Amtiri

AMTIRI SCRIPT

Function index

Generated on November 25, 2026

Every function and syntax entry of Amtiri Script, from A to Z. Each one links to its entry, with its signature, its parameters and tested examples.

Token	Name	Description
`!=`	Not equal (C style)	True when the two values differ.
`&&`	And (short-circuit)	True only when both sides are true.
`*`	Multiply	Multiplies numbers.
`+`	Add	Adds numbers.
`-`	Negate	Flips the sign of a number and keeps its kind: a whole number stays whole and a decimal stays decimal.
`-`	Subtract	Subtracts the right value from the left one.
`.`	Field	Reads one named field out of a record - a control function's state, a database row, an inline object.
`/`	Divide	Divides the left value by the right one.
`<`	Less than	True when the left value sorts before the right one.
`<=`	Less than or equal	True when the left value sorts before the right one, or matches it.
`<>`	Not equal	True when the two values differ.
`==`	Equals	True when both values match.
`>`	Greater than	True when the left value sorts after the right one.
`>=`	Greater than or equal	True when the left value sorts after the right one, or matches it.
`[]`	Item at	Reads one item out of a list by its position, counting from zero.
`^`	Power	Raises the left value to the power of the right one.
`||`	Or (short-circuit)	True when either side is true.
`addDays`	Add days	Shifts a date by a number of days.
`addHours`	Add hours	Shifts a date and time by a number of hours.
`addMinutes`	Add minutes	Shifts a date and time by a number of minutes.
`addMonths`	Add months	Shifts a date by a whole number of months.

`addWeeks`	Add weeks	Shifts a date by a number of weeks.
`addYears`	Add years	Shifts a date by a whole number of years.
`and`	And	True only when every condition given is true.
`array`	List	An ordered list written inline, [a, b, c].
`arrayAdd`	Append	Gives a new list with values appended to the end, leaving the original list untouched.
`arrayAverage`	Average	The arithmetic mean of a list's items.
`arrayConcat`	Concatenate	Joins two lists into a new one, leaving both originals untouched.
`arrayFilter`	Filter	Keeps only the items a condition accepts.
`arrayMax`	List maximum	The largest item of a list, or null when the list is empty.
`arrayMin`	List minimum	The smallest item of a list, or null when the list is empty.
`arraySum`	Sum	Adds up every item of a list.
`bit`	Bit test	Tests one bit of a whole number - how a packed status word from a PLC or Modbus device is split into individual alarms.
`case`	Case	Compares one value against a series of candidates and yields the matching result - a lookup table.
`ceil`	Ceiling	Rounds up, or up to the nearest multiple of module when it is given.
`clamp`	Clamp	Holds a value inside a range: anything below min reads as min, anything above max reads as max.
`closePanel`	Close panel	Takes away the panel occupying this id on the current screen, wherever it was opened from.
`cos`	Cosine	The cosine of an angle given in radians.
`count`	Count	How many items a list holds.
`ctrlKalman`	Kalman filter	Deprecated: use the Kalman module.
`ctrlPID`	PID controller	Deprecated: use the PID module.
`ctrlPWM`	PWM controller	Deprecated: use the PWM module.
`darker`	Darker	Darkens a colour by a factor, towards black.
`date`	Date	Builds a date and time from its parts.
`date_of`	Date part	Drops the time of day, giving the same date at midnight.
`dayOfMonth`	Day of month	The day part of a date, starting at 1.
`dayOfWeek`	Day of week	Which day of the week a date falls on, Monday being 1.
`delayedSet`	Delayed set	Writes a value into a variable once a delay has passed.
`double`	Decimal	A decimal number.
`dowhile`	Do while	Repeats an expression for as long as a condition holds, testing after each pass - so it always runs at least once.
`evl`	Error default	Substitutes a default when evaluating a value raises an error, instead of letting the whole script fail.

`exec`	Sequence	Evaluates several expressions in order and yields the last one's result.
`exp`	Exponential	e raised to the power of value - the inverse of ln.
`fact`	Factorial	The factorial of a whole number, value!.
`false`	False	The boolean a comparison yields when it does not hold.
`floor`	Floor	Rounds down, or down to the nearest multiple of module when it is given.
`for`	For	Repeats an expression over a counter: the counter is set once, tested before each pass and advanced after it.
`forEach`	For each	Converts every item of a list.
`format`	Format	Builds a text from a pattern and its arguments, using Java format specifiers - %s for text, %d for a whole number, %.2f for two decimals.
`functions`	List functions	Every function name registered on the environment, sorted - an interpreter introspection aid.
`gadient`	Gradient (old spelling)	The old, misspelt name of gradient, still accepted so stored formulas keep working.
`gradient`	Gradient	Picks the colour a named gradient assigns to a value - the usual way to drive an indicator from a reading.
`history`	History	Every sample a logger recorded for one variable over a time window, as a time series ready for the tsXxx functions.
`hsl`	HSL colour	Builds a colour from hue, saturation and lightness - easier than RGB when a colour has to vary smoothly with a reading.
`if`	If	Chooses between two values on a condition.
`inn`	Not-a-number default	Substitutes a default when a value is not a number: NaN, which is what a square root of a negative reading produces, or anything that is not numeric at all, such as text or null.
`integer`	Number	A whole number written straight into the script.
`ivl`	Infinite default	Substitutes a default when a value came out infinite - the guard to pair with a division whose divisor can reach zero.
`length`	Length	How many items a list or record holds, or how many characters a text has.
`lighter`	Lighter	Lightens a colour by a factor, towards white.
`ln`	Natural logarithm	The logarithm of a value in base e.
`log`	Logarithm (base 10)	The logarithm of a value in base 10.
`loggerValueAt`	Value at	What one variable's value was at a given instant, according to a logger: the last value it logged at or before that instant, of the type it was logged with.
`loggerValuesAt`	Values at	What several variables' values were at a given instant, according to a logger - loggerValueAt for each of them, one read of the logger per variable, gathered into one record.
`max`	Maximum	The largest of the values given.
`mbDouble`	Double	Decodes a 64-bit IEEE 754 floating-point value from four consecutive registers, the first holding the highest word (byte order ABCDEFGH).

`mbFloat`	Float	Decodes a 32-bit IEEE 754 floating-point value from two consecutive registers, the first holding the high word (byte order ABCD, as the Modbus master's own float register type): mbFloat([16256, 0], 0) is 1.0.
`mbFloatHL`	Float (high /low)	Decodes a 32-bit IEEE 754 floating-point value from two consecutive registers, the first holding the high word, with the two bytes of each register swapped (byte order BADC): mbFloatHL([32831, 0], 0) is 1.0.
`mbInt`	Int	Decodes a signed 32-bit whole number from two consecutive registers, the first holding the low word (word order CDAB): mbInt([0, 1], 0) is 65536.
`mbIntHL`	Int (high /low)	Decodes a signed 32-bit whole number from two consecutive registers, the first holding the high word, with the two bytes of each register swapped (byte order BADC): mbIntHL([256, 0], 0) is 65536.
`min`	Minimum	The smallest of the values given.
`mod`	Modulus	The remainder of dividing value by base - a wrapped counter, or an every-nth test.
`month`	Month	The month part of a date, 1 to 12.
`not`	Not	Inverts a condition.
`now`	Now	The current date and time in UTC.
`null`	Null	The absence of a value.
`nvl`	Null default	Substitutes a default when a value is null.
`object`	Object	A record of named fields written inline, {key: value}.
`openDashboard`	Open dashboard	Replaces what the current view is showing with another dashboard - how an Action button navigates the operator somewhere.
`openPanel`	Open panel	Shows a panel over the current view, centred at an offset from the component that ran this script.
`or`	Or	True when at least one condition given is true.
`psplit`	Period split	Cuts a time window into equal consecutive segments - the segment list tsSplit expects, and the basis of any bucketed aggregate.
`random`	Random	A fresh pseudo-random number between 0 inclusive and 1 exclusive on every evaluation.
`rgb`	RGB colour	Builds a colour from red, green and blue, each a fraction from 0 to 1: rgb(1, 0, 0) is pure red, and values of 1 and above are capped.
`round`	Round	Rounds to the nearest whole number, or to the nearest multiple of module when it is given - round(x, 0.5) snaps to halves.
`scale`	Scale	Maps a value from one range onto another - the usual way to turn a raw sensor reading into engineering units.
`scalein`	Scale (confined)	Maps a value from one range onto another exactly as scale does, then confines the result to the destination range, whichever way round it runs - an out-of-range reading cannot drive an output past its limits.
`serie`	Number range	Builds a list of consecutive whole numbers.
`sin`	Sine	The sine of an angle given in radians.
`sqrt`	Square root	The square root of a value.
`string`	Text	A fixed piece of text, in double quotes.
`substr`	Substring	The part of a text between two positions, counting from 0 and excluding the character at to.

`tan`	Tangent	The tangent of an angle given in radians.
`to_double`	To decimal	Converts a value to a decimal number.
`to_int`	To whole number	Converts a value to a whole number, truncating any decimals.
`to_lower`	To lower case	Converts a text to lower case.
`to_string`	To text	Converts a value to its text form.
`to_upper`	To upper case	Converts a text to upper case.
`toDate`	From day number	Turns a day number counted from 1899-12-31 back into a date and time - the inverse of toDays.
`toDays`	To day number	A date as a decimal count of days since 1899-12-31, the time of day as its fraction, so a duration becomes a subtraction.
`toLocale`	To local time	Reads a UTC date and time as the configured time zone.
`toUTC`	To UTC	Reads a date and time expressed in the configured time zone back as UTC - the inverse of toLocale, and what to apply to any date a person entered.
`toUTM`	To UTM	Converts a latitude and longitude to Universal Transverse Mercator coordinates - metres, so distances between two points can be worked out by subtraction.
`trim`	Trim	Removes leading and trailing whitespace from a text.
`true`	True	The boolean a comparison yields when it holds.
`tsAverage`	Time average	The time-weighted mean of a series - a sample held for an hour counts for an hour, not as one reading, which is what makes this the correct average for logged data.
`tsCenter`	Middle value	The value in force halfway through a series' time span.
`tsDistribution`	Distribution	How long a series spent at each distinct value - the run-time-per-state breakdown behind a machine-state or alarm report.
`tsFilter`	Filter series	Keeps only the samples a condition accepts, breaking the series where a run of rejected samples begins.
`tsFirst`	First value	The first sample of a series, as its [time, value] pair: tsFirst(s)[1] is the value, tsFirst(s)[0] its time.
`tsLast`	Last value	The last sample of a series, as its [time, value] pair: tsLast(s)[1] is the value, tsLast(s)[0] its time.
`tsMax`	Series maximum	The largest value in a series.
`tsMerge`	Merge	Interleaves several named series onto one timeline, so every instant carries the last known value of each - what a multi-variable chart or export needs.
`tsMin`	Series minimum	The smallest value in a series.
`tsSplit`	Split	Cuts a series into one sub-series per time segment - pair it with psplit to bucket a window, then aggregate each bucket.
`type_of`	Type of	The name of a value's underlying type, for diagnosing a script that is receiving something other than what it expects.

`variables`	List variables	Every variable name defined on the environment, sorted - an interpreter introspection aid.
`while`	While	Repeats an expression for as long as a condition holds, testing before each pass.
`with`	With	Evaluates an expression with a value bound to a name - a local alias, so an expensive sub-expression is written and computed once.
`xor`	Exclusive or	True when an odd number of the conditions given are true.
`year`	Year	The year part of a date.

This page describes Amtiri Script 5.5.0 as shipped with Data Orchester engine 6.22.0 and Site 1.27.0.