webMethods Business Rules Reference

Version 10.7

October 2020

WEBMETHODS

This document applies to webMethods Business Rules 10.7 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.

Document ID: RULES-REF-107-20201015

Table of Contents

About this Guide **5**
- Document Conventions 6
- Online Information and Support 7
- Data Protection 8

1 Functions **9**
- About 10
- Summary of Conversion Functions 10
- Summary of Date Functions 15
- Summary of List and Range Functions 25
- Summary of Math Functions 39
- Summary of String Functions 51

2 Rules-Related Event Types **59**
- About 60
- Summary of Rules-Related Event Types 60
- Publishing Business Rules Events 61

3 Working with Business Rules Auditing **63**
- About 64
- Summary of Business Rules Auditing Information 64
- Writing Business Rules Auditing Information to the Database 68

4 Services **69**
- About 70
- Summary of WmBusinessRules Built-in Services 70
- Summary of REST Services on My webMethods Server 72
- Summary of REST Services on Integration Server 85

5 Technical Details for Preconfigured Verification Services **89**

6 Technical Details for Preconfigured Data Provider Services **91**

7 Using Docker for Business Rules **93**

8 Administering Business Rules with Command Central **95**
- About 96
- Commands that Business Rules Supports 96
- Configuration Types that Business Rules Supports 98
- Run-Time Monitoring Statuses for Business Rules 100

About this Guide

- Document Conventions 6
- Online Information and Support 7
- Data Protection 8

webMethods Business Rules Reference provides information on functions, event types, and services that you can use with Business Rules. It also explains how to work with Business Rules auditing, how to use Docker for Business Rules, and how to administer Business Rules with Command Central.

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

- "Functions" on page 9.
- "Rules-Related Event Types" on page 59.
- "Working with Business Rules Auditing" on page 63.
- "Services" on page 69.
- "Technical Details for Preconfigured Verification Services" on page 89.
- "Technical Details for Preconfigured Data Provider Services" on page 91.
- "Using Docker for Business Rules" on page 93.
- "Administering Business Rules with Command Central" on page 95.

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
Bold	Identifies elements on a screen.
Narrowfont	Identifies service names and locations in the format *folder.subfolder.service*, APIs, Java classes, methods, properties.
Italic	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.

Convention	Description
|	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 and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at http://documentation.softwareag.com.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at https://empower.softwareag.com/.

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.

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at https://empower.softwareag.com/public_directory.aspx and give us a call.

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.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

1 Functions

- About 10
- Summary of Conversion Functions 10
- Summary of Date Functions 15
- Summary of List and Range Functions 25
- Summary of Math Functions 39
- Summary of String Functions 51

About

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 10.
- "Summary of Date Functions" on page 15.
- "Summary of List and Range Functions" on page 25.
- "Summary of Math Functions" on page 39.
- "Summary of String Functions" on page 51.

Summary of Conversion Functions

webMethods Rules Development provides predefined conversion functions as listed in the following table:

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

Function	Returns	Description
toString()	String Object	Returns a string object representing the value of this boolean.
toString()	String	Returns a string representing the value of this date.
toString(String dateFormat)	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 predefined date functions as listed in the following table:

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

Function	Returns	Description
int century()	Integer	Returns the century for this date.
int compareDates(Date date1, Date date2)	Integer	Compares one date against another date.
Date date()	Date	Returns the current date from the server where the decision entity is invoked.
Date dateMinusDays(int days)	Date	Returns a copy of this date minus the specified number of days.
Date datePlusDays(int days)	Date	Returns a copy of this date plus the specified number of days.
long dateToInt()	Integer	Returns the number of milliseconds since the standard base time.
int dayOfMonth()	Integer	Returns the day of the month for this date.
int dayOfWeek()	Integer	Returns the day of the week for this date.
int dayOfYear()	Integer	Returns the day of the year for this date.
int daysInMonth(int month, int year)	Integer	Returns the number of days in the specified month in the given year.
long diffInDays(Date anotherDate)	Integer	Returns the mathematical difference in days between this date and the specified date ignoring the times of both dates.
long diffInMonths(Date anotherDate)	Integer	Returns the mathematical difference in months between this date and the specified date ignoring the times of both dates.
long diffInYears(Date anotherDate)	Integer	Returns the mathematical difference in years between this date and the specified date ignoring the times of both dates.
Date intToDate(long millis)	Date	Returns a date that represents the specified number of milliseconds since the standard base time.
isLeapYear(int year)	Boolean	Indicates whether or not the specified year is a leap year.
int month()	Integer	Returns the numeric month for this date.
String time()	String	Returns the string representation of the time this date as Hours:Minutes:Seconds.Milliseconds.

Function	Returns	Description
boolean verifyDate(int year, int month, int day)	Boolean	Verifies that the date represented by the specified year, month and day is valid.
boolean verifyIntDate(long millis)	Boolean	Verifies that the specified number of milliseconds is a valid date representation.
boolean verifyMonth(int month)	Boolean	Verifies that the specified numeric month is valid.
boolean verifyYear(int year)	Boolean	Verifies that the specified numeric year is valid.
int year()	Integer	Returns the numeric 4-digit year for this date.
Date ymdToDate(int year, int month, int day)	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.

Date dateMinusDays(int days)

Returns a copy of this date minus the specified number of days.

Input Parameters

days	**Integer** The amount of days to subtract, may be negative.

Return Value

Date The new date minus the specified days.

Date datePlusDays(int days)

Returns a copy of this date plus the specified number of days.

Input Parameters

days	**Integer** The amount of days to add, may be negative.

Return Value

Date The new date plus the specified days.

long dateToInt()

Returns the number of milliseconds since the standard base time known as "the epoch", namely January 1, 1970, 00:00:00 GMT represented by this date.

Input Parameters

None.

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.

Input Parameters

millis	**Integer** The number of milliseconds since January 1, 1970, 00:00:00 GMT.

Return Value

Date A date that represents the specified number of milliseconds since January 1, 1970, 00:00:00 GMT.

isLeapYear(int year)

Indicates whether or not the specified year is a leap year.

Input Parameters

year	**Integer** The year to test the leap for.

Return Value

Boolean Returns `true` if the specified year is a leap year, `false` otherwise.

int month()

Returns the numeric month for this date (1 = January, 2 = February, ..., 12 = December).

Input Parameters

None.

Return Value

Integer The numeric month for this date.

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

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

Input Parameters

millis	**Integer** The number of milliseconds since January 1, 1970, 00:00:00 GMT.

Return Value

Boolean Returns `true` if the specified number of milliseconds is valid, `false` otherwise.

boolean verifyMonth(int month)

Verifies that the specified numeric month is valid.

Input Parameters

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

Return Value

Boolean Returns `true` if the specified month is valid, `false` otherwise.

boolean verifyYear(int year)

Verifies that the specified numeric year is valid.

Input Parameters

year	**Integer** The 4-digit year of the date.

Return Value

Boolean Returns `true` if the specified year is valid, `false` otherwise.

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

year	**Integer** The 4-digit year of the date.
month	**Integer** The numeric month within the specified year (1 = January, 2 = February, ..., 12 = December).
day	**Integer** 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 predefined list and range functions as listed in the following table:

Function	Returns	Description
boolean inList(String[] list)	Boolean	Indicates whether or not this string is in the specified list.
boolean inList(String[] list, boolean ignoreCase)	Boolean	Indicates whether or not this string is in the specified list. Case sensitivity can be ignored while checking.
boolean inRange(Date lowerBound, Date upperBound)	Boolean	Indicates whether or not this date is within the specified range.
boolean inRange(Date lowerBound, Date upperBound, Date[] exclusions)	Boolean	Indicates whether or not this date is within the specified range. Exclusions can be specified.
boolean inRange(Date[][] listOfRanges, Date[] exclusions)	Boolean	Indicates whether or not this date is within the specified list of ranges. Exclusions can be specified.
boolean inRange(Date[][] listOfRanges, boolean inclusiveLower, boolean inclusiveUpper, Date[] exclusions)	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.

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

Function	Returns	Description
boolean inRange(String[][] listOfRanges, String[] exclusions)	Boolean	Indicates whether or not this string is within the specified list of ranges. A separate list of exclusions can be provided.
boolean inRange(String[][] listOfRanges, boolean ignoreCase, boolean inclusiveLower, boolean inclusiveUpper, String[] exclusions)	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.
object list createList(Object[] list)	Object list	Creates a list from an array of items.
object list appendToList(Object[] list, Object item)	Object list	Appends an item to the end of a list.
object list insertIntoList(Object[] list, Object item, int position)	Object list	Inserts an item into an existing list at the specified position.
object list removeFromList(Object[] list, int position)	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

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

ignoreCase	**Boolean** 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

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

lowerBound	**Date** The lower bound of the range to check against (inclusive).
upperBound	**Date** The upper bound of the range to check against (inclusive).
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 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

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.
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

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

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

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(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

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

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

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

lowerBound	**String** The lower bound of the range to check against (inclusive).
upperBound	**String** The upper bound of the range to check against (inclusive).
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 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.
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	**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.

object list createList(Object[] list)

Creates a list from an array of items.

Input Parameters

list	**Object list** 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

list	**Object list** 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 predefined math functions as listed in the following table:

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

Function	Returns	Description
double exp(double val)	Integer	Returns Euler's number `e` raised to the power of the specified double value.
double floor(double val)	Integer	Returns the largest integer that is less than or equal to the specified double value.
double log(double val)	Integer	Returns the natural logarithm (base e) of the specified double value.
long max(long val1, long val2)	Integer	Returns the larger of the two specified long values. If the specified values are equal, then the result is that same value.
double max(double val1, double val2)	Integer	Returns the larger of the two specified double values. If the specified values are equal, then the result is that same value.
long min(long val1, long val2)	Integer	Returns the lesser of the two specified long values. If the specified values are equal, then the result is that same value.
double min(double val1, double val2)	Integer	Returns the lesser of the two specified double values. If the specified values are equal, then the result is that same value.
long mod(long val1, long val2)	Integer	Returns the remainder of two long values. The remainder is obtained when dividing `val1` by `val2`.
double mod(double val1, double val2)	Integer	Returns the remainder of two double values. The remainder is obtained when dividing `val1` by `val2`.
double pi()	Integer	Returns the value of pi to 15 decimal places.
double pow(double base, double exponent)	Integer	Returns $base^{exponent}$ or the value of the base argument raised to the power of the exponent argument.
double radiansToDegrees(double angrad)	Integer	Returns an approximately equivalent angle measured in degrees for the specified angle measured in radians.
long round(double val)	Integer	Returns the closest long integer to the specified double argument, with ties rounding up.

Function	Returns	Description
double round(double val, int scale)	Integer	Rounds the given value to the specified number of decimal places.
double round(double val, int scale, int roundingMethod)	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 `java.math.BigDecimal`.
double sin(double val)	Integer	Returns the trigonometric sine of the specified angle.
double sinh(double val)	Integer	Returns the hyperbolic sine of the specified double value.
double tan(double val)	Integer	Returns the trigonometric tangent of the specified angle.
double tanh(double val)	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 is less than or equal to the argument and is equal to a mathematical integer.

double log(double val)

Returns the natural logarithm (base e) of the specified double value.

Input Parameters

val	**Integer** The number whose natural logarithm (base e) is to be returned.

Return Value

Integer The value `ln val`, the natural logarithm of `val`.

long max(long val1, long val2)

Returns the larger 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 larger of `val1` and `val2`.

double max(double val1, double val2)

Returns the larger 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 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`.

long mod(long val1, long val2)

Returns the remainder of two long values. The remainder is obtained when dividing `val1` by `val2`.

Input Parameters

val1	**Integer** The dividend.
val2	**Integer** The divisor.

Return Value

Integer The remainder after performing the division.

double mod(double val1, double val2)

Returns the remainder of two double values. The remainder is obtained when dividing `val1` by `val2`.

Input Parameters

val1	**Integer** The dividend.
val2	**Integer** The divisor.

Return Value

Integer The remainder after performing the division.

double pi()

Returns the value of pi to 15 decimal places (3.141592653589793).

Input Parameters

None.

Return Value

Integer The value of pi to 15 decimal places.

double pow(double base, double exponent)

Returns base^{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 base^{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

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

roundingMethod	**Integer** Rounding method as defined in BigDecimal.

Return Value

Integer The rounded value.

double sin(double val)

Returns the trigonometric sine of the specified angle.

Input Parameters

val	**Integer** 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.

Input Parameters

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

Return Value

Integer The hyperbolic sine of the argument.

double tan(double val)

Returns the trigonometric tangent of the specified angle.

Input Parameters

val	**Integer** An angle in radians.

Return Value

Integer The tangent of the argument.

double tanh(double val)

Returns the hyperbolic tangent of the specified double value.

Input Parameters

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

Return Value

Integer The hyperbolic tangent of the argument.

Summary of String Functions

webMethods Rules Development provides predefined string functions as listed in the following table:

Function	Returns	Description
concat(String str): String	String	Appends the specified string to the end of this string.
contains(String str): boolean	Boolean	Indicates whether or not the specified string is contained within this string.
endsWith(String suffix): boolean	Boolean	Indicates whether or not this string ends with the specified suffix.
equals(String anotherString): boolean	Boolean	Indicates whether or not this string is equal to the specified string.
equalsIgnoreCase(String anotherString): boolean	Boolean	Indicates whether or not this string is equal to the specified string, ignoring case considerations.
matches(String regex): boolean	Boolean	Indicates whether or not this string matches the given regular expression.
regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len): boolean	Boolean	Indicates whether or not two string regions (substrings within the specified strings) are equal.

Function	Returns	Description
replaceAll(String regex, String replacement): String	String	Returns a string where each substring of this string that matches the given regular expression is replaced with the specified replacement value.
replaceFirst(String regex, String replacement): String	String	Returns a string where the first substring of this string that matches the given regular expression is replaced with the specified replacement value.
startsWith(String prefix): boolean	Boolean	Indicates whether or not this string starts with the specified prefix.
substring(int beginIndex): String	String	Returns a string that resides within this string.
substring(int beginIndex, int endIndex): String	String	Returns a string that resides within this string.
toLowerCase(): String	String	Returns a string with all of the characters of this string converted to lower case.
toUpperCase(): String	String	Returns a string with all of the characters of this string converted to upper case.
trim(): String	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 if 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

ignoreCase	**Boolean** Indicates whether or not case should be ignored when comparing characters.
toffset	**Integer** The starting offset of the subregion in this string.
other	**String** The string argument.

ooffset	**Integer** The starting offset of the subregion in the string argument.
len	**Integer** 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

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

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

prefix	**String** 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.

toUpperCase(): String

Returns a string with all of the characters of this string converted to upper 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

- About 60
- Summary of Rules-Related Event Types 60
- Publishing Business Rules Events 61

About

Business Rules provides a set of predefined event types that allow you to monitor rules-related events on Integration Server or on My webMethods Server.

For the existing event types, see "Summary of Rules-Related Event Types" on page 60.

For more information about how to configure Integration Server or My webMethods Server to subscribe to and work with rules-related events, see "Publishing Business Rules Events" on page 61.

Summary of Rules-Related Event Types

Business Rules provides predefined event types as listed in the following table:

Event Type	Emitted On	Description
Decision Table Changed	My webMethods Server	Is triggered when a difference is detected between two equally named decision tables.
Event Rule Changed	My webMethods Server	Is triggered when a difference is detected between two equally named event rules.
Hot Deployment Started	My webMethods Server	Is triggered when you start to hot deploy a rule project from My webMethods Server to Integration Server.
Project Deleted	My webMethods Server	Is triggered when a rule project is deleted from My webMethods Server.
Project Imported	My webMethods Server	Is triggered when a rule project is imported to My webMethods Server.
Project Exported	My webMethods Server	Is triggered when a rule project is exported or downloaded from My webMethods Server to the file system.
Project Deployed	Integration Server	Is triggered when a rule project is deployed on Integration Server.
Project Undeployed	Integration Server	Is triggered when a rule project is undeployed from Integration Server.

Publishing Business Rules Events

To subscribe to and work with Business Rules event types, you must configure Integration Server and My webMethods Server to publish Business Rules events. You can use the Command Central web user interface for this.

To publish Business Rules events:

1. Do one of the following:

    a. On Integration Server, locate `sag_install_folder\profiles\IS_default\configuration\custom_wrapper.conf` and add the parameters `wrapper.java.additional.109=-DRulesAuditingEDAOnOff=On` and `wrapper.java.additional.111=-DRulesAuditingDESOnOff=On`.

    b. On My webMethods Server, locate `sag_install_folder\profiles\MWS_default\configuration\custom_wrapper.conf` and add the parameters `wrapper.java.additional.109=-DRulesAuditingEDAOnOff=On` and `wrapper.java.additional.111=-DRulesAuditingDESOnOff=On`.

    c. In Command Central, proceed as described in the following steps.

2. On the **Instances** page, select the Integration Server instance or the My webMethods Server instance.

3. On the **Integration Server Instance** page or on the **My webMethods Server Instance** page, select the **Configuration** tab.

4. From the **JVM options** drop-down list, select **Java System Properties**.

5. Click **Edit** in the upper right corner.

6. Add the following entries to the java system properties: `RulesAuditingEDAOnOff=On` and `RulesAuditingDESOnOff=On`.

7. Click **Apply** in the upper right corner.

3 Working with Business Rules Auditing

- About 64
- Summary of Business Rules Auditing Information 64
- Writing Business Rules Auditing Information to the Database 68

About

Business Rules stores auditing information in database tables. The **Change History** panel in Business Console then displays this information. For more information about how to work with Business Console, see *Working with webMethods Business Console*.

For more information about how to configure Integration Server and My webMethods Server to write auditing information to database tables, see "Writing Business Rules Auditing Information to the Database" on page 68.

Summary of Business Rules Auditing Information

The following table lists the auditing information that is stored in the database:

Event Type	Stored Auditing Information	Description
Project Deployed	event start	Time at which the event was created.
	correlation Id	ID which identifies another event related to this deployment.
	display name	Readable name.
	host port	Host and port of the Integration Server on which the rule project was deployed.
	project name	Name of the deployed rule project.
	project version	Version of the deployed rule project.
	user id	User credentials that are used to login to Integration Server.
	deployed by	User who triggers the deployment.
	deployed with	Tool or user interface with which the rule project was deployed.
	invocation session Id	Session ID generated in the SeesionIdGenerator class.
	project creator app	Component that was used to build the rule project archive prior to deployment.
Project Undeployed	event start	Time at which the event was created.
	correlation Id	ID which identifies another event related to this undeployment.
	display name	Readable name.

Event Type	Stored Auditing Information	Description
	host port	Host and port of the Integration Server from which the rule project was undeployed.
	project name	Name of the undeployed rule project.
	project version	Version of the undeployed rule project.
	user id	User credentials that are used to login to Integration Server.
	invocation session Id	Session ID generated in the SeesionIdGenerator class.
Project Imported	event start	Time at which the event was created.
	correlation Id	UUID.
	display name	Readable name.
	host port	Host and port of the My webMethods Server to which the rule project was imported.
	project name	Name of the imported rule project.
	project version	Version of the imported rule project.
	user id	User credentials that are used to login to My webMethods Server.
	invocation session Id	Empty.
	decision entities	List of decision entities that are contained in the imported rule project, except decision trees.
Project Exported	event start	Time at which the event was created.
	display name	Readable name.
	host port	Host and port of the My webMethods Server from which the rule project was exported.
	project name	Name of the exported rule project.
	project version	Version of the exported rule project.
	user id	User credentials that are used to login to My webMethods Server.
	decision entities	List of decision entities that are contained in the exported rule project, except decision trees.
Project Deleted	event start	Time at which the event was created.

Event Type	Stored Auditing Information	Description
	display name	Readable name.
	host port	Host and port of the My webMethods Server from which the rule project was deleted.
	project name	Name of the deleted rule project.
	project version	Version of the deleted rule project.
	user id	User credentials that are used to login to My webMethods Server.
	invocation session Id	My webMethods Server session ID for the user's session.
Hot Deployment Started	event start	Time at which the event was created.
	correlation Id	ID which identifies another event related to this hot deployment.
	display name	Readable name.
	host port	Host and port of the My webMethods Server from which the hot deployment was started.
	project name	Name of the hot deployed rule project.
	project version	Version of the hot deployed rule project.
	user id	User credentials that are used to login to My webMethods Server.
	invocation session Id	My webMethods Server session ID for the user's session.
	Integration Server list	List of Integration Servers.
Decision Table Changed	event start	Time at which the event was created.
	correlation Id	Empty.
	display name	Readable name.
	host port	Host and port of the My webMethods Server on which the decision table was modified.
	project name	Name of the rule project that contains the modified decision table.
	user id	User who edited the decision table on My webMethods Server.

Event Type	Stored Auditing Information	Description
	invocation session Id	My webMethods Server session ID for the user's session.
	name	Name of the modified decision table.
	row number	Number of the modified decision table row.
	row ID	ID of the modified decision table row.
	change type	Added, deleted or changed a row.
	column name	Name of modified column.
	column type	Condition or result column.
	old value	Value before the change.
	new value	Value after the change.
	parameter element	Modified parameter element.
Event Rule Changed	event start	Time at which the event was created.
	correlation Id	Empty.
	display name	Readable name.
	host port	Host and port of the My webMethods Server on which the event rule was modified.
	project name	Name of the rule project that contains the modified event rule.
	user id	User who edited the decision table on My webMethods Server.
	invocation session Id	My webMethods Server session ID for the user's session.
	name	Name of the modified event rule.
	row number	Number of the modified event rule row.
	row ID	ID of the modified event rule row.
	change type	Added, deleted or changed a row.
	result name	Name of modified result.
	old value	Value before the change.
	new value	Value after the change.

Writing Business Rules Auditing Information to the Database

You must configure the settings of the respective servers to write auditing information to the database. You can use the Command Central web user interface for this.

To write auditing information to the database:

1. Do one of the following:

   a. On Integration Server, locate `sag_install_folder\profiles\IS_default\configuration\custom_wrapper.conf` and add the parameter `wrapper.java.additional.112=-DRulesAuditingDBOnOff=On`.

   b. On My webMethods Server, locate `sag_install_folder\profiles\MWS_default\configuration\custom_wrapper.conf` and add the parameter `wrapper.java.additional.112=-DRulesAuditingDBOnOff=On`.

   c. In Command Central, proceed as described in the following steps.

2. On the **Instances** page, select the Integration Server instance or the My webMethods Server instance.

3. On the **Integration Server Instance** page or on the **My webMethods Server Instance** page, select the **Configuration** tab.

4. From the **JVM options** drop-down list, select **Java System Properties**.

5. Click **Edit** in the upper right corner.

6. Add the following entry to the java system properties: `RulesAuditingDBOnOff=On`.

7. Click **Apply** in the upper right corner.

Note:
When you run Business Rules in a Microservice Runtime, the above described procedure does not apply. In this case, locate the *JAVA_CUSTOM_OPTS* variable in *SAG_Installation_directory*/IntegrationServer/bin/setenv.sh and add the `-DRulesAuditingDBOnOff=On` parameter to it.

4 Services

- About 70
- Summary of WmBusinessRules Built-in Services 70
- Summary of REST Services on My webMethods Server 72
- Summary of REST Services on Integration Server 85

About

webMethods Rules Development provides a set of services.

Three categories of services exist:

- "Summary of WmBusinessRules Built-in Services" on page 70.
- "Summary of REST Services on My webMethods Server" on page 72.
- "Summary of REST Services on Integration Server" on page 85.

Summary of WmBusinessRules Built-in Services

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

Element	Package and Description
pub.businessrules.client:invoke	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 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/`[DecisionTreeName]	Define a decision tree.

Create Missing Inputs

Boolean Optional. Creates an empty input parameter without input values if no input parameter is specified for the rule set or decision table. Set to:

- `true` if missing inputs should be created.
- `false` if missing inputs should not be created. This is the default.

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.

Key	Description
[Input Parameter Name]	**Document** Defines the input parameter elements and their values.
Fact IData	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.

Key	Description
[Output Parameter Name]	**Document** Defines the output parameter elements and their values.
Fact IData	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 on My webMethods Server

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

The following table lists REST services using base URL `<host>:<port>/wm_rma/rest/content`:

REST Service	Description
GET <base URL>/projects	Returns a list of rule projects that are currently available on My webMethods Server.
GET <base URL>/project/%ruleProjectName%	Returns the content and metadata for a given rule project.
PUT <base URL>/project/%ruleProjectName%/lock	Locks the given rule project.
PUT <base URL>/project/%ruleProjectName%/unlock	Unlocks the given rule project.
PUT <base URL>/project/%ruleProjectName%/unlockall	Unlocks the given rule project and all decision entities contained in it.
GET <base URL>/project/%ruleProjectName% /decisiontable/%decisionTableName%	This service has been deprecated and replaced with GET <base URL>/projects/%ruleProjectName%/ decision_tables/%decisionTableName%, using base URL `<host>:<port>/wm_rma/rest`.
PUT <base URL>/project/%ruleProjectName%/ decisiontable/%decisionTableName%	Writes the changes for the given decision table to 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 My webMethods Server.

The following table lists REST services using base URL `<host>:<port>/wm_rma/rest/raw`:

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

The following table lists 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.
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.

REST Service	Description
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.
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 My webMethods Server.

Input Parameters

ruleProjectName	**String** Name of the rule project the decision table is part of.
decisionTableName	**String** Name of the decision table to be saved.
body	**Structured JSON** Changed decision table contents.

Output Parameters

None.

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

Locks 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 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 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 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 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 My webMethods Server.

Input Parameters

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

Output Parameters

None.

GET <base URL>/projects/%ruleProjectName%/decision_trees/%decisionTreeName%

Retrieves the given decision tree.

Input Parameters

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

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.

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

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.
conditionNodeId	**String** The unique ID of the condition node 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%/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

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.
conditionNodeId	**String** 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

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.
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

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.
resultNodeId	**String** The unique ID of the result node 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%/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

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.
resultNodeId	**String** 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

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.
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

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.

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

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.

Output Parameters

None.

GET <base URL>/projects/%ruleProjectName%/decision_tables/%decisionTableName%

Retrieves the given decision table in JSON.

Input Parameters

ruleProjectName	**String** Name of the rule project the decision table is part of.
decisionTableName	**String** Name of the decision table to be returned.

Output Parameters

Structured JSON Decision table contents.

Summary of REST Services on Integration Server

webMethods Business Rules provides predefined REST services on Integration Server.

The following table lists REST services using base URL `<host>:<port>/restv2/`:

REST Service	Description
POST <base URL>/ pub.businessrules.restApi.api:businessrules/projects/invoke	Executes a rule set or decision entity and returns the results of execution.

POST <base URL>/ pub.businessrules.restApi.api:businessrules/projects/invoke

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

Input Parameters

projectName	**String** Name of the rule project to be invoked.
invocationTarget	**String** Type of decision entity and decision entity name. Specify one of the following:

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

`D3/`[DecisionTreeName]	Define a decision tree.
createMissingInputs	**Boolean** Optional. Defines whether missing input parameters should be added as empty documents to invoke the call. Set to: ■ `true` if missing inputs should be created. ■ `false` if missing inputs should not be created. This is the default.
inputs	**Structured JSON** Optional. Defines the input parameter documents with their values.
desiredOutputs	**Structured JSON** Optional. Defines the filtering of the results according to the respective parameter names.

Output Parameters

Structured JSON

Example

This is an example input for a REST invoke call. This is send via body during the POST request.

```
{
  "projectName": "CustomerOrder",
  "invocationTarget": "DT/DetermineDiscountFromOrder",
  "createMissingInputs": true,
  "inputs": {
    "IncomingOrder": {
      "orderValue": 5000,
      "company": {
        "companyName": "ACME Inc"
      },
      "preferred": "false",
      "orderDate": "2020-01-13T11:50:01+01:00"
    }
  },
  "desiredOutputs": [
    {
      "parameterName": "IncomingOrder"
    }
  ]
}
```

The output looks like:

```
{
    "outputs": {
        "IncomingOrder": {
            "preferred": false,
            "discount": 14.9,
            "orderValue": 5000,
            "discountDate": "2020-01-13T11:54:14.024+01:00",
            "orderDate": "2020-01-13T11:50:01+01:00",
```

```
            "company": {
                "companyName": "ACME Inc"
            }
        }
    },
    "error": null
}
```

In case of error, the output looks like:

```
{
    "outputs": null,
    "error": {
        "status": 404,
        "code": "0050.8003",
        "message": "Can not find project with key [projectName=Project1]"
    }
}
```

5 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"
    }
  ]
```

6 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" ]
        },
        "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
```

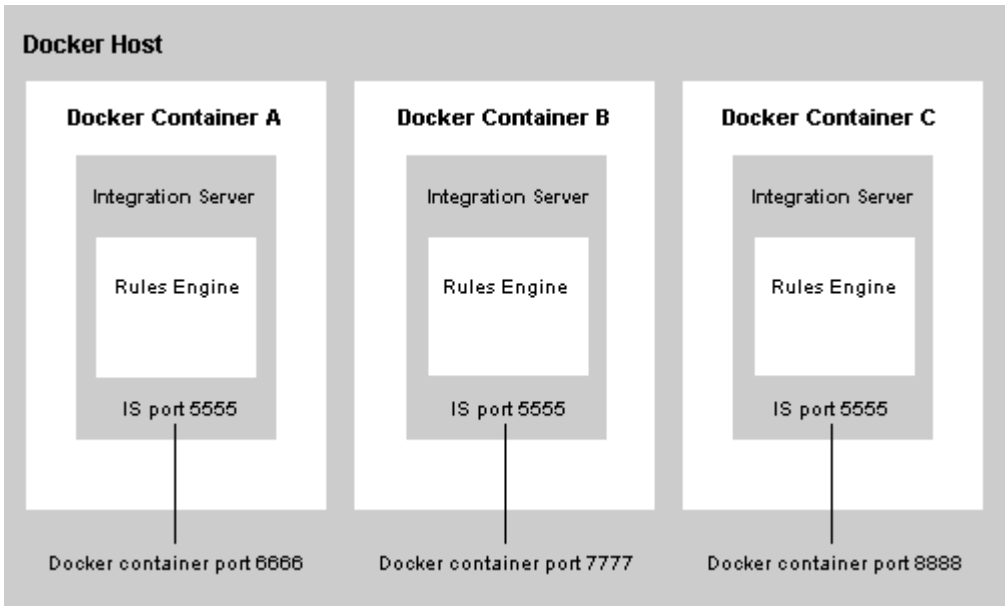
7 Using Docker for Business Rules

Docker is an open-source technology that allows you to run software applications in containers. You can run an instance of Rules Engine in a Docker container. To do so, you must first create a Docker image from an installed and configured Integration Server instance as described in *webMethods Integration Server Administrator's Guide*.

When starting a Docker container that includes an instance of Rules Engine, you must pass the following two environment variables:

- RULES_HOST_NAME Name of the Docker host.
- RULES_PORT Port of the Docker container that is to be mapped to the port of the Integration Server on which the Rules Engine instance is running.

The following image describes the Docker architecture for Business Rules.



To pass the environment variables when starting the Docker container:

1. Enter the following command:

```
docker run -d -e RULES_HOST_NAME='docker_host_name'
```

```
-e RULES_PORT='docker_container_port'
-p docker_container_port:5555 rules_engine_dockerimage
```

8 Administering Business Rules with Command Central

- About 96
- Commands that Business Rules Supports 96
- Configuration Types that Business Rules Supports 98
- Run-Time Monitoring Statuses for Business Rules 100

About

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" on page 100.

- 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 "Configuration Types that Business Rules Supports" on page 98.

- 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 "Configuration Types that Business Rules Supports" on page 98.

- 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 "Configuration Types that Business Rules Supports" on page 98.

- Configure whether users, groups and roles are displayed in the Principal Picker controls. For more information about configuring the Principal Picker, see "Configuration Types that Business Rules Supports" on page 98.

- Configure the event source for the Business Rules change history in the Business Console.

For information about the configuration commands that Business Rules supports, see "Commands that Business Rules Supports" on page 96.

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...
`sagcc get configuration data`	For general information about the command, see the topic about administrating commands in *Software AG Command Central Help*. For information about the configuration types that Business Rules supports, see "Configuration Types that Business Rules Supports" on page 98.
`sagcc update configuration data`	For general information about the command, see the topic about administrating commands in *Software AG Command Central Help*. For information about the configuration types that Business Rules supports, see "Configuration Types that Business Rules Supports" on page 98.
`sagcc get configuration instances`	For general information about the command, see the topic about administrating commands in *Software AG Command Central Help*. For information about the configuration types that Business Rules supports, see "Configuration Types that Business Rules Supports" on page 98.
`sagcc list configuration instances`	For general information about the command, see the topic about administrating commands in *Software AG Command Central Help*. For information about the configuration types that Business Rules supports, see "Configuration Types that Business Rules Supports" on page 98.
`sagcc get configuration types`	For general information about the command, see the topic about administrating commands in *Software AG Command Central Help*. For information about the configuration types that Business Rules supports, see "Configuration Types that Business Rules Supports" on page 98.
`sagcc list configuration types`	For general information about the command, see the topic about administrating commands in *Software AG Command Central Help*. For information about the configuration types that Business Rules supports, see "Configuration Types that Business Rules Supports" on page 98.

Commands	For more information, see...
`sagcc get monitoring`	For general information about the command, see the topic about administrating commands in *Software AG Command Central Help*. For information about the configuration types that Business Rules supports, see "Configuration Types that Business Rules Supports" on page 98.
`sagcc exec lifecycle components OSGI-MWS_default start`	For general information about the command, see the topic about administrating commands in *Software AG Command Central Help*.
`sagcc exec lifecycle components OSGI-MWS_default stop`	For general information about the command, see the topic about administrating commands in *Software AG Command Central Help*.

Configuration Types that Business Rules Supports

The Business Rules run-time component supports the configuration types that are listed in the following table.

Configuration Type	Use to configure...
COMMON-COMPONENT-ENDPOINTS	A connection from My webMethods Server to one or multiple Integration Server(s) to hot deploy rule projects.
BRMS_REST_ENDPOINT	Base URL and optional authentication data of a REST server that verifies user data in decision tables. Base URL is used by My webMethods Server whenever a verification 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 verification service.
BRMS_REST_ENDPOINT _DATA_PROVIDER	Base URL and optional authentication data of a REST server that verifies user data in 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.
AUDITING	The event source for the Business Rules change history in the Business Console.

Configuration Type	Use to configure...
	For more information on how to configure auditing in Command Central, see "Writing Business Rules Auditing Information to the Database" on page 68, *Steps 2 to 7*.
DISPLAY_OPTIONS	Display settings, for example, whether functions in expressions are truncated when displayed.

These configuration types correspond to the following configurations in the Business Rules UI on My webMethods Server:

- COMMON-COMPONENT-ENDPOINTS: Integration Server Connection(s). For more information see, *Working with Business Rules in My webMethods*, *Configuring an Integration Server Connection*.
- BRMS_REST_ENDPOINT: Business Verification. For more information see, *Working with Business Rules in My webMethods*, *Configuring a Server Connection for a Preconfigured Verification Service*.
- BRMS_REST_ENDPOINT_DATA_PROVIDER: Data Provider. For more information see, *Working with Business Rules in My webMethods*, *Configuring a Server Connection for a Preconfigured Data Provider Service*.
- PRINCIPAL_TYPES: Principal Types. For more information see, *Working with Business Rules in My webMethods*, *Configuring Principal Types*.
- DISPLAY_OPTIONS: Display Options. For more information see, *Working with Business Rules in My webMethods*, *Working with Expressions*.

To specify configuration types in Command Central

1. Log in to Command Central.
2. Navigate to **Instances > MWS_Default > Business Rules**.
3. Select the **Configuration** tab.
4. From the drop-down list, select the respective configuration type.
5. Modify the properties as required. For more information, see the respective topics in *Working with Business Rules in My webMethods*.
6. Click **Save** to save the configuration, or click **Cancel** to abort the process.

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.