

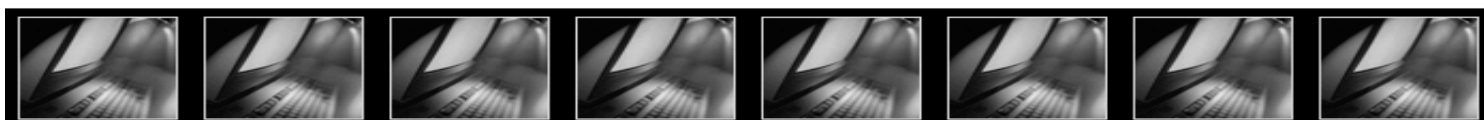
V8 Math Syntax User's Guide

Oracle Insurance Policy Administration - Life

Release 8.1

Part Number: E14444-01

May 2009



Oracle® Insurance Policy Administration - Life Release V8.1

Copyright© 2009, 2011 Oracle and/or its affiliates. All rights reserved.

License Restrictions & Warranty Disclaimer

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

Restricted Rights Notice

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Third Party Web Sites, Content, Products, and Services Disclaimer

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Trademark Notice

Oracle, JD Edwards, and PeopleSoft are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Table of Contents

Table of Contents	3
Introduction.....	4
Overview	4
Syntax Changes.....	5
Business Rule and Transaction Conversion Steps	7
NUMERICARRAY Syntax Changes	9
Common Errors	16

Introduction

This document describes the syntax used for mathematical calculations used in the Oracle Insurance Policy Administration (OIPA) system at the time of the version 8 release, which consists of the new calculation engine. It also lists the changes that must be made to existing calculations in order to run them with the new calculation engine.

Overview

In order to increase the performance of the calculation subsystem, a new engine was implemented that performs the calculations by translating them to java classes, which are compiled and executed at run time. The compiler used for this purpose is the open source Janino compiler, also referred to as the **Janino Calculation Engine**.

The previous calculation engine was based on a similar principle where the XML calculation definition is translated into executable code. However, the previous version used Mozilla's Rhino JavaScript instead of Java. JavaScript is not a strongly typed language and the JavaScript engine dynamically determined the data types of the variables in the scripts. This can be both a benefit and a burden. On one hand, the configurator does not have to worry about what datatype any variable should be. But on the other hand, there are situations where the engine chooses the wrong datatype for a variable and unintended consequences result. Such is the case, on occasion, where two "numbers" are concatenated when addition was intended. This happens when at least one of the numbers is being treated as a text value rather than a numerical value.

In order to update the XML calculations so they work with the version 8 calculation engine, datatype information must be added to the MathVariable elements of the calculations. Without such changes to the calculation XML, the calculations will still be executed, but it is unlikely that the engine will be able to properly determine the datatypes of the variables and the calculation will most likely fail.

In addition to the data type changes, there are a few other syntactical changes that must be made due to the need for stricter parsing algorithms. The next section lists the changes needed to update the calculation XML.

Syntax Changes

- All MathVariable elements must have a DATATYPE attribute. Allowable values for this attribute are:
 - TEXT
 - INTEGER
 - DECIMAL
 - DATE
 - OBJECT
 - MAP
- Once a MathVariable is assigned a DATATYPE value, you can never change it to a different datatype.
- MathVariables where TYPE="FUNCTION" often pass variable values to the named function. Often the current calculation XML encloses the variable names in the parameter list in quotes. Because of this, the calculation engine will treat the parameters as text literals rather than variable names. **The quotes must be removed.**
 - Example: Suppose there are variables DOB (date of birth) and MaturityDate
 - Current: `<MathVariable VARIABLENAME= "Age" TYPE = "FUNCTION">AgeOf("DOB", "MaturityDate")</MathVariable>`
 - New: `<MathVariable VARIABLENAME= "Age" TYPE = "FUNCTION" DATATYPE="INTEGER">AgeOf(DOB, MaturityDate)</MathVariable>`
- When comparing variable values to literals in an expression, make sure that TEXT variables, if literal, have single quotes around them. Likewise, if the variable is INTEGER or DECIMAL, **do not include quotes.**
 - Example: Suppose Status is a text variable and Age is an integer variable
 - `<MathIF IF="Status = '02'">` (Note single quotes around the 02)
 - `<MathIF IF="Age > 24">`
- **Be careful when performing a division operation with literal values and INTEGER variables.** An expression such as `"1 / 12"` will be evaluated to 0 (zero) because Java interprets this as integer division and thus the result will be an integer even if it is being assigned to a DECIMAL variable. To force floating point division, change this to `"1 / 12.0"`. The standard is to make the denominator into a DECIMAL variable in the expression.
- Once a variable datatype is defined, it cannot be changed into something else. This is true of datatypes and variables defined as an array or a scalar value.

- Due to some limitations in the array manipulation syntax in the Rhino engine, there are many cases where workarounds are used, particularly for initializing or clearing arrays. Most of the time, this was done by creating the variable as an EXPRESSION type and setting its value to 0. This will no longer work since that variable will never be able to hold an array. A new NUMERICARRAY operation, CREATE is now available to properly create the array with a given size. See the next section for the definition of the CREATE operation.
- Any defined Business Rule functions must have their parameters and return value updated with a DATATYPE attribute.
 - For the return value, this will be in the *Function* tag.
 - In addition to having the datatype specified, the *Parameter* and *Function* elements must include the attribute *ISARRAY="TRUE"* if they are arrays.

Business Rule and Transaction Conversion Steps

The following steps will demonstrate the process for converting Business Rules and Transactions.

Steps to Convert Business Rules and Transactions

1. Construct a list of all current Transactions.
 - a. Include a hierarchical tree of included CopyBooks and Functions.
2. Add to the list all Business Rules that include math calculations.
 - a. PolicyValues
 - b. CalculateGeneral*
 - c. MVAAmountFormula
 - d. FreeAmountFormula
 - e.
3. Order the list according to one of the following schemes:
 - a. Order them in the order that they would normally be executed during the course of adding a new policy and processing it up to some predetermined point.

Pros

 - Should be a well known execution path
 - No special policies have to be set up before conversion

Cons

 - Not necessarily the easiest or fastest way to get started; i.e. some of the more complex calculations may exist in the segment calculations or early transactions such as Issue. Therefore much effort will have to be expended before any results are seen.
4. Order them in some relative measure of complexity.

Pros

 - Can demonstrate early success by converting transactions that require little or no manual intervention

Cons

 - Special care will have to be taken to create policies already in states whose eligible transaction lists collectively include all transactions being converted.
5. Run the **Business Rule Conversion** utility.
 - a. This utility will connect to a database. Read each Rule and Transaction and convert it to the syntax necessary for running the version 8 calculation engine.
 - b. Each converted Rule and Transaction will be written out to the file system in a separate XML file. Optionally, the Rules can be written directly back to the database.

Note: Please exercise caution with this option and operate only on a disposable copy of the real database.

- c. The conversion is mostly concerned with adding the DATATYPE attribute to all MathVariable elements.
 - d. The utility guesses at the correct datatype by examining the variable type, its name and other factors.
- 6. Manually review the converted files and the attendant logs for correctness.
- 7. Perform the steps above and iterate through the Rules and Transactions.
 - a. Update the database with the converted XML.
 - b. Test it in the appropriate policy.
 - c. Correct any errors.

NUMERICARRAY Syntax Changes

The following table lists the OPERATION values available for use for the NUMERICARRAY MathVariable type.

OPERATION	Description	Attributes	Content
ADDSCALE	Creates a new array by adding the corresponding elements of array1 (SOURCEARRAY) and array2 after first scaling the elements of array2 by a value.	SOURCEARRAY	Comma separated list: array2, scalar factor.
$\langle \text{MathVariable } \text{VARIABLENAME}=\text{"AddScaleArray"} \text{ TYPE}=\text{"NUMERICARRAY"} \text{ OPERATION}=\text{"ADDSCALE"} \text{ SOURCEARRAY}=\text{"AddArray"} \rangle \text{ArrayAddArray, ScaleBy} \langle / \text{MathVariable} \rangle$			
ADD	Returns a new array where each element is the result of adding a value to each element of the array specified in SOURCEARRAY.	SOURCEARRAY	Scalar value or variable
$\langle \text{MathVariable } \text{VARIABLENAME}=\text{"AddArray"} \text{ TYPE}=\text{"NUMERICARRAY"} \text{ OPERATION}=\text{"ADD"} \text{ SOURCEARRAY}=\text{"CopiedArray"} \rangle 33 \langle / \text{MathVariable} \rangle$			
APPENDALL	Creates a new array by appending the values in the tag content array to SOURCEARRAY.	SOURCEARRAY	Array variable name
$\langle \text{MathVariable } \text{VARIABLENAME}=\text{"AppendAllArray"} \text{ TYPE}=\text{"NUMERICARRAY"} \text{ OPERATION}=\text{"APPENDALL"} \text{ SOURCEARRAY}=\text{"ArrayOne"} \rangle \text{ArrayTwo} \langle / \text{MathVariable} \rangle$			
APPEND	Creates a new array by appending the value in the tag content to SOURCEARRAY.	SOURCEARRAY	Scalar variable or value
$\langle \text{MathVariable } \text{VARIABLENAME}=\text{"ExpandArray2"} \text{ TYPE}=\text{"NUMERICARRAY"} \text{ OPERATION}=\text{"APPEND"} \text{ SOURCEARRAY}=\text{"ArrayOne"} \rangle \text{ValueToAppend} \langle / \text{MathVariable} \rangle$			

OPERATION	Description	Attributes	Content
ARRAYADD	Returns a new array where each element is the sum of the elements of two other (same sized) arrays. One array is named in SOURCEARRAY, the other is in the content of the tag.	SOURCEARRAY	Array variable name
$\text{<MathVariable VARIABLENAME="ArrayAddArray" TYPE="NUMERICARRAY" OPERATION="ARRAYADD" SOURCEARRAY="Array">AddArray</MathVariable>}$			
COPY	Copies the array named in SOURCEARRAY.	SOURCEARRAY	{empty}
$\text{<MathVariable VARIABLENAME="CopiedArray" TYPE="NUMERICARRAY" OPERATION="COPY" SOURCEARRAY="Array"/>}$			
CREATE	Creates a new (double) array initialized to the specified number of elements. All elements have an initial value of 0.0.		Value or variable indicating the size of the new array.
$\text{<MathVariable VARIABLENAME="NewArray" TYPE="NUMERICARRAY" OPERATION="CREATE">Length</MathVariable>}$			
DIVIDE	Creates a new array by dividing each element of SOURCEARRAY by the corresponding element of the array in the tag content.	SOURCEARRAY	Array variable name.
$\text{<MathVariable VARIABLENAME="DivideArray" TYPE="NUMERICARRAY" OPERATION="DIVIDE" SOURCEARRAY="AddArray">DivisorArray</MathVariable>}$			
EXPAND	Creates a new array by repeating each element of SOURCEARRAY the number of times given in the tag content.	SOURCEARRAY	A comma separated list of up to three integer values or variables.

OPERATION	Description	Attributes	Content
<p><MathVariable VARIABLENAME="ExpandArray" TYPE="NUMERICARRAY" OPERATION="EXPAND" SOURCEARRAY="RemoveThese">2,4,3</MathVariable></p> <p>OLD WAY: <MathVariable VARIABLENAME="ExpandArray" TYPE="NUMERICARRAY" OPERATION="EXPAND" MULTIPLIER="5,2,3"> arrayVar1</MathVariable></p>			
FILLBY-LIST	Creates a new array whose elements are listed in the tag content.		A comma separated list of numeric values or variable names.
<p><MathVariable VARIABLENAME="ExponentArray" TYPE="NUMERICARRAY" OPERATION="FILLBY-LIST">1.5, OneHalf, 10, -2</MathVariable></p> <p>NOTHING CHANGED????</p> <p>OLD WAY: <MathVariable VARIABLENAME="arrayVar1" TYPE="NUMERICARRAY" OPERATION="FILLBY-LIST">10, Twenty, 30, 40, 50,60,70,80,90</MathVariable></p>			
FILLBY-SQL	Creates a new array whose elements are values returned by the SQL statement.		A valid SQL SELECT statement. It should select from a single column.
<p><MathVariable VARIABLENAME="RateArray" TYPE="NUMERICARRAY" OPERATION="FILLBY-SQL">Select Rate from AsRate</MathVariable></p> <p>Old Way: <MathVariable VARIABLENAME="RateArray" TYPE="RATEARRAY" STARTINDEX="StartIndex" ENDINDEX="EndIndex"></p>			
INSERTALL	Creates a new array by inserting the values in the named array into SOURCEARRAY at position INDEX. INDEX may be a constant, variable or "FIRST" or "LAST". If INDEX is missing, the value is appended to the end of the array.	SOURCEARRAY; INDEX (optional)	Array variable name.
<p><MathVariable VARIABLENAME="InsertAllArray" TYPE="NUMERICARRAY" OPERATION="INSERTALL" SOURCEARRAY="AddArray" INDEX="InsertAt">RemoveThese</MathVariable></p>			

OPERATION	Description	Attributes	Content
INSERT	Creates a new array by inserting a value into SOURCEARRAY at position INDEX. INDEX may be a constant, variable or "FIRST" or "LAST". If INDEX is missing, the value is appended to the end of the array.	SOURCEARRAY; INDEX (optional)	Array variable name.
<p><MathVariable VARIABLENAME="InsertArray" TYPE="NUMERICARRAY" OPERATION="INSERT" SOURCEARRAY="AddArray" INDEX="InsertAt">ScaleBy</MathVariable></p> <p>OLD WAY: <MathVariable VARIABLENAME=" arrayVar2" TYPE="NUMERICARRAY" OPERATION="INSERTITEMS" INDEX="LAST"> arrayVar1</MathVariable></p>			
MULTIPLY	Creates a new array by multiplying each element of the array in SOURCEARRAY by the corresponding element of the array in the tag content.	SOURCEARRAY	Array variable name.
<p><MathVariable VARIABLENAME="MultiplyArray" TYPE="NUMERICARRAY" OPERATION="MULTIPLY" SOURCEARRAY="ScaleArray">DivideArray</MathVariable></p>			
POWERALL	Creates a new array by raising each element of SOURCEARRAY to the power given in the corresponding element of the array named in the tag content.	SOURCEARRAY	Array variable name.
<p><MathVariable VARIABLENAME="StridePowerAll" TYPE="NUMERICARRAY" OPERATION="POWERALL " SOURCEARRAY="StrideArray">ExponentArray</MathVariable></p>			
POWER	Creates a new array by raising each element of SOURCEARRAY to the given power.	SOURCEARRAY	
<p><MathVariable VARIABLENAME="StridePower2" TYPE="NUMERICARRAY" OPERATION="POWER" SOURCEARRAY="StrideArray">ReplaceAt</MathVariable></p>			

OPERATION	Description	Attributes	Content
REMOVEALL	Creates a new array by removing all elements at the indices found in the array named in INDEX from SOURCEARRAY.	SOURCEARRAY	{empty}
$\text{<MathVariable VARIABLENAME="RemoveAllArray" TYPE="NUMERICARRAY" OPERATION="REMOVEALL" SOURCEARRAY="MultiplyArray" INDEX="RemoveThese">}$			
REMOVE	Creates a new array by removing the element at INDEX from SOURCEARRAY.	SOURCEARRAY	{empty}
$\text{<MathVariable VARIABLENAME="RemoveArray" TYPE="NUMERICARRAY" OPERATION="REMOVE" SOURCEARRAY="MultiplyArray" INDEX="RemoveAt">}$			
REPLACE	When INDEX contains a single value or variable, replaces the value of the element at index INDEX with the given value. When INDEX is a comma separated list of two values and/or variables, all elements from value 1 through value 2 are replaced.	INDEX	A number or numeric variable name.
$\text{<MathVariable VARIABLENAME="CopiedArray" TYPE="NUMERICARRAY" OPERATION="REPLACE" INDEX="ReplaceAt">1.1</MathVariable>}$ $\text{<MathVariable VARIABLENAME="CopiedArray" TYPE="NUMERICARRAY" OPERATION="REPLACE" INDEX="ReplaceFrom,ReplaceTo">TwoPointTwo</MathVariable>}$ OLD WAY: $\text{<MathVariable VARIABLENAME="Array" TYPE="NUMERICARRAY" OPERATION="REPLACE" INDEX="Start,Finish">ReplacementValue</MathVariable>}$			
SCALE	Returns a new array where each element is the product of an element in SOURCEARRAY and the given scalar factor.	SOURCEARRAY	A number or numeric variable name.
$\text{<MathVariable VARIABLENAME="ScaleArray" TYPE="NUMERICARRAY" OPERATION="SCALE" SOURCEARRAY="ArrayAddArray">ScaleBy</MathVariable>}$			

OPERATION	Description	Attributes	Content
SETMAXALL	Creates a new array whose values are the greater of the corresponding elements of SOURCEARRAY and the tag value.	SOURCEARRAY	Array variable name.
\langle MathVariable VARIABLENAME="MaxAllArray" TYPE="NUMERICARRAY" OPERATION="SETMAXALL" SOURCEARRAY="ArrayAddArray">AddScaleArray</MathVariable>			
SETMAX	Creates a new array whose values are the greater of the corresponding element of SOURCEARRAY and the tag value.	SOURCEARRAY	A number or numeric variable name.
\langle MathVariable VARIABLENAME="SetMaxArray" TYPE="NUMERICARRAY" OPERATION="SETMAX" SOURCEARRAY="RemoveArray">5</MathVariable>			
SETMINALL	Creates a new array whose values are the smaller of the corresponding elements of SOURCEARRAY and the tag value.	SOURCEARRAY	Array variable name.
\langle MathVariable VARIABLENAME="MinAllArray" TYPE="NUMERICARRAY" OPERATION="SETMINALL" SOURCEARRAY="ArrayAddArray">AddScaleArray</MathVariable>			
SETMIN	Creates a new array whose values are the smaller of the corresponding element of SOURCEARRAY and the tag value.	SOURCEARRAY	A number or numeric variable name.
\langle MathVariable VARIABLENAME="SetMinArray" TYPE="NUMERICARRAY" OPERATION="SETMIN" SOURCEARRAY="RemoveArray">5</MathVariable>			
STRIDE	Creates a new array by taking every nth element from SOURCEARRAY.	SOURCEARRAY	
\langle MathVariable VARIABLENAME="StrideArray" TYPE="NUMERICARRAY" OPERATION="STRIDE" SOURCEARRAY="ExpandArray3">4</MathVariable>			

OPERATION	Description	Attributes	Content
TRANSFORM	Creates a new array by evaluating the expression in the tag content.		A mathematical expression containing numeric values and variable names. The variables can be arrays or scalar values. At least one array must be present. If multiple arrays are in the expression, they must all have the same cardinality.
<p> <i><MathVariable VARIABLENAME="TransformTest" TYPE="NUMERICARRAY"</i> <i>OPERATION="TRANSFORM">ArrayFillByList * 5</MathVariable></i> </p>			

Common Errors

Below is a list of common errors received while executing converted Transactions:

Error	Cause	Solution
Caused by: org.codehaus.janino.Parser\$ParseException: File Function_LoanRateUL, Line 32, Column 152: Operator ")" expected	This generally means that a variable of type FUNCTION is calling a function and has double quotes around one or more of the arguments.	Remove the quotes.
Type mismatch: "cannot compare..." etc,	Missing datatype declaration on one or more variables that makes the parser assume text. Frequently happens with SQL and XML types.	Add the proper DATATYPE value.
Call to set input parameters failed for function xxx failed	Input parameters missing datatypes.	Add the proper DATATYPE value.
Invalid datatype: Field 'False' is not a text field.	An expression variable with no datatype is being set to False, which is normally defined as an INTEGER.	Set the DATATYPE to INTEGER.