



Process Intelligence for Manufacturing.

Derived Parameters: Math Functions

Derived data is a new parameter created by applying a mathematical formula to existing parameters. You can create both discrete and continuous derived parameters using various calculations.

Numeric Operators

$- \text{expression1}$	Negates the value. This is the same as: $-1 * \text{expression1}$
$\text{expression1} - \text{expression2}$	Subtracts expression2 from expression1 . If either value is null, the result is null.
$\text{expression1} + \text{expression2}$	Adds the two expressions. If either expression is null, the result is null. Note: To ignore nulls when adding, use the sum() function. E.g. $1 + \text{null} + 2 = \text{null}$ $\text{sum}(1, \text{null}, 2) = 3$
$\text{expression1} * \text{expression2}$	Multiplies the two expressions. If either expression is null, the result is null.
$\text{expression1} / \text{expression2}$	Divides expression1 by expression2 . If either value is null, or if the result is infinite, the final result value is null.



$expression1 \wedge expression2$	Evaluates <i>expression1</i> raised to the power <i>expression2</i> . This works the same as the pow functions.
----------------------------------	---

Constants

exp()	$e1$
pi()	π
null	Used to set a numeric parameter to not a number

Boolean Operators

if(condition, true-expression [,falseexpression])	Evaluation condition, true value, optional false value E.g. if ("Discrete 1" < 50, 'good', 'bad')
and	Boolean and operator
or	Boolean or operator
<	Strictly less than
<=	Less than or equal to
=	Equals
==	Equals
>=	Greater than or equal to
>	Strictly greater than
!=	Not equal to
<>	Not equal to

Numeric Functions

abs(expression)	Absolute value of an expression
sqrt(expression)	Square root of an expression
ceiling(expression)	Rounds up to the nearest integer
floor(expression)	Rounds down to the nearest integer
round(expression)	Rounds to the nearest integer. .5 rounds up.
ln(expression)	Natural log of an expression
log10(expression)	Log base 10 of an expression

log(expression, base)	Log of an expression for a specified base
pow(expression, value-raised-to)	An expression raised to a specified power. value-raised-to may be a floating number.
average(expression1,expression2,...)	Average of two or more expressions. Nulls are ignored.
stddev(expression1,expression2,...)	Standard Deviation of two or more expressions. Null values are ignored.
sum(expression1,expression2,...)	Total of two or more expressions. Null values are ignored.
min(expression1,expression2,...)	Minimum value of two or more expressions. Null values are ignored.
max(expression1,expression2,...)	Maximum value of two or more expressions. Null values are ignored.
standardize(expression)	Equal to (parameter value-average) /stddev, where average and stddev are based on all the Parameter Sets for discrete parameters and all the values within a batch for continuous parameters.
to_string(expression[,format])	Converts numbers and dates to strings. The format is optional. Number format uses Java number formatting. See: Decimal Format e.g. Format '00.00' converts 2 to '02.00' Format '##.0#' converts 2 to '2.0' For dates, if the format is not specified, the Discoverant global date format is used. Date format uses the Java date formatting symbols. See: Simple Date Format
area_minutes	Calculates the area of the passed-in expression. The area is calculated using time units of minutes. <i>The expression must be a continuous parameter.</i>

derivative_minutes(expression)	Takes the derivative of the passed-in expression. The derivative is calculated using time units of minutes. <i>The expression must be a continuous parameter.</i>
interpolate(expression)	Interpolates (linearly) the passed-in expression in order to replace null values. <i>The expression must be a continuous parameter.</i>
rollavg(expression, samples)	Calculates the rolling average for the pass-in expression. <i>Samples</i> is the number of samples on either side of the current value to include in the average. Null values are ignored. <i>The expression must be a continuous parameter.</i>

String Functions

	Concatenates two strings. Example: 'a' ':' 'b' = a:b
length(expression)) Length of a string
instr(expression1,expression2)	Position of first occurrence of expression2 found in expression1. Position starts at one, and zero is returned when not found. E.g., instr("param", 'u') returns 2 when "param" is 'bud'.
left(expression, n)	Left 'n' characters of expression
right(expression, n)	Right 'n' characters of expression
lower(expression)	Expression in all lower case
upper(expression)	Expression in all upper case
substring(expression, startindex [, stopindex])	Substring of expression from the start index to the stop index (inclusive).
lower(expression)	Expression in all lower case
trim(expression)	Strips leading and trailing spaces from expression
to_number(expression)	Converts expression to a number, if possible. If not, null is returned.

Date Functions

day(DateArgument1,DateArgument2)	Difference in days (DateArgument1-DateArgument2)
hour(DateArgument1,DateArgument2)	Difference in hours (DateArgument1-DateArgument2)
minute(DateArgument1,DateArgument2)	Difference in minutes (DateArgument1-DateArgument2)
second(DateArgument1,DateArgument2)	Difference in seconds (DateArgument1-DateArgument2)
timestamp()	Returns the current timestamp associated with the parameter. An optional "lag" index may be included as an argument to this function. Continuous parameters only. e.g. timestamp(1) returns the timestamp associated with the previous value, while timestamp(-3) returns the timestamp from 3 offsets in the future, relative to the current offset.

General Functions

lag(expression, n)	Refers to the n th previous value when n is positive and n th future value when n is negative.
prev_value()	Refers to the previous value of the parameter being derived. Example using the MixTest demo AG: if (prev_value()=null,1,if("Discrete 9"='Bud',1,prev_value()+1)) will start at one and reset to one whenever "Discrete 9"='Bud'.
decode(expression1,expression2,...)	This works like the Oracle decode function (much like the <i>case</i> statement). The first expression is the expression to decode, and is often a Parameter. Next are pairs of



input/output values, where the output value is returned when *expression1* equals the input value. Finally, there is an optional default value for when the expression does not match any input value. Note that the input/output values and the default value may also be expressions.

Decode("Valve", 0, 'closed', 1, 'open', 'offline') takes the numeric valve parameter and converts to a meaningful string parameter. 0 gets replaced with 'closed', one gets replaced with 'open', and any other value gets replaced with 'offline'.

'closed', one gets replaced with 'open', and any other value gets replaced with 'offline'.

