</code>	Boolean	Indicates whether or not this string is in the specified list. Case sensitivity can be ignored while checking.
<code>boolean inRange(Date lowerBound, Date upperBound)</code>	Boolean	Indicates whether or not this date is within the specified range.
<code>boolean inRange(Date lowerBound, Date upperBound, Date[] exclusions)</code>	Boolean	Indicates whether or not this date is within the specified range. Exclusions can be specified.
<code>boolean inRange(Date[][] listOfRanges, Date[] exclusions)</code>	Boolean	Indicates whether or not this date is within the specified list of ranges. Exclusions can be specified.
<code>boolean inRange(Date[][] listOfRanges, boolean inclusiveLower, boolean inclusiveUpper, Date[] exclusions)</code>	Boolean	Indicates whether or not this date is within the specified list of ranges. Exclusions and inclusion of lower and upper end of range can be specified.
<code>boolean inRange(Double lowerBound, Double upperBound)</code>	Boolean	Indicates whether or not this floating point (double, float) value is within the specified range.
<code>boolean inRange(Double lowerBound, Double upperBound, Double[] exclusions)</code>	Boolean	Indicates whether or not this floating point (double, float) value is within the specified range. A separate list of exclusions can be provided.
<code>boolean inRange(Double[][] listOfRanges, Double[] exclusions)</code>	Boolean	Indicates whether or not this floating point (double, float) value is within the specified

Function	Returns	Description
		list of ranges. A separate list of exclusions can be provided.
<code>boolean inRange(Double[][] listOfRanges, boolean inclusiveLower, boolean inclusiveUpper, Double[] exclusions)</code>	Boolean	Indicates whether or not this floating point (double, float) value is within the specified list of ranges. Upper and lower end of list can optionally be included. A separate list of exclusions can be provided.
<code>boolean inRange(Long lowerBound, Long upperBound)</code>	Boolean	Indicates whether or not this integer (long, integer, short) value is within the specified range.
<code>boolean inRange(Long lowerBound, Long upperBound, Long[] exclusions)</code>	Boolean	Indicates whether or not this integer (long, integer, short) value is within the specified range. A separate list of exclusions can be provided.
<code>boolean inRange(Long[][] listOfRanges, Long[] exclusions)</code>	Boolean	Indicates whether or not this integer (long, integer, short) value is within the specified list of ranges. A separate list of exclusions can be provided.
<code>boolean inRange(Long[][] listOfRanges, boolean inclusiveLower, boolean inclusiveUpper, Long[] exclusions)</code>	Boolean	Indicates whether or not this integer (long, integer, short) value is within the specified list of ranges. Upper and lower end of list can optionally be included. A separate list of exclusions can be provided.
<code>boolean inRange(String lowerBound, String upperBound)</code>	Boolean	Indicates whether or not this string is within the specified range.
<code>boolean inRange(String lowerBound, String upperBound, String[] exclusions)</code>	Boolean	Indicates whether or not this string is within the specified range. A separate list of exclusions can be provided.

Function	Returns	Description
<code>boolean inRange(String[] list, String[] exclusions)</code>	Boolean	Indicates whether or not this string is within the specified list of ranges. A separate list of exclusions can be provided.
<code>boolean inRange(String[] list, boolean ignoreCase, boolean inclusiveLower, boolean inclusiveUpper, String[] exclusions)</code>	Boolean	Indicates whether or not this string is within the specified list of ranges. Upper and lower end of list can optionally be included. A separate list of exclusions can be provided. Case sensitivity can be ignored while checking.
<code>object list createList(Object[] list)</code>	Object list	Creates a list from an array of items.
<code>object list appendToList(Object[] list, Object item)</code>	Object list	Appends an item to the end of a list.
<code>object list insertIntoList(Object[] list, Object item, int position)</code>	Object list	Inserts an item into an existing list at the specified position.
<code>object list removeFromList(Object[] list, int position)</code>	Object list	Removes an item from a list at the specified position.

## boolean inList(String[] list)

Indicates whether or not this string is in the specified list. The case (upper, lower) of the string is taken into consideration when matching against the list.

### Input Parameters

*list* **String List** A list of strings to match this string against.

### Return Value

**Boolean** Returns `true` if this string exists in the specified list, `false` otherwise.

---

## boolean inList(String[] list, boolean ignoreCase)

Indicates whether or not this string is in the specified list. Case sensitivity can be ignored while checking.

### Input Parameters

---

<i>list</i>	<b>String List</b> A list of strings to match this string against.
<i>ignoreCase</i>	<b>Boolean</b> Indicates whether or not case sensitivity should be ignored.

### Return Value

---

**Boolean** Returns `true` if this string exists in the specified list, `false` otherwise.

---

## boolean inRange(Date lowerBound, Date upperBound)

Indicates whether or not this date is within the specified range. The checking is inclusive, meaning that the lower and upper bound of the range will be tested for the date.

### Input Parameters

---

<i>lowerBound</i>	<b>Date</b> The lower bound of the range to check against (inclusive).
<i>upperBound</i>	<b>Date</b> The upper bound of the range to check against (inclusive).

### Return Value

---

**Boolean** Returns `true` if this date exists within the specified range, `false` otherwise.

---

## boolean inRange(Date lowerBound, Date upperBound, Date[] exclusions)

Indicates whether or not this date is within the specified range. The checking is inclusive, meaning that the lower and upper bound of the range will be tested for the date. A separate list of exclusions can be provided to indicate that even though the date is within the list of ranges, it should not be accepted if found within the exclusions list.

### Input Parameters

---

<i>lowerBound</i>	<b>Date</b> The lower bound of the range to check against (inclusive).
<i>upperBound</i>	<b>Date</b> The upper bound of the range to check against (inclusive).
<i>exclusions</i>	<b>Date</b> An array of items to be excluded from the range check.

### Return Value

---

**Boolean** Returns `true` if this date exists within the specified range, `false` otherwise.

---

## boolean inRange(Date[][] listOfRanges, Date[] exclusions)

Indicates whether or not this date is within the specified list of ranges. The checking is inclusive, meaning that the lower and upper bound of the range will be tested for the date. A separate list of exclusions can be provided to indicate that even though the date is within the list of ranges, it should not be accepted if found within the exclusions list.

### Input Parameters

---

<i>listOfRanges</i>	<b>Date List</b> The list of ranges to check against. This is a two-dimensional date array with the outer dimension containing the list of ranges and the inner dimension containing the upper and lower bounds of each range. The outer dimension can contain 1 - n elements, while the inner dimension must always contain exactly two elements.
---------------------	--

---

*exclusions*                      **Date** An array of items to be excluded from the range check.

### Return Value

---

**Boolean** Returns `true` if this date exists within the specified list of ranges, `false` otherwise.

---

## boolean `inRange(Date[][] listOfRanges, boolean inclusiveLower, boolean inclusiveUpper, Date[] exclusions)`

Indicates whether or not this date is within the specified list of ranges. A separate list of exclusions can be provided to indicate that even though the date is within the list of ranges, it should not be accepted if found within the exclusions list.

### Input Parameters

---

*listOfRanges*                      **Date List** The list of ranges to check against. This is a two-dimensional date array with the outer dimension containing the list of ranges and the inner dimension containing the upper and lower bounds of each range. The outer dimension can contain 1 - n elements, while the inner dimension must always contain exactly two elements.

*inclusiveLower*                      **Boolean** Indicates whether or not to include the lower end of each range.

*inclusiveUpper*                      **Boolean** Indicates whether or not to include the upper end of each range.

*exclusions*                      **Date** An array of items to be excluded from the range check.

### Return Value

---

**Boolean** Returns `true` if this date exists within the specified list of ranges, `false` otherwise.

---

## boolean inRange(Double lowerBound, Double upperBound)

Indicates whether or not this floating point (double, float) value is within the specified range. The checking is inclusive, meaning that the lower and upper bound of the range will be tested for the double value.

**Note:** The double data type is a double-precision 64-bit IEEE 754 floating point. A double literal is of type double if it contains a decimal (e.g., 7.9). Double and float data types can be passed as arguments to the function. The inRange function is overloaded and supports other numeric ranges such as integer, short, long, etc. To ensure that the correct signature is invoked, make sure that the proper numeric syntax is used (no decimal for integer, decimal for floating point).

### Input Parameters

---

*lowerBound*                      **Integer** The lower bound of the range to check against (inclusive).

*upperBound*                      **Integer** The upper bound of the range to check against (inclusive).

### Return Value

---

**Boolean** Returns `true` if this integer exists within the specified range, `false` otherwise.

---

## boolean inRange(Double lowerBound, Double upperBound, Double[] exclusions)

Indicates whether or not this floating point (double, float) value is within the specified range. The checking is inclusive, meaning that the lower and upper bounds of the range will be tested for the double value. A separate list of exclusions can be provided to indicate that even though the double value is within the range, it should not be accepted if found within the exclusions list.

**Note:** The double data type is a double-precision 64-bit IEEE 754 floating point. A double literal is of type double if it contains a decimal (e.g., 7.9). Double and float data types can be passed as arguments to the function. The inRange function is overloaded and supports other numeric ranges such as integer, short, long, etc. To ensure that the correct signature is invoked, make sure that

the proper numeric syntax is used (no decimal for integer, decimal for floating point).

### Input Parameters

<i>lowerBound</i>	<b>Integer</b> The lower bound of the range to check against (inclusive).
<i>upperBound</i>	<b>Integer</b> The upper bound of the range to check against (inclusive).
<i>exclusions</i>	<b>Integer List</b> An array of items to be excluded from the range check.

### Return Value

**Boolean** Returns `true` if this integer exists within the specified range, `false` otherwise.

## **boolean inRange(Double[][] listOfRanges, Double[] exclusions)**

Indicates whether or not this floating point (double, float) value is within the specified list of ranges. The checking is inclusive, meaning that the lower and upper bounds of the ranges will be tested for the double value. A separate list of exclusions can be provided to indicate that even though the double value is within the list of ranges, it should not be accepted if found within the exclusions list.

**Note:** The double data type is a double-precision 64-bit IEEE 754 floating point. A double literal is of type `double` if it contains a decimal (e.g., `7.9`). `Double` and `float` data types can be passed as arguments to the function. The `inRange` function is overloaded and supports other numeric ranges such as `integer`, `short`, `long`, etc. To ensure that the correct signature is invoked, make sure that the proper numeric syntax is used (no decimal for integer, decimal for floating point).

### Input Parameters

<i>listOfRanges</i>	<b>Integer List</b> The list of ranges to check against. This is a two-dimensional long array with the outer dimension containing the list of ranges and the inner dimension containing the upper and lower bounds of each range.
---------------------	---



The outer dimension can contain 1-n elements, while the inner dimension must always contain exactly two elements.

*exclusions*

**Integer List** An array of items to be excluded from the range check.

---

### Return Value

**Boolean** Returns `true` if this integer exists within the specified list of ranges, `false` otherwise.

---

## **boolean inRange(Double[][] listOfRanges, boolean inclusiveLower, boolean inclusiveUpper, Double[] exclusions)**

Indicates whether or not this floating point (double, float) value is within the specified list of ranges. A separate list of exclusions can be provided to indicate that even though the double value is within the list of ranges, it should not be accepted if found within the exclusions list. Upper and lower end of list can optionally be included.

**Note:** The double data type is a double-precision 64-bit IEEE 754 floating point. A double literal is of type `double` if it contains a decimal (e.g., 7.9). `Double` and `float` data types can be passed as arguments to the function. The `inRange` function is overloaded and supports other numeric ranges such as `integer`, `short`, `long`, etc. To ensure that the correct signature is invoked, make sure that the proper numeric syntax is used (no decimal for integer, decimal for floating point).

---

### Input Parameters

*listOfRanges*

**Integer List** The list of ranges to check against. This is a two-dimensional long array with the outer dimension containing the list of ranges and the inner dimension containing the upper and lower bounds of each range. The outer dimension can contain 1-n elements, while the inner dimension must always contain exactly two elements.

*inclusiveLower*

**Boolean** Indicates whether or not to include the lower end of each range.

---

<i>inclusiveUpper</i>	<b>Boolean</b> Indicates whether or not to include the upper end of each range.
<i>exclusions</i>	<b>Integer List</b> An array of items to be excluded from the range check.

### Return Value

---

**Boolean** Returns `true` if this integer exists within the specified list of ranges, `false` otherwise.

---

## boolean inRange(Long lowerBound, Long upperBound)

Indicates whether or not this integer (long, integer, short) value is within the specified range. The checking is inclusive, meaning that the lower and upper bound of the range will be tested for the long value.

**Note:** The long data type is a 64-bit two's complement integer. The signed long has a minimum value of  $-2^{63}$  and a maximum value of  $2^{63}-1$ . Long, integer and short data types can be passed as arguments to the function. The `inRange` function is overloaded and supports other numeric ranges such as double and float. To ensure that the correct signature is invoked, make sure that the proper numeric syntax is used (no decimal for integer, decimal for floating point).

### Input Parameters

---

<i>lowerBound</i>	<b>Integer</b> The lower bound of the range to check against (inclusive).
<i>upperBound</i>	<b>Integer</b> The upper bound of the range to check against (inclusive).

### Return Value

---

**Boolean** Returns `true` if this integer exists within the specified range, `false` otherwise.

---

## boolean inRange(Long lowerBound, Long upperBound, Long[] exclusions)

Indicates whether or not this integer (long, integer, short) value is within the specified range. The checking is inclusive, meaning that the lower and upper bounds of the range will be tested for the long value. A separate list of exclusions can be provided to indicate that even though the long value is within the range, it should not be accepted if found within the exclusions list.

**Note:** The long data type is a 64-bit two's complement integer. The signed long has a minimum value of  $-2^{63}$  and a maximum value of  $2^{63}-1$ . Long, integer and short data types can be passed as arguments to the function. The inRange function is overloaded and supports other numeric ranges such as double and float. To ensure that the correct signature is invoked, make sure that the proper numeric syntax is used (no decimal for integer, decimal for floating point).

---

### Input Parameters

<i>lowerBound</i>	<b>Integer</b> The lower bound of the range to check against (inclusive).
<i>upperBound</i>	<b>Integer</b> The upper bound of the range to check against (inclusive).
<i>exclusions</i>	<b>Integer List</b> An array of items to be excluded from the range check.

---

### Return Value

**Boolean** Returns `true` if this integer exists within the specified range, `false` otherwise.

---

## boolean inRange(Long[][] listOfRanges, Long[] exclusions)

Indicates whether or not this integer (long, integer, short) value is within the specified list of ranges. The checking is inclusive, meaning that the lower and upper bounds of the ranges will be tested for the long value. A separate list of exclusions can be provided to indicate that even though the long value is within the list of ranges, it should not be accepted if found within the exclusions list.

**Note:** The long data type is a 64-bit two's complement integer. The signed long has a minimum value of  $-2^{63}$  and a maximum value of  $2^{63}-1$ . Long, integer and short data types can be passed as arguments to the function. The `inRange` function is overloaded and supports other numeric ranges such as double and float. To ensure that the correct signature is invoked, make sure that the proper numeric syntax is used (no decimal for integer, decimal for floating point).

### Input Parameters

*listOfRanges*

**Integer List** The list of ranges to check against. This is a two-dimensional long array with the outer dimension containing the list of ranges and the inner dimension containing the upper and lower bounds of each range. The outer dimension can contain 1-n elements, while the inner dimension must always contain exactly two elements.

*exclusions*

**Integer List** An array of items to be excluded from the range check.

### Return Value

**Boolean** Returns `true` if this integer exists within the specified list of ranges, `false` otherwise.

## **boolean inRange(Long[][] listOfRanges, boolean inclusiveLower, boolean inclusiveUpper, Long[] exclusions)**

Indicates whether or not this integer (long, integer, short) value is within the specified list of ranges. A separate list of exclusions can be provided to indicate that even though the long value is within the list of ranges, it should not be accepted if found within the exclusions list. Upper and lower end of list can optionally be included.

**Note:** The long data type is a 64-bit two's complement integer. The signed long has a minimum value of  $-2^{63}$  and a maximum value of  $2^{63}-1$ . Long, integer and short data types can be passed as arguments to the function. The `inRange` function is overloaded and supports other numeric ranges such as double and float. To ensure that the correct signature is invoked, make sure that the proper numeric syntax is used (no decimal for integer, decimal for floating point).

---

## Input Parameters

---

<i>listOfRanges</i>	<b>Integer List</b> The list of ranges to check against. This is a two-dimensional long array with the outer dimension containing the list of ranges and the inner dimension containing the upper and lower bounds of each range. The outer dimension can contain 1-n elements, while the inner dimension must always contain exactly two elements.
<i>inclusiveLower</i>	<b>Boolean</b> Indicates whether or not to include the lower end of each range.
<i>inclusiveUpper</i>	<b>Boolean</b> Indicates whether or not to include the upper end of each range.
<i>exclusions</i>	<b>Integer List</b> An array of items to be excluded from the range check.

## Return Value

---

**Boolean** Returns `true` if this integer exists within the specified list of ranges, `false` otherwise.

---

## **boolean inRange(String lowerBound, String upperBound)**

Indicates whether or not this string is within the specified range. The checking is inclusive, meaning that the lower and upper bounds of the range will be tested for the string. Case differences (upper/lower) are not ignored.

## Input Parameters

---

<i>lowerBound</i>	<b>String</b> The lower bound of the range to check against (inclusive).
<i>upperBound</i>	<b>String</b> The upper bound of the range to check against (inclusive).

### Return Value

---

**Boolean** Returns `true` if this string exists within the specified range, `false` otherwise.

---

## **boolean inRange(String lowerBound, String upperBound, String[] exclusions)**

Indicates whether or not this string is within the specified range. The checking is inclusive, meaning that the lower and upper bounds of the range will be tested for the string. A separate list of exclusions can be provided to indicate that even though the string is within the given range, it should not be accepted if found within the exclusions list. Case differences (upper/lower) are not ignored.

### Input Parameters

---

<i>lowerBound</i>	<b>String</b> The lower bound of the range to check against (inclusive).
<i>upperBound</i>	<b>String</b> The upper bound of the range to check against (inclusive).
<i>exclusions</i>	<b>String List</b> An array of items to be excluded from the range check.

### Return Value

---

**Boolean** Returns `true` if this string exists within the specified range, `false` otherwise.

---

## **boolean inRange(String[][] listOfRanges, String[] exclusions)**

Indicates whether or not this string is within the specified list of ranges. The checking is inclusive, meaning that the lower and upper bounds of the ranges will be tested for the string. A separate list of exclusions can be provided to indicate that even though the string is within the given list of ranges, it should not be accepted if found within the exclusions list. Case differences (upper/lower) are not ignored.

---

## Input Parameters

---

*listOfRanges*

**String List** The list of ranges to check against. This is a two-dimensional string array with the outer dimension containing the list of ranges and the inner dimension containing the upper and lower bounds of each range. The outer dimension can contain 1 - n elements, while the inner dimension must always contain exactly two elements.

*exclusions*

**String List** An array of items to be excluded from the range check.

---

## Return Value

---

**Boolean** Returns `true` if this string exists within the specified list of ranges, `false` otherwise.

---

# **boolean inRange(String[][] listOfRanges, boolean ignoreCase, boolean inclusiveLower, boolean inclusiveUpper, String[] exclusions)**

Indicates whether or not this string is within the specified list of ranges. A separate list of exclusions can be provided to indicate that even though the string is within any of the lists of ranges, it should not be accepted if found within the exclusions list. Upper and lower end of list can optionally be included. Case sensitivity can be ignored while checking.

---

## Input Parameters

---

*listOfRanges*

**String List** The list of ranges to check against. This is a two-dimensional string array with the outer dimension containing the list of ranges and the inner dimension containing the upper and lower bounds of each range. The outer dimension can contain 1 - n elements, while the inner dimension must always contain exactly two elements.

*ignoreCase*

**Boolean** Indicates whether or not case differences should be ignored.

---

<i>inclusiveLower</i>	<b>Boolean</b> Indicates whether or not to include the lower end of each range.
<i>inclusiveUpper</i>	<b>Boolean</b> Indicates whether or not to include the upper end of each range.
<i>exclusions</i>	<b>String List</b> An array of items to be excluded from the range check.

---

### Return Value

**Boolean** Returns `true` if this string exists within the specified list of ranges, `false` otherwise.

---

## object list `createList(Object[] list)`

Creates a list from an array of items.

---

### Input Parameters

<i>list</i>	<b>Object list</b> An array of elements whose data types must match the data type of the parameter element for which the function is being applied. This argument can either be a single parameter element that is an existing list or a newly created list made up of literals and/or existing data elements of the same type.
-------------	---

---

### Return Value

**Object list** Returns a list of elements as specified by the argument array.

---

## object list `appendToList(Object[] list, Object item)`

Appends an item to the end of a list.

---

### Input Parameters

<i>list</i>	<b>Object list</b> The list onto which the item is to be appended.
-------------	--



---

*item*                      **Varying data type** An item whose data type must match the data type of the parameter element for which the function is being applied.

#### Return Value

---

**Object list** Returns the original list of items with the new item appended to the end.

---

## object list insertIntoList(Object[] list, Object item, int position)

Inserts an item into an existing list at the specified position.

#### Input Parameters

---

*list*                      **Object list** The list into which the item is to be inserted.

*item*                      **Varying data type** The item to be inserted.

*position*                **Integer** The position within the list where the item is to be inserted.

#### Return Value

---

**Object list** Returns the original list with the inserted item.

---

## object list removeFromList(Object[] list, int position)

Removes an item from a list at the specified position.

#### Input Parameters

---

*list*                      **Object list** The list from which the item is to be removed.

*position*                **Integer** The position within the list where the item has to be removed.

---

## Return Value

---

**Object list** Returns the original list without the removed item.

---

## Summary of Math Functions

webMethods Rules Development provides the following predefined math functions:

Function	Returns	Description
<code>long abs(long value)</code>	Integer	Returns the absolute value of the specified long value.
<code>double abs(long double)</code>	Integer	Returns the absolute value of the specified double value.
<code>double acos(double val)</code>	Integer	Returns the arc cosine of the specified double value.
<code>double asin(double val)</code>	Integer	Returns the arc sine of the specified double value.
<code>double atan(double val)</code>	Integer	Returns the arc tangent of the specified double value.
<code>double ceil(double val)</code>	Integer	Returns the smallest integer that is greater than or equal to the specified double value.
<code>double cos(double val)</code>	Integer	Returns the trigonometric cosine of the specified angle.
<code>double cosh(double val)</code>	Integer	Returns the hyperbolic cosine of the specified double value.
<code>double degreesToRadians(double angdeg)</code>	Integer	Returns an approximately equivalent angle measured in radians for the specified angle measured in degrees.

---

Function	Returns	Description
<code>double exp(double val)</code>	Integer	Returns Euler's number $e$ raised to the power of the specified double value.
<code>double floor(double val)</code>	Integer	Returns the largest integer that is less than or equal to the specified double value.
<code>double log(double val)</code>	Integer	Returns the natural logarithm (base $e$ ) of the specified double value.
<code>long max(long val1, long val2)</code>	Integer	Returns the larger of the two specified long values. If the specified values are equal, then the result is that same value.
<code>double max(double val1, double val2)</code>	Integer	Returns the larger of the two specified double values. If the specified values are equal, then the result is that same value.
<code>long min(long val1, long val2)</code>	Integer	Returns the lesser of the two specified long values. If the specified values are equal, then the result is that same value.
<code>double min(double val1, double val2)</code>	Integer	Returns the lesser of the two specified double values. If the specified values are equal, then the result is that same value.
<code>long mod(long val1, long val2)</code>	Integer	Returns the remainder of two long values. The remainder is obtained when dividing <code>val1</code> by <code>val2</code> .
<code>double mod(double val1, double val2)</code>	Integer	Returns the remainder of two double values. The remainder is obtained when dividing <code>val1</code> by <code>val2</code> .

Function	Returns	Description
<code>double pi()</code>	Integer	Returns the value of pi to 15 decimal places.
<code>double pow(double base, double exponent)</code>	Integer	Returns $\text{base}^{\text{exponent}}$ or the value of the base argument raised to the power of the exponent argument.
<code>double radiansToDegrees(double angrad)</code>	Integer	Returns an approximately equivalent angle measured in degrees for the specified angle measured in radians.
<code>long round(double val)</code>	Integer	Returns the closest long integer to the specified double argument, with ties rounding up.
<code>double round(double val, int scale)</code>	Integer	Rounds the given value to the specified number of decimal places.
<code>double round(double val, int scale, int roundingMethod)</code>	Integer	Rounds the given value to the specified number of decimal places. The value is rounded using the given method which is any method defined in <code>java.math.BigDecimal</code> .
<code>double sin(double val)</code>	Integer	Returns the trigonometric sine of the specified angle.
<code>double sinh(double val)</code>	Integer	Returns the hyperbolic sine of the specified double value.
<code>double tan(double val)</code>	Integer	Returns the trigonometric tangent of the specified angle.
<code>double tanh(double val)</code>	Integer	Returns the hyperbolic tangent of the specified double value.

---

## **long abs(long value)**

Returns the absolute value of the specified long value.

### **Input Parameters**

---

*val* **Integer** The argument whose absolute value is to be determined.

### **Return Value**

---

**Integer** The absolute value of the argument.

---

## **double abs(long double)**

Returns the absolute value of the specified double value.

### **Input Parameters**

---

*val* **Integer** The argument whose absolute value is to be determined.

### **Return Value**

---

**Integer** The absolute value of the argument.

---

## **double acos(double val)**

Returns the arc cosine of the specified double value.

### **Input Parameters**

---

*val* **Integer** The value whose arc cosine is to be returned.

**Return Value**

---

**Integer** The arc cosine of the argument.

---

**double asin(double val)**

Returns the arc sine of the specified double value.

**Input Parameters**

---

*val* **Integer** The value whose arc sine is to be returned.

**Return Value**

---

**Integer** The arc sine of the argument.

---

**double atan(double val)**

Returns the arc tangent of the specified double value.

**Input Parameters**

---

*val* **Integer** The value whose arc tangent is to be returned.

**Return Value**

---

**Integer** The arc tangent of the argument.

---

**double ceil(double val)**

Returns the smallest integer that is greater than or equal to the specified double value.

**Input Parameters**

---

*val* **Integer** The value whose ceiling is to be returned.

**Return Value**

---

**Integer** The smallest (closest to negative infinity) floating-point value that is greater than or equal to the argument and is equal to a mathematical integer.

---

**double cos(double val)**

Returns the trigonometric cosine of the specified angle.

**Input Parameters**

---

*val* **Integer** An angle in radians.

**Return Value**

---

**Integer** The cosine of the argument.

---

**double cosh(double val)**

Returns the hyperbolic cosine of the specified double value.

**Input Parameters**

---

*val* **Integer** The number whose hyperbolic cosine is to be returned.

**Return Value**

---

**Integer** The hyperbolic cosine of the argument.

---

**double degreesToRadians(double angdeg)**

Returns an approximately equivalent angle measured in radians for the specified angle measured in degrees.

### Input Parameters

---

*angdeg* **Integer** An angle in degrees.

### Return Value

---

**Integer** The measurement of the angle *angdeg* in radians.

## double exp(double val)

Returns Euler's number  $e$  raised to the power of the specified double value.

### Input Parameters

---

*val* **Integer** The exponent to raise  $e$  to.

### Return Value

---

**Integer** Value  $e^{val}$ , where  $e$  is the base of the natural logarithms.

## double floor(double val)

Returns the largest integer that is less than or equal to the specified double value.

### Input Parameters

---

*val* **Integer** The value whose floor is to be returned.

### Return Value

---

**Integer** The largest (closest to positive infinity) floating-point value that less than or equal to the argument and is equal to a mathematical integer.





*val2* **Integer** The second argument.

---

### Return Value

---

**Integer** The larger of *val1* and *val2*.

---

## long min(long val1, long val2)

Returns the lesser of the two specified long values. If the specified values are equal, then the result is that same value.

---

### Input Parameters

---

*val1* **Integer** The first argument.

*val2* **Integer** The second argument.

---

### Return Value

---

**Integer** The lesser of *val1* and *val2*.

---

## double min(double val1, double val2)

Returns the lesser of the two specified double values. If the specified values are equal, then the result is that same value.

---

### Input Parameters

---

*val1* **Integer** The first argument.

*val2* **Integer** The second argument.

---

### Return Value

---

**Integer** The lesser of *val1* and *val2*.



**Return Value**

---

**Integer** The value of pi to 15 decimal places.

---

**double pow(double base, double exponent)**

Returns  $\text{base}^{\text{exponent}}$  or the value of the base argument raised to the power of the exponent argument.

**Input Parameters**

---

*base* **Integer** The base.

*exponent* **Integer** The exponent.

**Return Value**

---

**Integer** The value of  $\text{base}^{\text{exponent}}$ .

---

**double radiansToDegrees(double angrad)**

Returns an approximately equivalent angle measured in degrees for the specified angle measured in radians.

**Input Parameters**

---

*angrad* **Integer** An angle in radians.

**Return Value**

---

**Integer** The measurement of the angle *angrad* in degrees.

---

**long round(double val)**

Returns the closest long integer to the specified double argument, with ties rounding up.

---

### Input Parameters

---

*val* **Integer** A floating-point value to be rounded to a long.

### Return Value

---

**Integer** The value of the argument rounded to the nearest long value.

---

## double round(double val, int scale)

Rounds the given value to the specified number of decimal places. The value is rounded using the `BigDecimal.ROUND_HALF_UP` method (rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant, in which case round up).

---

### Input Parameters

---

*val* **Integer** The value to round.

*scale* **Integer** The number of digits to the right of the decimal point.

### Return Value

---

**Integer** The rounded value.

---

## double round(double val, int scale, int roundingMethod)

Rounds the given value to the specified number of decimal places. The value is rounded using the given method which is any method defined in `java.math.BigDecimal`. `BigDecimal` values are:

- `ROUND_UP = 0`, Rounding mode to round away from zero.
- `ROUND_DOWN = 1`, Rounding mode to round towards zero.
- `ROUND_CEILING = 2`, Rounding mode to round towards positive infinity.
- `ROUND_FLOOR = 3`, Rounding mode to round towards negative infinity.
- `ROUND_HALF_UP = 4`, Rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant, in which case round up.

- `ROUND_HALF_DOWN = 5`, Rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant, in which case round down.
- `ROUND_HALF_EVEN = 6`, Rounding mode to round towards the "nearest neighbor" unless both neighbors are equidistant, in which case, round towards the even neighbor.
- `ROUND_UNNECESSARY = 7`, Rounding mode to assert that the requested operation has an exact result, hence no rounding is necessary.

### Input Parameters

---

<i>val</i>	<b>Integer</b> The value to round.
<i>scale</i>	<b>Integer</b> The number of digits to the right of the decimal point.
<i>roundingMethod</i>	<b>Integer</b> Rounding method as defined in <code>BigDecimal</code> .

### Return Value

---

**Integer** The rounded value.

---

## double sin(double val)

Returns the trigonometric sine of the specified angle.

### Input Parameters

---

<i>val</i>	<b>Integer</b> An angle in radians.
------------	-------------------------------------

### Return Value

---

**Integer** The sine of the argument.

---

## double sinh(double val)

Returns the hyperbolic sine of the specified double value.



## Summary of String Functions

webMethods Rules Development provides the following predefined string functions:

Function	Returns	Description
<code>concat(String str): String</code>	String	Appends the specified string to the end of this string.
<code>contains(String str): boolean</code>	Boolean	Indicates whether or not the specified string is contained within this string.
<code>endsWith(String suffix): boolean</code>	Boolean	Indicates whether or not this string ends with the specified suffix.
<code>equals(String anotherString): boolean</code>	Boolean	Indicates whether or not this string is equal to the specified string.
<code>equalsIgnoreCase(String anotherString): boolean</code>	Boolean	Indicates whether or not this string is equal to the specified string, ignoring case considerations.
<code>matches(String regex): boolean</code>	Boolean	Indicates whether or not this string matches the given regular expression.
<code>regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len): boolean</code>	Boolean	Indicates whether or not two string regions (substrings within the specified strings) are equal.
<code>replaceAll(String regex, String replacement): String</code>	String	Returns a string where each substring of this string that matches the given regular expression is replaced with the specified replacement value.
<code>replaceFirst(String regex, String replacement): String</code>	String	Returns a string where the first substring of this string



Function	Returns	Description
		that matches the given regular expression is replaced with the specified replacement value.
<code>startsWith(String prefix): boolean</code>	Boolean	Indicates whether or not this string starts with the specified prefix.
<code>substring(int beginIndex): String</code>	String	Returns a string that resides within this string.
<code>substring(int beginIndex, int endIndex): String</code>	String	Returns a string that resides within this string.
<code>toLowerCase(): String</code>	String	Returns a string with all of the characters of this string converted to lower case.
<code>toUpperCase(): String</code>	String	Returns a string with all of the characters of this string converted to upper case.
<code>trim(): String</code>	String	Returns a copy of this string, with leading and trailing empty characters removed.

## `concat(String str): String`

Appends the specified string to the end of this string.

### Input Parameters

*str*                      **String** The string that is appended to the end of this String.

### Return Value

**String** Returns a string that represents the concatenation of this object's characters followed by the string argument's characters.

---

## **contains(String str): boolean**

Indicates whether or not the specified string is contained within this string.

### **Input Parameters**

---

*str*                                      **String** The string to search for.

### **Return Value**

---

**Boolean** Returns `true` if this string contains the specified string, `false` otherwise.

---

## **endsWith(String suffix): boolean**

Indicates whether or not this string ends with the specified suffix.

### **Input Parameters**

---

*suffix*                                      **String** The suffix.

### **Return Value**

---

**Boolean** Returns `true` if the character sequence represented by the argument is a suffix of the character sequence represented by this object; `false` otherwise.

**Note:**     The result will be true if the argument is the empty string or is equal to this String object as determined by the `equals(String)` method.

---

## **equals(String anotherString): boolean**

Indicates whether or not this string is equal to the specified string. The result is true if and only if the argument is not null and is a string that represents the same sequence of characters as this string. Case (upper/lower) must match in order for the strings to be considered equal.

### Input Parameters

---

*anotherString*                      **String** The string to compare this string against.

### Return Value

---

**Boolean** Returns `true` if the argument is not null and it represents an equivalent string, `false` otherwise.

---

## equalsIgnoreCase(String anotherString): boolean

Indicates whether or not this string is equal to the specified string, ignoring case considerations. Two strings are considered equal ignoring case if they are of the same length and corresponding characters in the two strings are equal ignoring case.

### Input Parameters

---

*anotherString*                      **String** The string to compare this string against.

### Return Value

---

**Boolean** Returns `true` if the argument is not null and it represents an equivalent string ignoring case, `false` otherwise.

---

## matches(String regex): boolean

Indicates whether or not this string matches the given regular expression.

### Input Parameters

---

*regex*                                      **String** The regular expression to which this string is to be matched.

### Return Value

---

**Boolean** Returns `true` if this string matches the given regular expression, `false` otherwise.

---

## regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len): boolean

Indicates whether or not two string regions (substrings within the specified strings) are equal.

### Input Parameters

---

<i>ignoreCase</i>	<b>Boolean</b> Indicates whether or not case should be ignored when comparing characters.
<i>toffset</i>	<b>Integer</b> The starting offset of the subregion in this string.
<i>other</i>	<b>String</b> The string argument.
<i>ooffset</i>	<b>Integer</b> The starting offset of the subregion in the string argument.
<i>len</i>	<b>Integer</b> The number of characters to compare.

### Return Value

---

**Boolean** Returns `true` if the two string regions are equal, `false` otherwise.

---

## replaceAll(String regex, String replacement): String

Returns a string where each substring of this string that matches the given regular expression is replaced with the specified replacement value. If no match is found, then the original string is returned.

### Input Parameters

---

<i>regex</i>	<b>String</b> The regular expression to which this string is to be matched.
<i>replacement</i>	<b>String</b> The string to be substituted for each match.

### Return Value

---

**String** The resulting string.

## replaceFirst(String regex, String replacement): String

Returns a string where the first substring of this string that matches the given regular expression is replaced with the specified replacement value. If no match is found, then the original string is returned.

### Input Parameters

---

<i>regex</i>	<b>String</b> The regular expression to which this string is to be matched.
<i>replacement</i>	<b>String</b> The string to be substituted for each match.

### Return Value

---

**String** The resulting string.

## startsWith(String prefix): boolean

Indicates whether or not this string starts with the specified prefix.

### Input Parameters

---

<i>prefix</i>	<b>String</b> The prefix.
---------------	---------------------------

### Return Value

---

**Boolean** Returns `true` if the character sequence represented by the argument is a prefix of the character sequence represented by this string; `false` otherwise.

**Note:** The result will be `true` if the argument is the empty string or is equal to this string object as determined by the `equals(String)` method.

---

## substring(int beginIndex): String

Returns a string that resides within this string. The returned substring begins with the character at the specified index and extends to the end of this string.

---

### Input Parameters

*beginIndex*                      **Integer** The beginning index, inclusive.

---

### Return Value

**String** The specified substring.

---

## substring(int beginIndex, int endIndex): String

Returns a string that resides within this string. The returned substring begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`. Thus the length of the substring is `endIndex - beginIndex`.

---

### Input Parameters

*beginIndex*                      **Integer** The beginning index, inclusive.

*endIndex*                         **Integer** The ending index, exclusive.

---

### Return Value

**String** The specified substring.

---

## toLowerCase(): String

Returns a string with all of the characters of this string converted to lower case.

---

### Input Parameters

None.

**Return Value**

---

**String** The string converted to lower case.

---

**toLowerCase(): String**

Returns a string with all of the characters of this string converted to lower case.

**Input Parameters**

---

None.

**Return Value**

---

**String** The string converted to upper case.

---

**trim(): String**

Returns a copy of this string, with leading and trailing empty characters removed.

**Input Parameters**

---

None.

**Return Value**

---

**String** A copy of this string with leading and trailing white space removed, or this string if it has no leading or trailing white space.





---

## 2 Rules-Related Event Types Overview

---

■ Summary of Rules-Related Event Types .....	74
■ Decision Entity Changed .....	75
■ Hot Deployment Started .....	75
■ Project Deleted .....	76
■ Project Imported .....	76
■ Project Exported .....	77
■ Project Deployed .....	77
■ Project Undeployed .....	78

webMethods Rules Development provides a set of predefined event types that allow you to monitor rules-related events on the Integration Server or on the My webMethods Server. For detailed information about how to work with rules-related event types, see *webMethods BPM Rules Development Help*. For the existing event types, see "[Summary of Rules-Related Event Types](#)" on page 74.

## Summary of Rules-Related Event Types

webMethods Rules Development provides the following predefined event types:

Event Type	Emitted On	Description
<a href="#">Decision Entity Changed</a>	My webMethods Server	Is triggered when a difference is detected between two equally named decision entities.
<a href="#">Hot Deployment Started</a>	My webMethods Server	Is triggered when you start to hot deploy a rule project from the My webMethods Server to the Integration Server.
<a href="#">Project Deleted</a>	My webMethods Server	Is triggered when a rule project is deleted from the My webMethods Server.
<a href="#">Project Imported</a>	My webMethods Server	Is triggered when a rule project is imported to the My webMethods Server.
<a href="#">Project Exported</a>	My webMethods Server	Is triggered when a rule project is exported from the My webMethods Server to the file system.
<a href="#">Project Deployed</a>	Integration Server	Is triggered when a rule project is deployed on the Integration Server.
<a href="#">Project Undeployed</a>	Integration Server	Is triggered when a rule project is undeployed from the Integration Server.

## Decision Entity Changed

This event type is triggered on the My webMethods Server when a difference is detected between two equally named decision entities. The event instance on the Universal Messaging Server contains the following entries:

- dateTime
- projectName
- sessionId
- host
- port
- user
- decisionEntityType
- decisionEntityName
- correlationId
- changesType (1=rule added, 2=rule deleted, 3=rule changed)
- cellType (1=condition cell, 2=result cell)
- row
- columnName
- parameterElementName
- old Cell Value
- new Cell Value

---

## Hot Deployment Started

This event type is triggered on the My webMethods Server when you start to hot deploy a rule project. The event instance on the Universal Messaging Server contains the following entries:

- dateTime
- projectName
- sessionId
- host
- port
- user

- deployedISHost (the Integration Server host you configured on the My webMethods Server for hot deployment)
- deployedISPort (the Integration Server port you configured on the My webMethods Server for hot deployment)
- correlationId
- projectVersion

---

## Project Deleted

This event type is triggered when a rule project is deleted from the My webMethods Server. The event instance on the Universal Messaging Server contains the following entries:

- dateTime
- projectName
- host
- port
- user

---

## Project Imported

This event type is triggered when a rule project is imported to the My webMethods Server. The event instance on the Universal Messaging Server contains the following entries:

- dateTime
- projectName
- sessionID (My webMethods Server)
- host (My webMethods Server)
- port (My webMethods Server)
- user (My webMethods Server userid used to connect to the My webMethods Server)
- decisionEntities (all decision tables and event rules that belong to this rule project)
- projectVersion
- correlationId (an identifier to connect between Project Imported and Decision Entity Changed events)

## Project Exported

This event type is triggered when a rule project is exported from the My webMethods Server to the file system. The event instance on the Universal Messaging Server contains the following entries:

- dateTime
- projectName
- sessionID (My webMethods Server)
- host (My webMethods Server)
- port (My webMethods Server)
- user (My webMethods Server userid used to connect to the My webMethods Server)
- decisionEntities (all decision tables and event rules that belong to this rule project)
- projectVersion

---

## Project Deployed

This event type is triggered when a rule project is deployed on the Integration Server. The event instance on the Universal Messaging Server contains the following entries:

- dateTime
- projectName
- sessionId
- host
- port
- user
- deployedWith
- projectCreatorApp
- correlationId
- projectVersion
- deployedBy

## Project Undeployed

This event type is triggered when a rule project is undeployed from the Integration Server. The event instance on the Universal Messaging Server contains the following entries:

- dateTime
- projectName
- sessionId
- host
- port
- user
- correlationId
- projectVersion

# 3 Services Overview

■ Summary of WmBusinessRules Built-in Services .....	81
■ pub.businessrules.client.invoke .....	81
■ Summary of REST Services .....	83
■ GET <base URL>/projects .....	86
■ GET <base URL>/project/%ruleProjectName% .....	86
■ PUT <base URL>/project/%ruleProjectName%/lock .....	87
■ PUT <base URL>/project/%ruleProjectName%/unlock .....	87
■ PUT <base URL>/project/%ruleProjectName%/unlockall .....	87
■ PUT <base URL>/project/%ruleProjectName%/decisiontable/%decisionTableName% .....	88
■ PUT <base URL>/project/%ruleProjectName%/decisiontable/%decisionTableName%/lock .....	88
■ PUT <base URL>/project/%ruleProjectName%/decisiontable/%decisionTableName%/unlock .....	89
■ PUT <base URL>/project/%ruleProjectName%/deploy .....	89
■ GET <base URL>/project/%ruleProjectName% .....	89
■ PUT <base URL>/project/%ruleProjectName% .....	90
■ DELETE <base URL>/project/%ruleProjectName% .....	90
■ GET <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName% .....	91
■ PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName% .....	91
■ GET <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/condition_nodes/%conditionNodeId% .....	92
■ PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/condition_nodes/%conditionNodeId% .....	92
■ GET <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/lines/%conditionLinkId% .....	93
■ PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/lines/%conditionLinkId% .....	94
■ GET <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/result_nodes/%resultNodeId% .....	95

---

■ PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/result_nodes/%resultNodeId% .....	95
■ GET <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/results/%resultId% .....	96
■ PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/results/%resultId% .....	97
■ PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/lock .....	97
■ PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/unlock .....	98
■ GET <base URL>/projects/%ruleProjectName%/decision_tables/%decisionTableName% .....	99



webMethods Rules Development provides a set of services.

Two categories of services exist:

- ["Summary of WmBusinessRules Built-in Services" on page 81.](#)
- ["Summary of REST Services" on page 83.](#)

---

## Summary of WmBusinessRules Built-in Services

webMethods Rules Development provides the following built-in service in the WmBusinessRules package:

Element	Package and Description
<a href="#">pub.businessrules.client:invoke</a>	WmBusinessRules. Executes a rule set or decision table and returns the results of execution.

---

### pub.businessrules.client:invoke

WmBusinessRules. Executes a rule set or decision table and returns the results of execution.

#### Input Parameters

---

*Project Name*

**String** Name of the rule project that contains the rule set or decision table that you want to execute.

**Note:** The rule project that contains the rule set or decision table that you want to execute must be deployed on the Integration Server.

*Invocation Target*

**String** Name of the rule set or decision table that you want to execute.

Specify one of the following:

Specify...	To...
RS/[RuleSetName]	Define a rule set.
DT/[DecisionTableName]	Define a decision table.

	D3/[ <i>DecisionTreeName</i> ]	Define a decision tree.
<i>Create Missing Inputs</i>	<p><b>Boolean</b> Optional. Creates an empty input parameter without input values if no input parameter is specified for the rule set or decision table. Set to:</p> <ul style="list-style-type: none"> <li>■ <code>true</code> if missing inputs should be created. This is the default.</li> <li>■ <code>false</code> if missing inputs should not be created.</li> </ul>	

*Inputs* **Document** Defines the input parameters for the rule set or decision table.

**Note:** The generic invoke service `pub.businessrules.client.genericInvoke` that is included in the `WmBusinessRules` package demonstrates how to specify inputs.

<u>Key</u>	<u>Description</u>
[ <i>Input Parameter Name</i> ]	<b>Document</b> Defines the input parameter elements and their values.
<i>Fact IData</i>	Parameter instance at runtime.

*Desired Outputs* **Document List** Optional. Defines a list of output parameters that you want the service execution to return.

**Note:** If no output parameters are specified, all output parameters of the *Invocation Target* are returned. If a specified output parameter does not exist, the specified output parameter is ignored.

## Output Parameters

*Outputs* **Document** Defines the output parameters returned by the rule set or decision table.

<u>Key</u>	<u>Description</u>
------------	--------------------

<i>[Output Parameter Name]</i>	<b>Document</b> Defines the output parameter elements and their values.
<i>Fact IData</i>	Parameter instance at runtime.

### Usage Note

Dragging and dropping a rule set or decision entity from the Rules Explorer view to a flow service inserts an invoke step that calls the `pub.businessrules.client.invoke` into the flow service. Software AG Designer specifies the *Project Name* and *Invocation Target* values automatically based on the rule set or decision entity dragged into the flow service. Software AG Designer creates the input parameters under *Service In > Inputs* and the outputs parameters under *Service Out > Outputs* automatically based on the input and output parameters of the rule set or decision table dragged into the flow service.

## Summary of REST Services

webMethods Business Rules provides predefined REST services. For detailed information on available REST services, see the WADL URL `<host>:<port>/wm_rma/rest/application.wadl`.

REST services using base URL `<host>:<port>/wm_rma/rest/content`:

REST Service	Description
GET <code>&lt;base URL&gt;/projects</code>	Returns a list of rule projects that are currently available on My webMethods Server.
GET <code>&lt;base URL&gt;/project/%ruleProjectName%</code>	Returns the content and metadata for a given rule project.
PUT <code>&lt;base URL&gt;/project/%ruleProjectName%/lock</code>	Locks the given rule project.
PUT <code>&lt;base URL&gt;/project/%ruleProjectName%/unlock</code>	Unlocks the given rule project.
PUT <code>&lt;base URL&gt;/project/%ruleProjectName%/unlockall</code>	Unlocks the given rule project and all decision entities contained in it.

REST Service	Description
GET <base URL>/project/ %ruleProjectName%/decisiontable/ %decisionTableName%	This service has been deprecated and replaced with <a href="#">GET &lt;base URL&gt;/projects/%ruleProjectName%/decision_tables/%decisionTableName%</a> , using base URL <host>:<port>/wm_rma/rest.
PUT <base URL>/project/%ruleProjectName%/decisiontable/%decisionTableName%	Writes the changes for the given decision table to the My webMethods Server.
PUT <base URL>/project/%ruleProjectName%/decisiontable/%decisionTableName%/lock	Locks the given decision table.
PUT <base URL>/project/%ruleProjectName%/decisiontable/%decisionTableName%/unlock	Unlocks the given decision table.
PUT <base URL>/project/%ruleProjectName%/deploy	Hot deploys the given rule project to an Integration Server that was configured on the My webMethods Server.

REST services using base URL <host>:<port>/wm\_rma/rest/raw:

REST Service	Description
GET <base URL>/project/%ruleProjectName%	Retrieves the rule project from the My webMethods Server.
PUT <base URL>/project/%ruleProjectName%	Stores the given rule project on the My webMethods Server.
DELETE <base URL>/project/ %ruleProjectName%	Deletes the given rule project from the My webMethods Server.

REST services using base URL <host>:<port>/wm\_rma/rest:

REST Service	Description
GET <base URL>/projects/%ruleProjectName %/decision_trees/%decisionTreeName%	Retrieves the given decision tree.

REST Service	Description
PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%	Updates the specified decision tree in the given project with the provided decision tree JSON.
GET <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/condition_nodes/%conditionNodeId%	Retrieves a condition node with the specified conditionNodeId in the given decision tree and rule project.
PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/condition_nodes/%conditionNodeId%	Updates a condition node with the specified ConditionNode JSON for the conditionNodeId in the given decision tree and rule project.
GET <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/lines/%conditionLinkId%	Retrieves a condition link with the specified conditionLinkId in the given decision tree and rule project.
PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/lines/%conditionLinkId%	Updates a condition link with the specified ConditionLink JSON for the conditionLinkId in the given decision tree and rule project.
GET <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/result_nodes/%resultNodeId%	Retrieves a result node with the specified resultNodeId in the given decision tree and rule project.
PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/result_nodes/%resultNodeId%	Updates the result node with the specified ResultNode JSON for the resultNodeId in the given decision tree and rule project.
GET <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/results/%resultId%	Retrieves a result with the specified resultId in the given decision tree and rule project.
PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/results/%resultId%	Updates a result with the specified Result JSON for the resultId in the given decision tree and rule project.
PUT <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%/lock	Locks a decision tree so that it can be modified exclusively by the user applying the lock.

REST Service	Description
PUT <base URL>/projects/%ruleProjectName %/decision_trees/%decisionTreeName%/unlock	Unlocks a decision tree so that it can be locked by another user for exclusive modification.
GET <base URL>/projects/%ruleProjectName %/decision_tables/%decisionTableName%	Returns the specified decision table from the specified rule project in JSON.

## GET <base URL>/projects

Returns a list of rule projects that are currently available on My webMethods Server.

### Input Parameters

None.

### Output Parameters

**JSON List** List of rule project names.

## GET <base URL>/project/%ruleProjectName%

Returns the content and metadata for a given rule project.

### Input Parameters

*ruleProjectName*

**String** Name of the rule project the content and metadata is required for.

### Output Parameters

**Structured JSON** Names of decision tables, event rules, data models, event models, rule sets, actions, rule project name and version.

---

## PUT <base URL>/project/%ruleProjectName%/lock

Locks the given rule project.

### Input Parameters

---

*ruleProjectName*                      **String** Name of the rule project to be locked.

### Output Parameters

---

None.

---

## PUT <base URL>/project/%ruleProjectName%/unlock

Unlocks the given rule project.

### Input Parameters

---

*ruleProjectName*                      **String** Name of the rule project to be unlocked.

### Output Parameters

---

None.

---

## PUT <base URL>/project/%ruleProjectName%/unlockall

Unlocks the given rule project and all decision entities contained in it.

### Input Parameters

---

*ruleProjectName*                      **String** Name of the rule project to be unlocked.

### Output Parameters

---

None.

---

## PUT <base URL>/project/%ruleProjectName%/decisiontable/ %decisionTableName%

Writes the changes for the given decision table to the My webMethods Server.

### Input Parameters

---

<i>ruleProjectName</i>	<b>String</b> Name of the rule project the decision table is part of.
<i>decisionTableName</i>	<b>String</b> Name of the decision table to be saved.
<i>body</i>	<b>Structured JSON</b> Changed decision table contents.

### Output Parameters

---

None.

---

## PUT <base URL>/project/%ruleProjectName%/decisiontable/ %decisionTableName%/lock

Locks the given decision table.

### Input Parameters

---

<i>ruleProjectName</i>	<b>String</b> Name of the rule project the decision table is part of.
<i>decisionTableName</i>	<b>String</b> Name of the decision table to be locked.

### Output Parameters

---

None.



---

## PUT <base URL>/project/%ruleProjectName%/decisiontable/ %decisionTableName%/unlock

Unlocks the given decision table.

### Input Parameters

---

*ruleProjectName*                      **String** Name of the rule project the decision table is part of.

*decisionTableName*                      **String** Name of the decision table to be unlocked.

### Output Parameters

---

None.

---

## PUT <base URL>/project/%ruleProjectName%/deploy

Hot deploys the given rule project to an Integration Server that was configured on the My webMethods Server. For more information, see *Working with Business Rules in My webMethods*.

### Input Parameters

---

*ruleProjectName*                      **String** Name of the rule project to be deployed.

### Output Parameters

---

None.

---

## GET <base URL>/project/%ruleProjectName%

Retrieves the rule project from the My webMethods Server.

**Input Parameters**

---

*ruleProjectName*                      **String** Name of the rule project the archive is required for.

**Output Parameters**

---

**Binary** Rule project archive (in jar/zip format).

---

**PUT <base URL>/project/%ruleProjectName%**

Stores the given rule project on the My webMethods Server. If the rule project already exists on this server, it will be replaced.

**Input Parameters**

---

*ruleProjectName*                      **String** Name of the rule project to be stored.

*body*                                      **Binary** Rule project archive (in jar/zip format).

**Output Parameters**

---

None.

---

**DELETE <base URL>/project/%ruleProjectName%**

Deletes the given rule project from the My webMethods Server.

**Input Parameters**

---

*ruleProjectName*                      **String** Name of the rule project to be deleted from the My webMethods Server.

**Output Parameters**

---

None.

---

## GET <base URL>/projects/%ruleProjectName%/decision\_trees/%decisionTreeName%

Retrieves the given decision tree.

### Input Parameters

---

<i>ruleProjectName</i>	<b>String</b> Name of the rule project the decision tree is part of.
<i>decisionTreeName</i>	<b>String</b> Name of the decision tree to be returned.

### Output Parameters

---

**Structured JSON** Decision tree contents.

---

## PUT <base URL>/projects/%ruleProjectName%/decision\_trees/%decisionTreeName%

Updates the specified decision tree in the given project with the provided decision tree JSON.

### Input Parameters

---

<i>authorization</i>	<b>String</b> Base64 encoded HTTP "authorization" header value. Example: username:password.
<i>ruleProjectName</i>	<b>String</b> The name of the rule project containing the specified decision tree.
<i>decisionTreeName</i>	<b>String</b> The name of the decision tree within the specified rule project.

### Output Parameters

---

**Structured JSON** Decision tree contents.

---

## GET <base URL>/projects/%ruleProjectName%/decision\_trees/%decisionTreeName%/condition\_nodes/%conditionNodeId%

Retrieves a condition node with the specified conditionNodeId in the given decision tree and rule project.

### Input Parameters

---

<i>authorization</i>	<b>String</b> Base64 encoded HTTP "authorization" header value. Example: username:password.
<i>ruleProjectName</i>	<b>String</b> The name of the rule project containing the specified decision tree.
<i>decisionTreeName</i>	<b>String</b> The name of the decision tree within the specified rule project.
<i>conditionNodeId</i>	<b>String</b> The unique ID of the condition node within the specified decision tree and rule project.
<i>deep</i>	<b>Boolean</b> (Optional) Indicates whether or not to get the specified condition with any/all of its descendent links and nodes. Default value is true.

### Output Parameters

---

**Structured JSON** Decision tree contents.

---

## PUT <base URL>/projects/%ruleProjectName%/decision\_trees/%decisionTreeName%/condition\_nodes/%conditionNodeId%

Updates a condition node with the specified ConditionNode JSON for the conditionNodeId in the given decision tree and rule project.

---

### Input Parameters

---

<i>authorization</i>	<b>String</b> Base64 encoded HTTP "authorization" header value. Example: username:password.
<i>ruleProjectName</i>	<b>String</b> The name of the rule project containing the specified decision tree.
<i>decisionTreeName</i>	<b>String</b> The name of the decision tree within the specified rule project.
<i>conditionNodeId</i>	<b>String</b> The unique ID of the condition node within the specified decision tree and rule project.

### Output Parameters

---

**Structured JSON** Decision tree contents.

---

## GET <base URL>/projects/%ruleProjectName%/decision\_trees/%decisionTreeName%/lines/%conditionLinkId%

Retrieves a condition link with the specified conditionLinkId in the given decision tree and rule project.

### Input Parameters

---

<i>authorization</i>	<b>String</b> Base64 encoded HTTP "authorization" header value. Example: username:password.
<i>ruleProjectName</i>	<b>String</b> The name of the rule project containing the specified decision tree.
<i>decisionTreeName</i>	<b>String</b> The name of the decision tree within the specified rule project.
<i>conditionLinkId</i>	<b>String</b> The unique ID of the condition link within the specified decision tree and rule project.

*deep* **Boolean** (Optional) Indicates whether or not to get the specified condition with any/all of its descendent links and nodes. Default value is true.

---

### Output Parameters

**Structured JSON** Decision tree contents.

---

**PUT <base URL>/projects/%ruleProjectName%/decision\_trees/%decisionTreeName%/lines/%conditionLinkId%**

Updates a condition link with the specified ConditionLink JSON for the conditionLinkId in the given decision tree and rule project.

---

### Input Parameters

*authorization* **String** Base64 encoded HTTP "authorization" header value. Example: username:password.

*ruleProjectName* **String** The name of the rule project containing the specified decision tree.

*decisionTreeName* **String** The name of the decision tree within the specified rule project.

*conditionLinkId* **String** The unique ID of the condition link within the specified decision tree and rule project.

---

### Output Parameters

**Structured JSON** Decision tree contents.

---

## GET <base URL>/projects/%ruleProjectName%/decision\_trees/%decisionTreeName%/result\_nodes/%resultNodeId%

Retrieves a result node with the specified resultNodeId in the given decision tree and rule project.

### Input Parameters

---

<i>authorization</i>	<b>String</b> Base64 encoded HTTP "authorization" header value. Example: username:password.
<i>ruleProjectName</i>	<b>String</b> The name of the rule project containing the specified decision tree.
<i>decisionTreeName</i>	<b>String</b> The name of the decision tree within the specified rule project.
<i>resultNodeId</i>	<b>String</b> The unique ID of the result node within the specified decision tree and rule project.
<i>deep</i>	<b>Boolean</b> (Optional) Indicates whether or not to get the specified condition with any/all of its descendent links and nodes. Default value is true.

### Output Parameters

---

**Structured JSON** Decision tree contents.

---

## PUT <base URL>/projects/%ruleProjectName%/decision\_trees/%decisionTreeName%/result\_nodes/%resultNodeId%

Updates the result node with the specified ResultNode JSON for the resultNodeId in the given decision tree and rule project.

---

### Input Parameters

<i>authorization</i>	<b>String</b> Base64 encoded HTTP "authorization" header value. Example: username:password.
<i>ruleProjectName</i>	<b>String</b> The name of the rule project containing the specified decision tree.
<i>decisionTreeName</i>	<b>String</b> The name of the decision tree within the specified rule project.
<i>resultNodeId</i>	<b>String</b> The unique ID of the result node within the specified decision tree and rule project.

---

### Output Parameters

**Structured JSON** Decision tree contents.

---

## GET <base URL>/projects/%ruleProjectName%/decision\_trees/%decisionTreeName%/results/%resultId%

Retrieves a result node with the specified resultId in the given decision tree and rule project.

---

### Input Parameters

<i>authorization</i>	<b>String</b> Base64 encoded HTTP "authorization" header value. Example: username:password.
<i>ruleProjectName</i>	<b>String</b> The name of the rule project containing the specified decision tree.
<i>decisionTreeName</i>	<b>String</b> The name of the decision tree within the specified rule project.
<i>resultId</i>	<b>String</b> The unique ID of the result within the specified decision tree and rule project.



*deep* **Boolean** (Optional) Indicates whether or not to get the specified condition with any/all of its descendent links and nodes. Default value is true.

---

### Output Parameters

**Structured JSON** Decision tree contents.

---

## PUT <base URL>/projects/%ruleProjectName%/decision\_trees/%decisionTreeName%/results/%resultId%

Updates a result with the specified Result JSON for the resultId in the given decision tree and rule project.

---

### Input Parameters

*authorization* **String** Base64 encoded HTTP "authorization" header value. Example: username:password.

*ruleProjectName* **String** The name of the rule project containing the specified decision tree.

*decisionTreeName* **String** The name of the decision tree within the specified rule project.

*resultId* **String** The unique ID of the result within the specified decision tree and rule project.

---

### Output Parameters

**Structured JSON** Decision tree contents.

---

## PUT <base URL>/projects/%ruleProjectName%/decision\_trees/%decisionTreeName%/lock

Locks a decision tree so that it can be modified exclusively by the user applying the lock.

---

### Input Parameters

---

<i>authorization</i>	<b>String</b> Base64 encoded HTTP "authorization" header value. Example: username:password.
<i>ruleProjectName</i>	<b>String</b> The name of the rule project containing the specified decision tree.
<i>decisionTreeName</i>	<b>String</b> The name of the decision tree within the specified rule project.

### Output Parameters

---

None.

---

## PUT <base URL>/projects/%ruleProjectName%/decision\_trees/%decisionTreeName%/unlock

Unlocks a decision tree so that it can be locked by another user for exclusive modification.

### Input Parameters

---

<i>authorization</i>	<b>String</b> Base64 encoded HTTP "authorization" header value. Example: username:password.
<i>ruleProjectName</i>	<b>String</b> The name of the rule project containing the specified decision tree.
<i>decisionTreeName</i>	<b>String</b> The name of the decision tree within the specified rule project.

### Output Parameters

---

None.

---

## GET <base URL>/projects/%ruleProjectName%/decision\_tables/%decisionTableName%

Retrieves the given decision table in JSON.

### Input Parameters

---

<i>ruleProjectName</i>	<b>String</b> Name of the rule project the decision table is part of.
<i>decisionTableName</i>	<b>String</b> Name of the decision table to be returned.

### Output Parameters

---

**Structured JSON** Decision table contents.



# 4 Technical Details for Preconfigured Verification Services

webMethods Rules Development allows you to verify decision table columns on the basis of preconfigured REST services. For detailed information about how to work with preconfigured verification services, see *webMethods BPM Rules Development Help* and *Working with Business Rules in My webMethods*.

The REST service must meet the following requirements:

- **Request Type** The REST service must be of request type POST, and it must be available on the following URL: <backendserver>:<port>/<base path>/<service URI>.
- **HTTP Header** The HTTP header must be of Content-Type application/json.
- **Authentication** The REST service must support Basic Authentication or None.
- **Input** Structured JSON describing the decision table column.
- **Required Output** JSON describing errors and warnings, and using the following structure:

```
{
  "messages": [
    {
      "description": "<Enter text for error message>",
      "row": "<Enter row number, starting with 1>",
      "severity": "<Enter ERROR or enter WARNING>"
    }
  ]
}
```

Example:

```
{
  "messages": [
    {
      "description": "Invalid zip code",
      "row": "3",
      "severity": "ERROR"
    }
  ]
}
```



# 5 Technical Details for Preconfigured Data Provider Services

To avoid unexpected or misspelled cell entries in a decision table, Business Rules supports external dynamic data provider REST services. These services provide options to be selected in a drop-down box. The user can define one data provider REST service per column. If the user edits a cell with data provider service configuration, a background REST service call retrieves values and descriptions for a drop-down box. The data provider server can be configured in My webMethods. For more information, see *Working with Business Rules in My webMethods*. The actual data provider REST service can be assigned to a decision table with the Rules Development feature of Software AG Designer. For more information, see *webMethods BPM Rules Development Help*.

The REST service must meet the following requirements:

- The REST service must return the following two parameters:
  - **mode** An enumeration that sets the behavior of the input element: `STRICT`: Only values from the service response are available in the drop-down box and no free text input is allowed.
  - **options** An array of value-description pairs that are selectable in the drop-down box where `value` is the actual value put into the decision table cell and `description` (optional) is the text that shows up in the drop-down box and represents the associated value (if no description is given, the value will be shown in the drop-down box).
- The REST service must use the following structure:

```
{
  "title": "Data Provider Service Schema",
  "type": "object",
  "properties": {
    "mode": {
      "type": "string",
      "enum": [ "STRICT", "OPTIONAL" ]
    },
    "options": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "description": {
            "type": "string"
          },
          "value": {
            "type": "string"
          }
        }
      },
      "required": [ "value" ]
    }
  }
}
```

```
"required": [ "mode", "options" ]
}
```

**Example:**

```
{
  mode: "STRICT",
  options: [ { description: "Alice", value: "10" },
             { description: "Bob", value: "20" },
             { description: "Carol", value: "30" }
            ]
}
```

- If you specify date values, the following (ISO 8601) format is required:  
<DATE>T<TIME><OFFSET>, where DATE is yyyy-mm-dd, TIME is hh:mm:ss, and OFFSET determines the timezone offset (Z for UTC, +hh:mm for hour offset).

**Example:**

```
2016-10-04T09:11:00Z
2016-10-04T09:11:00+03:00
```



# 6 Administering Business Rules with Command Central

---

■ Commands that Business Rules Supports .....	106
■ Configuration Types that Business Rules Supports .....	108
■ Run-Time Monitoring Statuses for Business Rules .....	109

You use the Command Central web user interface and command line interface to manage Business Rules. The Business Rules run-time component is a layered product of My webMethods Server.

You can use Command Central to manage the following operations for Business Rules:

- Monitor run-time statuses for a Business Rules instance.

For information about monitoring a product instance, see the topic about monitoring KPIs in *Software AG Command Central Help*. For information about the Business Rules run-time statuses, see [Run-Time Monitoring Statuses for Business Rules](#).

- Configure an endpoint connection from My webMethods Server to Integration Server for hot deployment of Business Rules projects.

You can configure endpoint connections to multiple Integration Servers. For more information about the fields and values to specify when configuring an endpoint connection to Integration Server for hot deployment of Business Rules projects, see *Working with Business Rules in My webMethods*.

- Configure a business verification endpoint connection to a REST service that verifies user data in Business Rules decision tables.

You can configure a single connection to a business verification server. A *business verification server* is any server that provides REST services, including Integration Server. For more information about the fields and values to specify when editing a business verification endpoint connection, see *Working with Business Rules in My webMethods*.

- Configure a data provider endpoint connection to a REST service that provides data to be used in Business Rules decision tables.

You can configure a single connection to a data provider server. A *data provider server* is any server that provides REST services, including Integration Server. For more information about the fields and values to specify when editing a data provider endpoint connection, see *Working with Business Rules in My webMethods*.

- Configure whether users, groups and roles are displayed in the Principal Picker controls.

For information about the configuration commands that Business Rules supports, see [Commands that Business Rules Supports](#).

## Commands that Business Rules Supports

---

Business Rules supports the Platform Manager commands listed in the following table. The table lists where you can find information about each command.

Commands	For more information, see...
<pre>sagcc get configuration data</pre>	<p>For general information about the command, see the topic about administrating commands in <i>Software AG Command Central Help</i>.</p> <p>For information about the configuration types that Business Rules supports, see <a href="#">Configuration Types that Business Rules Supports</a>.</p>
<pre>sagcc update configuration data</pre>	<p>For general information about the command, see the topic about administrating commands in <i>Software AG Command Central Help</i>.</p> <p>For information about the configuration types that Business Rules supports, see <a href="#">Configuration Types that Business Rules Supports</a>.</p>
<pre>sagcc get configuration instances</pre>	<p>For general information about the command, see the topic about administrating commands in <i>Software AG Command Central Help</i>.</p> <p>For information about the configuration types that Business Rules supports, see <a href="#">Configuration Types that Business Rules Supports</a>.</p>
<pre>sagcc list configuration instances</pre>	<p>For general information about the command, see the topic about administrating commands in <i>Software AG Command Central Help</i>.</p> <p>For information about the configuration types that Business Rules supports, see <a href="#">Configuration Types that Business Rules Supports</a>.</p>
<pre>sagcc get configuration types</pre>	<p>For general information about the command, see the topic about administrating commands in <i>Software AG Command Central Help</i>.</p> <p>For information about the configuration types that Business Rules supports, see <a href="#">Configuration Types that Business Rules Supports</a>.</p>

Commands	For more information, see...
<code>sagcc list configuration types</code>	<p>For general information about the command, see the topic about administrating commands in <i>Software AG Command Central Help</i>.</p> <p>For information about the configuration types that Business Rules supports, see <a href="#">Configuration Types that Business Rules Supports</a>.</p>
<code>sagcc get monitoring</code>	<p>For general information about the command, see the topic about administrating commands in <i>Software AG Command Central Help</i>.</p> <p>For information about the configuration types that Business Rules supports, see <a href="#">Configuration Types that Business Rules Supports</a>.</p>
<code>sagcc exec lifecycle components OSGI-MWS_default start</code>	<p>For general information about the command, see the topic about administrating commands in <i>Software AG Command Central Help</i>.</p>
<code>sagcc exec lifecycle components OSGI-MWS_default stop</code>	<p>For general information about the command, see the topic about administrating commands in <i>Software AG Command Central Help</i>.</p>

## Configuration Types that Business Rules Supports

The Business Rules run-time component supports the following configuration types.

Configuration Type	Use to configure...
COMMON-COMPONENT-ENDPOINTS	A connection from My webMethods Server to multiple Integration Servers to execute hot deployment of Business Rules projects.
BRMS_REST_ENDPOINT	Base URL and optional authentication data of a REST server that verifies user data in Business Rules decision tables. Base URL is used by My webMethods Server whenever a verification service is invoked from a decision table. Trailing

Configuration Type	Use to configure...
	part of the URL is specified in Software AG Designer at the decision table column when configuring the verification service.
BRMS_REST_ENDPOINT _DATA_PROVIDER	Base URL and optional authentication data of a REST server that verifies user data in Business Rules decision tables. Base URL is used by My webMethods Server whenever a data provider service is invoked from a decision table. Trailing part of the URL is specified in Software AG Designer at the decision table column when configuring the data provider service.
PRINCIPAL_TYPES	Whether user, groups and roles are displayed in the Principal Picker.

## Run-Time Monitoring Statuses for Business Rules

The following table lists the run-time statuses that the Business Rules run-time component can return in response to the `sagcc get monitoring runtimestatus` and `sagcc get monitoring state` commands, along with the meaning of each run-time status.

Run-time Status	Meaning
ONLINE	The Business Rules instance is running.
STOPPED	The Business Rules instance is stopped.