

History

Generated on October 14, 2026

History

History

```
history · history(logger: name, variable: string, timeFrom: DateTime, timeTo: DateTime)
```

Every sample a logger recorded for one variable over a time window, as a time series ready for the tsXxx functions. Inside a compacted period the samples are the simplified series: the segment's average, with a 1 ms spike to its max and one to its min where they happened.

Parameters

Name	Type	Description
logger	name	The code of the configured history logger, written bare rather than quoted - it is read as an identifier, not as text.
variable	string	The code of the variable whose samples are wanted.
timeFrom	DateTime	The start of the window, in UTC.
timeTo	DateTime	The end of the window, in UTC.

Returns `time series`: The samples in the window, oldest first.

Examples

```
history(PLANT_LOG, "LIT_101", START, addHours(START, 2)) # [[2026-09-29T08:00, 41.5], [2026-09-29T08:30, 44.0],  
PLANT_LOG.history("lit_101", START, addMinutes(START, 45)) # [[2026-09-29T08:00, 41.5], [2026-09-29T08:30, 44.0],  
history(PLANT_LOG, "LIT_101", addDays(START, -1), addHours(addDays(START, -1), 1)) # []
```

Data Orchestrator: available in plant formulas.

Period split

```
psplit · psplit(timeFrom: DateTime, timeTo: DateTime, splits: int)
```

Cuts a time window into equal consecutive segments - the segment list tsSplit expects, and the basis of any bucketed aggregate. It reads no logger: it only divides the window it is given.

Parameters

Name	Type	Description
timeFrom	DateTime	The start of the window, in UTC.
timeTo	DateTime	The end of the window, in UTC.
splits	int	How many equal segments to cut it into; at least 1.

Returns `[]`: A list of start and end pairs covering the window without gaps.

Examples

```
psplit(START, addHours(START, 2), 2) # [[2026-09-29T08:00, 2026-09-29T09:00], [2026-09-29T09:00, 2026-09-29T10:00]]
psplit(START, addHours(START, 2), 0) # error
```

Data Orchestrator: available in plant formulas.

Value at

`loggerValueAt` · `loggerValueAt(logger: string, variable: string, date: DateTime)`

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. Inside a compacted period that is the segment's average, not a measured value.

Parameters

Name	Type	Description
logger	string	The code of the LOGGER module to read from, as text in quotes.
variable	string	The code of the variable to read, as text; upper or lower case alike.
date	DateTime	The instant to read at, in UTC.

Returns `any`: The value in force at that instant - a number stays a number - or null when the logger had logged none by then.

Examples

```
loggerValueAt("PLANT_LOG", "LIT_101", addHours(START, 1)) # 44.0
loggerValueAt("PLANT_LOG", "pump_on", START) # true
loggerValueAt("PLANT_LOG", "LIT_101", addDays(START, -1)) # null
```

Data Orchestrator: available in plant formulas.

Values at

`loggerValuesAt` · `loggerValuesAt(logger: string, variables: [], date: DateTime)`

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. Inside a compacted period each is the segment's average.

Parameters

Name	Type	Description
logger	string	The code of the LOGGER module to read from, as text in quotes.
variables	[]	The codes of the variables to read, as texts; upper or lower case alike.
date	DateTime	The instant to read at, in UTC.

Returns `object`: A record with one field per code, written as given, holding the value in force at that instant, or null when the logger had logged none by then.

Examples

```
loggerValuesAt("PLANT_LOG", ["LIT_101", "PUMP_ON"], START) # {LIT_101: 41.5, PUMP_ON: true}
```

Data Orchester: available in plant formulas.

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