Supported functions in the Ultimus Adaptive BPM Suite 8.*x* formulas engine

1 Introduction

This document outlines supported functions within the Ultimus Adaptive BPM Suite 8.*x* formulas engine.

The following table lists down the functions that are implemented for the formula engine and how they will behave. The implementations of the functions have been modified to make more sense in context of schema nodes.

Function Name	Description
ABS()	Returns the absolute value of a number **Syntax** ABS(Formula returning Number) ABS(Constant) ABS(Simple Numeric Node)
ACOS()	Returns the arc cosine of a number **Syntax** ACOS(Formula returning Number) ACOS(Constant) ACOS(Simple Numeric Node)
ACOSH()	Returns the inverse hyperbolic cosine of a number **Syntax** ACOSH(Formula returning Number) ACOSH(Constant) ACOSH(Simple Numeric Node)
AND()	Returns True if all arguments are true; returns False if at least one argument is false. **Syntax** AND(Simple Boolean Node/Formula/Constant, Simple Boolean Node/Formula/Constant ...) AND(Repeated Simple Boolean Node) AND(Simple Boolean Nodes in Complex Repeated)
ASIN()	Returns the arc sine of a number **Syntax** ASIN (Formula returning Number) ASIN (Constant) ASIN (Simple Numeric Node)
ASINH()	Returns the inverse hyperbolic sine of a number

	Syntax ASINH (Formula returning Number) ASINH (Constant) ASINH (Simple Numeric Node)
ATAN()	Returns the arc tangent of a number **Syntax** ATAN (Formula returning Number) ATAN (Constant) ATAN (Simple Numeric Node)
ATAN2()	Returns the arctangent of the specified coordinates. **Syntax** ATAN2 (x, y) • x is the x coordinate. • y is the y coordinate. x and y coordinates can be specified as ATAN2(Simple Numeric Node/Constant/Formula, Simple Numeric Node/Constant/Formula)
ATANH()	Returns the inverse hyperbolic tangent of a number **Syntax** ATANH (Formula returning Number) ATANH (Constant) ATANH (Simple Numeric Node)
AVERAGE()	Returns the average of the supplied numbers. The result of AVERAGE is also known as the arithmetic mean. **Syntax** AVERAGE(Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant ...) AVERAGE(Repeated Simple Numeric Node) AVERAGE(Simple Numeric Nodes in Complex Repeated)
CEILING()	Rounds a number up to the nearest multiple of a specified significance. **Syntax** CEILING (number, significance) • number is the value to round.

	• significance is the multiple to which to round. number and significance can be specified as CEILING(Constant/Formula/Simple Numeric Node, Constant/Formula/Simple Numeric Node)
CHAR()	Returns a character that corresponds to the supplied ASCII code. **Syntax** CHAR (Fomula returning Number) CHAR (Constant) CHAR (Simple Numeric Node)
CHOOSE()	Returns a value from a list of numbers based on the index number supplied. **Syntax** CHOOSE (index, item_list) • index is a number that refers to an item in item_list. • item_list is a list of numbers, formulas, or text index and item_list can be specified as CHOOSE(Constant/Formula/Simple Numeric Node, Constant/Formula/Simple Node, Constant/Formula/Simple Node …) CHOOSE(Constant/Formula/Simple Numeric Node, Simple Repeated Node) CHOOSE(Constant/Formula/Simple Numeric Node, Simple Node in Complex Repeated)
CODE()	Returns a numeric code representing the first character of the supplied string. **Syntax** CODE(Simple Numeric Node) CODE(Formula returning number) CODE(Constant)
CONCATENATE()	Joins several text strings into one string.

	Syntax CONCATENATE(Simple String Node/Constant/Formula returning string, Simple String Node/Constant/Formula returning string …) CONCATENATE(Repeated Simple String Node) CONCATENATE(Simple String Node in Complex Repeated)
COS()	Returns the cosine of a number **Syntax** COS (Formula returning Number) COS (Constant) COS (Simple Numeric Node)
COSH()	Returns the hyperbolic cosine of a number **Syntax** COSH (Formula returning Number) COSH (Constant) COSH (Simple Numeric Node)
COUNT()	Returns the count of Numeric values (Referenced or Constants) in the supplied list **Syntax** COUNT(Simple Node/Formula/Constant, Simple Node/Formula/Constant…) COUNT(Simple Repeated) COUNT(Simple Node in Complex Repeated) *Null values are **not** counted, i.e. reference variables which are not initialized are not counted.*
COUNTIF()	Returns the number of cells within a range which meet the given criteria. **Syntax** COUNTIF (range, criteria) • range is repeated nodes that you want to count. • criteria is a Number, expression, or text that defines which nodes are counted. range and criteria can be specified as

	COUNTIF(Simple Repeated, Formula/Constant/Simple Node) COUNTIF(Simple Node in Complex Repeated, Formula/Constant/Simple Node)
DATE()	Returns the serial number of the supplied date. **Syntax** DATE (year, month, day) • year is a number from 1900 to 2078. I • month is a number representing the month (for example, 12 represents December). If a number greater than 12 is supplied, the number is added to the first month of the specified year. • day is a number representing the day of the month. If the number you specify for day exceeds the number of days in that month, the number is added to the first day of the specified month. year, month and date can be specified as DATE(Simple Numeric Node/Constant/Formula, Simple Numeric Node/Constant/Formula, Simple Numeric Node/Constant/Formula) **Note: Dates before or on 29 February, 1900 are considered as invalid dates*
DATEVALUE()	Returns the serial number of a date supplied as a text string. **Syntax** DATEVALUE (text) • text is a date in text format between January 1, 1900, and December 31, 2078. If you omit the year, the current year is used. text can be specified as DATEVALUE(Simple String Node) DATEVALUE(Constant) DATEVALUE(Formula) **Note: DATEVALUE function will only support string type schema elements as input and does not support date or datetime type elements.*
DAY(),MONTH(),WEEKDAY(), YEAR()	Returns the DAY/MONTH/WEEKDAY/YEAR of the date represented by the supplied number

	Syntax XXXX(Simple String Node/Simple Numeric Node/Simple Datetime Node); XXXX(Constant) XXXX(Formula) where XXXX is replaced by the appropriate function name
HOUR(), MINUTE(),SECOND()	Returns the HOUR/MINUTE/ SECOND of the date represented by the supplied number **Syntax** XXXX(Simple Numeric Node/Simple Datetime Node); XXXX(Constant) XXXX(Formula) where XXXX is replaced by the appropriate function name
DAYS360()	Returns the number of days between two dates based on a 360-day year (twelve 30-day months). Use this function to help compute payments if your accounting system is based on twelve 30-day months. **Syntax** DAYS360 (start_date, end_date, [method]) • start_date and end_date are the two dates between which you want to know the number of days. • method is a logical value that specifies whether the European or US method should be used in the calculation. If False (or omitted), the US (NASD) method is used. If True, the European method is used. The default is based on the local translation. Where parameters can be specified as DAYS360(Simple String Node/Simple Numeric Node/Simple Datetime Node/Formula/Constant, Simple String Node/Simple Numeric Node/Simple Datetime Node/Formula/Constant, Simple Boolean Node/Formula/Constant) If method is set to False and start_date is the 31st of a month, it becomes equal to the 30th of the same month.

	If end_date is the 31st of a month and start_date is less than the 30th of a month, the ending date becomes equal to the 1st of the next month, otherwise the ending date becomes equal to the 30th of the same month.
DB()	Returns the real depreciation of an asset for a specific period of time using the fixed-declining balance method. **Syntax** DB (cost, salvage, life, period [, months]) • cost is the initial cost of the asset. • salvags is the salvage value of the asset. • life is thehe number of periods in the useful life of the asset. • period is the period for which to calculate the depreciation. The time units used to determine period and life must match. • months is the number of months in the first year of the item's life. Omitting this argument assumes there are 12 months in the first year. Where the parameters can be specified as DB(Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant)
DDB()	Returns the real depreciation of an asset for a specific period of time using the fixed-declining balance method. **Syntax** DDB (cost, salvage, life, period [, factor]) • cost is the initial cost of the asset. • salvags is the salvage value of the asset. • life is thehe number of periods in the useful life of the asset. • period is the period for which to calculate the depreciation. The time units used to determine period and life must match. • factor is the the rate at which the balance declines. Omitting this argument assumes a default factor of 2, the double-declining balance factor.

	Where the parameters can be specified as DDB(Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant)
DOLLAR() /USDOLLAR()	Returns the specified number as text, using currency format and the supplied precision. **Syntax** DOLLAR (number [, precision]) • number is a number in the the number to convert. • Precision is a value representing the number of decimal places to the right of the decimal point. Omitting this argument assumes the standard number of decimal places for the local currency. number and precision can be specified as DOLLAR(Simple Numeric Node/Formula/Constant, [Simple Numeric Node/Formula/Constant])
EVEN()	Rounds the specified number up to the nearest even integer **Syntax** EVEN (Formula returning Number) EVEN (Constant) EVEN (Simple Numeric Node)
EXACT()	Compares two expressions for identical, case-sensitive matches. True is returned if the expressions are identical; False is returned if they are not. **Syntax** EXACT(Simple String Node/Formula/Constant, Simple String Node/Formula/Constant)
EXP()	Returns e raised to the specified power. The constant e is 2.71828182845904 (the base of the natural logarithm). **Syntax** EXP (Formula returning Number) EXP (Constant) EXP (Simple Numeric Node)
FACT()	Returns the factorial of a specified number **Syntax** FACT (Formula returning Number) FACT (Constant)

	FACT (Simple Numeric Node)
FALSE(), TRUE()	Returns the logical value FALSE or TRUE **Syntax** FALSE() TRUE()
FIND()	Searches for a string of text within another text string and returns the character position at which the search string first occurs. **Syntax** FIND (search_text, text [, start_position]) • search_text is the text to find. If you specify an empty string (" "), FIND matches the first character in text. • text is the text to be searched. • start_position is the character position in text where the search begins. The first character in text is character number 1. When you omit this argument, the default starting position is character number 1. where the parameters can be specified as; Find(Simple String Node/Formula/Constant, Simple String Node/Formula/Constant, Simple Numeric Node/Formula/Constant) FIND is case-sensitive. You cannot use wildcard characters in the search_text.
FIXED()	Rounds a number to the supplied precision, formats the number in decimal format, and returns the result as text. **Syntax** FIXED (number [, precision][, no_commas]) • number is any number • precision is the number of digits that appear to the right of the decimal place. When this argument is omitted, a default precision of 2 is used. If you specify negative precision, number is rounded to the left of the decimal point. You can specify a precision as great as 127 digits. • no_commas determines if thousands separators (commas) are used in the result. Use 1 to exclude commas in the result. If no_commas is 0 or the argument is omitted, thousands separators are

	included (for example, 1,000.00). where the parameters can be specified as; FIXED(Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Boolean Node/Formula/Constant)
FLOOR()	Rounds a number down to the nearest multiple of a specified significance. **Syntax** FLOOR (number, significance) • Number is the value to round. • Significance is the multiple to which to round. where the parameters can be specified as FLOOR(Simple Numeric Node/Formula/Constant, Simple Numeric Node/Constant)
FV()	Returns the future value of an annuity based on regular payments and a fixed interest rate. **Syntax** FV (interest, nper, payment [, pv] [, type]) • interest is the fixed interest rate. • nper is the number of payments in an annuity. • payment is the fixed payment made each period. • pv is the present value, or the lump sum amount, the annuity is currently worth. When you omit this argument, a present value of 0 is assumed. • type ndicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. When you omit this argument, 0 is assumed where the parameters can be specified as FV(Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant)
IF()	Tests the condition and returns the specified value. **Syntax**

	IF (condition, true_value, false_value)
	IF (condition, true_value)
	
	• condition is any logical expression.
	• true_value is the value to be returned if condition evaluates to True.
	• false_value is the value to be returned if condition evaluates to False.
	Note: If false_value is missing by default false is considered here.
	where the parameters can be specified as
	IF(Logical Expression, Simple Node/Formula/Constant, Simple Node/Formula/Constant)

INT()	Rounds the supplied number down to the nearest integer. **Syntax** INT (Formula returning Number) INT (Constant) INT (Simple Numeric Node)
IPMT()	Returns the interest payment of an annuity for a given period, based on regular payments and a fixed periodic interest rate. **Syntax** IPMT (interest, per, nper, pv, [fv], [type]) • interest is the fixed periodic interest rate. • per is the period for which to return the interest payment. This number must be between 1 and nper. • nper is the number of payments. • pv is the present value, or the lump sum amount the annuity is currently worth. • fv is the future value, or the value after all payments are made. If this argument is omitted, the future value is assumed to be 0. • type indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. When you omit this argument, 0 is assumed. where the parameters can be specified as IPMT(

	Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant)
IRR()	Returns internal rate of return for a series of periodic cash flows. **Syntax** IRR (cash_flow [, guess]) • cash_flow is A reference to a range that contains values for which to calculate the internal rate of return. The values must contain at least one positive and one negative value. During calculation, IRR uses the order in which the values appear to determine the order of the cash flow. Text, logical values, and empty cells in the range are ignored. • guess is the estimate of the internal rate of return. If no argument is supplied, a rate of return of 10 percent is assumed. where cash_flow and guess can be specified as; IRR(Simple Numeric Repeated Node/Simple Numeric Node in Complex Repeated, Simple Numeric Node/Formula/Constant)
ISBLANK()	Return TRUE if string referenced is Empty string or is a NULL value. For rest of types it works same as ISNULL. **Syntax** ISBLANK(Simple Node[formula or Constant])
ISNULL() ISLOGICAL() ISNONTEXT() ISNUMBER() ISTEXT() ISDATE()	Return TRUE if the node returns the type else false **Syntax** ISNULL(Simple Node[formula or Constant]) ISLOGICAL(Simple Node[formula or Constant]) ISNONTEXT(Simple Node[formula or Constant]) ISNUMBER(Simple Node[formula or Constant]) ISTEXT(Simple Node[formula or Constant]) ISDATE(Simple Node[formula or Constant])
LEFT()	Returns the leftmost characters from the specified text

	string. **Syntax** LEFT (text [, num_chars]) • text is any text string • num_chars is the number of characters to return. This value must be greater than or equal to zero. If num_chars is greater than the number of characters in text, the entire string is returned. Omitting this argument assumes a value of 1. Where text and num_chars can be specified as LEFT(Simple String Node/Constant/Formula, Simple Numeric Node/Formula/Constant)
LEN()	Returns the number of characters in the supplied text string. **Syntax** LEN(Simple String Node/Formula/Constant)
LN()	Returns the natural logarithm (based on the constant e) of a number. **Syntax** LN (Formula returning Number) LN (Constant) LN (Simple Numeric Node)
LOG()	Returns the logarithm of a number to the specified base. Syntax LOG (number [, base]) • number is any positive real number. • base is the base of the logarithm. Omitting this argument assumes a base of 10. Where number and base can be specified as LOG(Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant
LOG10()	Returns the base-10 logarithm of a number. **Syntax** LOG10 (Formula returning Number) LOG10 (Constant) LOG10 (Simple Numeric Node)

LOOKUP()	Searches for a value in a repeated or complex repeated type and returns its corresponding vale in result range. **Syntax** LOOKUP (lookup_range, lookup_value, result_range); • lookup_range is the value to look for • lookup_value is the schema node in which to search for the value • range_result return value where the parameters can be specified as LOOKUP(Simple Repeated Node/Simple Node in Complex Repeated/ Simple node**,** Simple Node/Constant/Formula**,** Simple Repeated Node/Simple Node in Complex Repeated/Simple Node) *LOOKUP() is being provided as an alternate to Spreadsheet's LOOKUP(), HLOOKUP() and VLOOKUP() which have been deprecated in the formula engine*
LOWER()	Changes the characters in the specified string to lowercase characters. Numeric characters in the string are not changed. **Syntax** LOWER (Simple String Node/Formula/Constant)
MAX(), MIN()	Returns the largest/smallest value in the specified list of numbers. **Syntax** MAX(Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant…) MAX(Numeric Repeated MAX(Simple Numeric Node in Complex Repeated) MIN(Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant…) MIN (Numeric Repeated MIN (Simple Numeric Node in Complex Repeated) In the case of lists, maximum number of values that can be specified is 30
MID()	Returns the specified number of characters from a text string, beginning with the specified starting position. **Syntax**

	MID (text, start_position, num_chars) • Text is the string from which to return characters. • start_position is the position of the first character to return from text. If start_position is 1, the first character in text is returned.If start_position is greater than the number of characters in text, an empty string (" ") is returned. • num_chars is the number of characters to return. Where the parameters can be specified as; MID(Simple String Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant)
MIRR()	Returns the modified internal rate of return for a series of periodic cash flows. **Syntax** MIRR (cash_flows, finance_rate, reinvest_rate) • cash_flow is a reference to a range that contains values for which to calculate the modified internal rate of return. The values must contain at least one positive and one negative value. Values that represent cash received should be positive; negative values represent cash paid. During calculation, MIRR uses the order in which the values appear to determine the order of cash flow.Text, logical values, and empty cells in the range are ignored. • finance_rate is the interest rate paid on money used in the cash flow. • reinvest_rate is the interest rate received on money reinvested from the cash flow. where the parameters can be specified as MIRR(Simple Numeric Repeated Nodes/Simple Numeric Node in Complex Repeated, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant)
MOD()	Returns the remainder after dividing a number by a

	specified divisor. **Syntax** MOD (number, divisor) • number is any number. • divisor is any nonzero number. Where the number and divisors can be specified by; MOD(Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant)
NOT()	Returns a logical value that is the opposite of its value. **Syntax** NOT(Simple Boolean Node/Formula/Constant)
NOW()	Returns the current Date and Time **Syntax** NOW()
NPER()	Returns the number of periods of an investment based on regular periodic payments and a fixed interest rate. **Syntax** NPER (interest, pmt, pf [, fv] [, type]) • interest is the fixed interest rate. • pmt is the fixed payment made each period. Generally, pmt includes the principle and interest, not taxes or other fees. • pf is the present value, the lump-sum amount that a series of future payments is currently worth. • Fv is the future value, the balance to attain after the final payment. Omitting this argument assumes a future balance of 0. • type indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. When you omit this argument, 0 is assumed. Where the parameters can be specified as NPER(Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant,

	Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant)
NPV()	Returns the net present value of an investment based on a series of periodic payments and a discount rate. Syntax NPV (discount_rate, value_list) • discount_rate is the rate of discount for one period. • value_list is a list of that contains values that represent payments and income.During calculation, NPV uses the order in which the values appear to determine the order of cash flow.Numbers, empty cells, and text representations of numbers are included in the calculation. Errors and text that cannot be translated into numbers are ignored. If value_list is a range reference, only numeric data in the range is included in the calculation. Other types of data in the range, such as empty cells, logical values, text, and error values are ignored where discount_rate and value_list can be specified as; NPV(Simple Numeric Node/Formula/Constant, Simple Numeric Repeated Node/Simple Numeric Node in complex repeated)
ODD()	Rounds the specified number up to the nearest odd integer **Syntax** ODD (Formula returning Number) ODD (Constant) ODD (Simple Numeric Node)
OR()	Returns TRUE if at least one of a series of logical arguments is true. **Syntax** OR (Simple Boolean Node/Formula/Constant, Simple Boolean Node/Formula/Constant ...) OR (Repeated Simple Boolean Node) OR (Simple Boolean Nodes in Complex Repeated)

PI()	Returns the value of pi (p), which is approximately 3.14159265358979 when calculated to 15 significant digits. **Syntax** PI ( )
PMT()	Returns the periodic payment of an annuity, based on regular payments and a fixed periodic interest rate. **Syntax** PMT (interest, nper, pv [, fv] [, type]) • interest is the fixed periodic interest rate. • nper is the number of periods in the annuity. • pv is the present value, or the amount the annuity is currently worth. • fv is the future value, or the amount the annuity will be worth. When you omit this argument, a future value of 0 is assumed. • Type indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. When you omit this argument, 0 is assumed. where the parameters can be specified as PMT (Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant)
PPMT()	Returns the principle paid on an annuity for a given period. **Syntax** PPMT (interest, nper, pv [, fv] [, type]) • interest is the fixed periodic interest rate. • per is the period for which to return the principle. • nper is the number of periods in the annuity. • pv is the present value, or the amount the annuity is currently worth. • fv is the future value, or the amount the annuity

	will be worth. When you omit this argument, a future value of 0 is assumed. • Type indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. When you omit this argument, 0 is assumed. where the parameters can be specified as PPMT (Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant)
PRODUCT()	Multiplies a list of numbers and returns the result. **Syntax** PRODUCT(Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant…) PRODUCT(Simple Numeric Nodes Repeated) PRODUCT(Simple Numeric Nodes in Complex Repeated)
PROPER()	Returns the specified string in proper-case format. In proper-case format, the first alphabetic character in a word is capitalized. If an alphabetic character follows a number, punctuation mark, or space, it is capitalized. All other alphabetic characters are lowercase. Numbers are not changed by PROPER **Syntax** PROPER(Simple String Node/Formula/Constant)
PV()	Returns the present value of an annuity, considering a series of constant payments made over a regular payment period. **Syntax** PV (interest, nper, payment [, fv] [, type]) • interest is the fixed interest rate. • nper is the number of payments in an annuity. • payment is the fixed payment made each period. • fv is he future value, or the amount the annuity will be worth. When you omit this argument, a

	future value of 0 is assumed. • type ndicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. When you omit this argument, 0 is assumed where the parameters can be specified as PV(Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant)
RAND()	Returns a number selected randomly from a uniform distribution greater than or equal to 0 and less than 1. **Syntax** RAND()
RATE()	Returns the interest rate per period of an annuity, given a series of constant cash payments made over a regular payment period. **Syntax** RATE(nper, payment, pv [, fv] [, type] [, guess]) • nper is the number of payments in an annuity. • payment is the fixed payment made each period. Generally, payment includes only principle and interest, not taxes or other fees • pv is the present value of annuity • fv is he future value, or the amount the annuity will be worth. When you omit this argument, a future value of 0 is assumed. • type ndicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. When you omit this argument, 0 is assumed • guess is your estimate of the interest rate. If no argument is supplied, a value of 0.1 (10 percent) is assumed where the parameters can be specified as PV(Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant,

	Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant)
REPLACE()	Replaces part of a text string with another text string. Syntax REPLACE (orig_text, start_position, num_chars, repl_text) • orig_text is the original text string. • start_position is the character position where the replacement begins. If start_position is greater than the number of characters in orig_text, repl_text is appended to the end of orig_text. • num_chars is the number of characters to replace. If this argument is negative, #VALUE! is returned. • repl_text is the replacement text string. where the parameters can be specified as; REPLACE(Simple String Node/Constant/Formula, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple String Node/Formula/Constant)
REPT()	Repeats a text string the specified number of times. **Syntax** REPT (text, number) • text is any text string. • number is the number of times you want text to repeat. If number is 0, empty text (" ") is returned. Where text and number can be specified as; REPT(Simple String Node/Formula/Constant, Simple Numeric Node/Formula/Constant
RIGHT()	Returns the rightmost characters from the specified text string. **Syntax** RIGHT (text [, num_chars])

	• text is any text string • num_chars is the number of characters to return. This value must be greater than or equal to zero. If num_chars is greater than the number of characters in text, the entire string is returned. Omitting this argument assumes a value of 1. Where text and num_chars can be specified as RIGHT(Simple String Node/Constant/Formula, Simple Numeric Node/Formula/Constant)
ROUND()	Rounds the given number to the supplied number of decimal places. **Syntax** ROUND (number, significance) • number is the value to round. • significance is the multiple to which to round. number and significance can be specified as ROUND (Constant/Formula/Simple Numeric Node, Constant/Formula/Simple Numeric Node)
ROUNDDOWN()	Rounds a number down. **Syntax** ROUNDDOWN (number, numberOfDigits) • number is the value to round. • numberOfDigits is the number of decimal places to which number is rounded. When a negative precision is used, the digits to the right of the decimal point are dropped and the absolute number of significant digits specified by precision are replaced with zeros.If precision is 0, number is rounded down to the nearest integer number and numberOfDigits can be specified as ROUNDDOWN (Constant/Formula/Simple Numeric Node, Constant/Formula/Simple Numeric Node)
ROUNDUP()	Rounds a number down. **Syntax** ROUNDDOWN (number, numberOfDigits) • number is the value to round. • numberOfDigits is the number of decimal places

	to which number is rounded. When a negative precision is used, the digits to the right of the decimal point are dropped and the absolute number of significant digits specified by precision are replaced with zeros.If precision is 0, number is rounded up to the nearest integer. number and numberOfDigits can be specified as ROUNDUP (Constant/Formula/Simple Numeric Node, Constant/Formula/Simple Numeric Node)
SEARCH()	Locates the position of the first character of a specified text string within another text string.. **Syntax** SEARCH (search_text, text [, start_position]) • search_text is he text to find. To search for an asterisk or question mark, include a tilde (~) before the character.The search string can contain wildcard characters. The available wildcard characters are * (asterisk), which matches any sequence of characters, and ? (question mark), which matches any single character. • text is the text to be searched. • start_position is the character position in text where the search begins. The first character in text is character number 1. When you omit this argument, the default starting position is character number 1. where the parameters can be specified as; SEARCH(Simple String Node/Formula/Constant, Simple String Node/Formula/Constant, Simple Numeric Node/Formula/Constant) SEARCH is not case-sensitive.
SIGN()	Determines the sign of the specified number. SIGN returns 1 if the specified number is positive, -1 if it is negative, and 0 if it is 0. **Syntax** SIGN (Formula returning Number) SIGN (Constant) SIGN (Simple Numeric Node)
SIN()	Returns the sine of a number

	Syntax SIN (Formula returning Number) SIN (Constant) SIN (Simple Numeric Node)
SINH()	Returns the hyperbolic sine of a number **Syntax** SINH (Formula returning Number) SINH (Constant) SINH (Simple Numeric Node)
SLN()	Returns the depreciation of an asset for a specific period of time using the straight-line balance method. Syntax SLN(cost, salvage, life) • cost is the initial cost of the asset • salvage is the salvage value of the asset • life is the number of periods of the useful life of the asset where cost, salvage and life can be specified as SLN(Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant)
SQRT()	Returns the square root of a number **Syntax** SQRT (Formula returning Number) SQRT (Constant) SQRT (Simple Numeric Node)
STDDEV()	Returns the standard deviation of a population based on a sample of supplied values. The standard deviation of a population represents an average of deviations from the population mean within a list of values. **Syntax** STDDEV (Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant ...) STDDEV (Repeated Simple Numeric Node) STDDEV (Simple Numeric Nodes in Complex Repeated)
STDDEVP()	Returns the standard deviation of a population based on an entire population of values. The standard deviation of

	a population represents an average of deviations from the population mean within a list of values. **Syntax** STDDEVP (Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant ...) STDDEVP (Repeated Simple Numeric Node) STDDEVP (Simple Numeric Nodes in Complex Repeated)
SUBSTITUTE	Replaces a specified part of a text string with another text string. Syntax SUBSTITUTE (text, old_text, new_text [, instance]) • text is a text string that contains the text to replace. You can also specify a reference to a cell that contains text. • old_text is the text string to be replaced. • new_text is the replacement text. • instance specifies the occurrence of old_text to replace. If this argument is omitted, every instance of old_text is replaced. where the parameters can be specified as; SUBSTITUTE(Simple String Node/Formula/Constant, Simple String Node/Formula/Constant, Simple String Node/Formula/Constant, Simple Numeric Node/Formula/Constant)
SUM()	Returns the sum of the supplied numbers. **Syntax** SUM (Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant ...) SUM (Repeated Simple Numeric Node) SUM (Simple Numeric Nodes in Complex Repeated)
SUMIF()	Returns the sum of the specified numbers based on the given criteria. **Syntax** SUMIF (range, criteria, sum_range)

	• range is the numbers you want evaluated. • criteria is a number, expression, or text that defines which cells are added. For example, criteria can be expressed as 15, "15", ">15", "cars". • sum_range are the actual numbers to sum. These are only summed if their corresponding cells in range match the criteria. If this argument is omitted, the numbers in range are summed. Where the parameters can be specified as; SUMIF(Simple Numeric Nodes Repeated/Simple Numeric Nodes in Complex Repeated, Formula, Simple Numeric Nodes Repeated/Simple Numeric Nodes in Complex Repeated)
SUMPRODUCT()	Multiplies the corresponding numbers in the given nodes, then returns the sum of those products **Syntax** SUMPRODUCT(Simple Numeric Nodes Repeated/Simple Numeric Node in Complex Repeated, Simple Numeric Nodes Repeated/Simple Numeric Node in Complex Repeated) The numbers of nodes should be the same for both parameters
SUMSQ()	Squares each of the supplied numbers and returns the sum of the squares.. **Syntax** SUMSQ (Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant ...) SUMSQ (Repeated Simple Numeric Node) SUMSQ (Simple Numeric Nodes in Complex Repeated)
SYD()	Returns the depreciation of an asset for a specified period using the sum-of-years method. This depreciation method uses an accelerated rate, where the greatest depreciation occurs early in the useful life of the asset. **Syntax**

	SYD (cost, salvage, life, period) • cost is the initial cost of the asset. • salvage is the salvage value of the asset. • life is the number of periods in the useful life of the asset. • period is the period for which to calculate the depreciation. The time units used to determine per and life must match. where the parameters can be specified as; SYD(Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant)
TAN()	Returns the tangent of the specified angle **Syntax** TAN (Formula returning Number) TAN (Constant) TAN (Simple Numeric Node)
TANH()	Returns the hyperbolic tangent of the specified number **Syntax** TANH (Formula returning Number) TANH (Constant) TANH (Simple Numeric Node)
TEXT()	Returns the given number as text, using the specified formatting. Syntax TEXT (number, format) number is any number format is a string representing a number format. The format must be surrounded by a set of double quotation marks. Asterisks cannot be included in format. Where number and format can be specified as TEXT(Simple Numeric Node/Simple Date Node/Formula/Constant, Simple String Node/Constant/Formula)

	The following is a list of all format IDs and format types: **ID Format Types** 0 $1,234 1 1234 2 1,234 3 0% 4 1234.56E+12 5 1/3/2002 6 Thursday, January 03, 2002 7 Thursday, January 03, 2002 12:00 AM 8 Thursday, January 03, 2002 12:00:00 AM 9 1/3/2002 12:00 AM 10 1/3/2002 12:00:00 AM 11 January 03 12 Thu, 03 January 2002 00:00:00 GMT 13 2002-01-03T00:00:00 14 12:00 AM 15 12:00:00 AM 16 2002-01-03 00:00:00 Z 17 January, 2002 18 P2007Y7M30DT10H51M45S [this format is the XML representation of a duration value]
TIME()	Returns a serial number for the supplied time. Syntax TIME (hour, minute, second) • hour is a number from 0 to 23. • minute is a number from 0 to 59. • second is a number from 0 to 59. where hour, minute and seconds can be specified as; TIME(Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant)
TIMEVALUE()	Returns a serial number for the supplied text representation of time. **Syntax** TIMEVALUE(Simple String Node/Formula/Constant) **Note: TIMEVALUE function will only support string type schema elements as input and does not support date or datetime*

	type elements.
TODAY()	Returns the current date as a serial number. **Syntax** TODAY ( )
TRIM()	Removes all spaces from text except single spaces between words. **Syntax** TRIM (Simple String Node/Formula/Constant)
TRUE()	Returns the logical TRUE value **Syntax** TRUE()
TRUNC()	Truncates the given number to an integer. **Syntax** TRUNC (number, precision) • number is the value to truncate. • precision is the number of decimal places allowed in the truncated number. Omitting this argument assumes a precision of 0. number and precision can be specified as TRUNC (Constant/Formula/Simple Numeric Node, Constant/Formula/Simple Numeric Node)
UPPER()	Changes the characters in the specified string to upper characters. Numeric characters in the string are not changed. **Syntax** UPPER (Simple String Node/Formula/Constant)
VALUE()	Returns the specified text as a number. **Syntax** VALUE(Simple String Node/Formula/Constant)
VAR()	Returns the variance of a population based on a sample of values. **Syntax** VAR (Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant ...) VAR (Repeated Simple Numeric Node) VAR (Simple Numeric Nodes in Complex Repeated)

VARP()	Returns the variance of a population based on an entire population of values **Syntax** VARP (Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant ...) VARP (Repeated Simple Numeric Node) VARP (Simple Numeric Nodes in Complex Repeated)
VDB()	Returns the depreciation of an asset for a specified period using a variable method of depreciation. Syntax VDB (cost, salvage, life, start_period, end_period [, factor] [, method]) • cost is the initial cost of the asset. • salvage is the salvage value of the asset. • life is the number of periods in the useful life of the asset. • start_period is the beginning period for which to calculate the depreciation is the time units used to determine start_period and life must match. • end_period is the ending period for which to calculate the depreciation. The time units used to determine end_period and life must match. • factor is the rate at which the balance declines. Omitting this argument assumes a default of 2, which is the double-declining balance factor. • method is a logical value that determines if you want to switch to straight-line depreciation when depreciation is greater than the declining balance calculation. Use True to maintain declining balance calculation; use False or omit the argument to switch to straight-line depreciation calculation. where the parameters can be specified as VDB(Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Numeric Node/Formula/Constant, Simple Boolean Node/Formula/Constant

	)