InfoWorks ICM

Theory of Time Series Expressions

Technical Paper
April 2017

Innovyze®

Contents

Theory of time series expressions

The theory of time series expressions is outlined in the following sections. Because time series expressions build on (non-time series) SQL expressions, a basic familiarity with SQL expressions is assumed – see SQL in InfoWorks ICM for more details.

Time series expressions can be used in TVD Connectors in InfoWorks ICM for generating time series for input to a simulation .

In a TVD Connector expression, time series data is loaded using the TSDATA function and all time series are defined over the same time period (domain) which is usually the period of the simulation (plus an allowance for resampling). The units of measurement are specified as an argument of the TSDATA function.

In a derived stream expression, time series data is loaded using the TSINPUT function and all time series required to satisfy a user's request (e.g. for display or to answer a question) are defined over the same domain which is determined by the request (plus an allowance for resampling).

Quick review of conventional (numeric) expressions

The SQL engine can evaluate expressions composed of operators, functions and simple numeric values (and also boolean, date and string values, but these are less relevant here). It can also assign values to variables and use these variables in expressions. The value of an expression containing multiple semicolon-separated clauses is the value of the last clause, with the previous clauses used to calculate intermediate variables. Examples of expressions using various operators and functions are:

- 3 + 4
- SIN(3) – COS(4)
- SET $A = 3; SET $B = $A + 4; 2.4 * LOG($A * $B) – 3

An expression is evaluated in the context of a specific object in a network. In InfoWorks ICM, this is a TVD connector. The expression can include the names of fields of this object and these are replaced by the values stored in those fields when the expression is evaluated. It can also include fields of linked objects (although there are currently no linked objects for a TVD connector, in future it may be possible to use its "connected object" as a linked object). Examples are:

- 3 + user_number_1
- LOG(user_number_1) + LOG(user_number_5)

Time series data as a new type of value

To work with time series in expressions, we treat an individual, finite sequence of time series data points as a single value of a new type, which we call "time series data". This new type is on a par with the existing types (numeric, boolean, date, string and null) and can be used in expressions in very much the same way. Typically, the value of an expression containing time series data values will itself be a time series data value.

Note that in the discussion that follows, we are careful to distinguish between a time series data value, which is a value that encapsulates the whole of a finite period of a time series, and an individual (number or null) value taken from a particular data point within that time series.

The theory of time series expressions consists of defining the properties of time series data values and the way in which the different operators and functions act on these values to generate new values. The following sections provide these definitions.

Properties of time series data values

Time series data values used in an expression have the following properties:

- Each time series data value in an expression is defined on the same, finite, time domain defined by a time zone (e.g. "(UTC+10:00) Canberra, Melbourne, Sydney") and an absolute start and end time (e.g. 2015-01-12 04:45:00 to 2015-02-06 14:20:00).
- The time domain of a time series data value is determined by the data-loading function and/or the context in which the expression is evaluated, as follows:
    - For data loaded by the TSDATA function, the time zone is specified explicitly (in the Run options) and the start and end time are set either in the Run options (for simulation) or the Test dialog (for testing). These times are specified in local standard time (i.e. not including daylight saving)
    - For numeric values that are automatically converted into time series data values, the time domain is large enough to encompass all data already loaded via TSDATA.
- Internally, a time series data value comprises an array of pairs of [time stamp, number or null] that define a possibly-discontinuous, piece-wise linear or step function within the time domain.
- The time stamps are strictly monotonic increasing in the array.
- The start and end of the time domain are always within the range of times in the array. If necessary, data points are added at the start and/or end of the time domain to ensure this is true. Input data from a TSD will run from the first point at or before the start of the domain to the last point at or after the end of the domain.
- All time stamps are in the local standard time of the specified time zone. When sourcing data from a TSD (which uses UTC internally) a fixed offset is applied to convert to local standard time.
- The internal representation of time stamps cannot be accessed directly. If calculations based on date-time are required, the TSDATETIME function can be used to create a series whose values are the numeric equivalent of its time stamps (i.e. double-precision number of days since 00:00 30-Dec-1899) and which can be used with SQL functions that work with dates.
- The numeric values in the array are represented as double-precision numbers, with null represented by a "quiet NaN" value.
- A time series data value has an "interpolation" property that determines whether the function is linear (interpolation="linear") or step-wise (interpolation="extend" or "step"). This is used when an operator or function needs to obtain a numeric value for an intermediate time which does not appear in the array.
- The units of measurement are user-defined in the sense that the user specifies the desired units at the point that the time series data value is constructed(see the TSDATA function for details).

- A time stamp paired with a null means that the value at that time-stamp is unknown/missing. This has implications for how values between data points are interpolated and in turn affects calculations that include null values or statistics of intervals that include null periods. This is discussed further below.

How null values are generated within a time series data value

Null-valued data points can appear in a time series for any of the following reasons:

- The data comes from a TSD stream and that stream includes explicitly-entered, null-valued points.
- The data comes from a TSD stream and the gap between two data points is greater than the data interval specified for the stream. In this case, a null-valued data point is automatically inserted at the time at which the data stream becomes invalid (i.e. data_interval seconds after the previous valid data point).
- The data comes from a TSD stream and the first data in the stream is after the start of the time domain. In this case, a null-valued data point is automatically inserted at the start of the time domain.
- The data comes from a TSD stream and the last data in the stream is before the end of the time domain. In this case, a null-valued data point is automatically inserted at the end of the time domain.
- A non-time series null value (e.g. from a field of the TVD connector) is converted to a time series data value (see Conversion rules)
- An operator or function generates a null-valued data point for one of the reasons documented for that function (the most common reason being that one of the inputs is null at that time)

In general, operators and functions do not generate multiple sequential null-valued data points. Once a null-valued data point has been generated, subsequent null-valued data points are skipped until the next data point that has a non-null value is reached, or the end of the time domain is reached.

Interpolation rules and the effect of null values

As noted above, a time series defines a piece-wise linear or step function within the time domain, according to whether the interpolation property is "linear" or "extend" (note that "step" can be used as a synonym for "extend"). In the absence of nulls, this means that numeric values are well-defined at all times within the domain and that time-integrals and similar operations over arbitrary time periods can be computed.

If a data point has a null value, then values at intermediate times between this data point and the next non-null data point will be treated as null. Additionally, linear interpolation between a non-null and null value is not possible, so for a linear time series, intermediate values after the preceding data point are also treated as null. Thus null-valued data points create periods of null value within the time series.

Functions and operators that take values from specific times within such periods have their own rules about how to handle nulls, as discussed in the sections describing operators and functions. Typically, but not always, the resulting value is also null.

Functions that calculate statistics (e.g. mean, min, max) over specific intervals will usually calculate the statistic over the non-null periods in the interval, if any, returning null only if the entire interval is null. Statistics about null data are also available (e.g. countnonnull, coverage). Some statistics (e.g. count, percentile) work only on individual data points (i.e. not weighted by time). See TSRESAMPLE for more details.

Conversion rules for time series data values

A time series data value will never be automatically converted to any other type of value.

Other types of value will be automatically converted to a time series value as needed for use in an operator or function (see the following sections). The conversion rules are:

- Null value converts to a time series data value which has only a null value at the start and end times.
- Numeric value converts to a time series data value that has interpolation="extend" and two points, one at the start time with value equal to the specified number and one at the end time with a null value.
- Date value converts to its numeric equivalent, which is the number of days (and fractional days) since 00:00 30-Dec-1899.
- Other types convert first to a number (following the usual conversion rules) and then to time series data value as described above.

Network object fields available for use in time series data expressions

Any of the fields of the TVD connector that contains the expression may be used in the expression. The most commonly used fields will be the inputs (input_a, input_b, input_c) but other fields can be used. In particular the user fields can be used to hold parameters that are specific to the connector, allowing the same expression to be used in multiple connectors yet have connector-specific behaviour.

In addition, there is a linked 'run' object that has a single field called 'origin' which is a date field that holds the date/time of the run origin. Thus run.origin can be used to carry out calculations that are dependent on time before or after the run origin.

Units in time series data expressions

SQL expressions elsewhere in InfoWorks ICM are evaluated using user units. This is not ideal for time series data expressions, as these expressions will generate inputs to the simulation and those inputs must be the same regardless of which user runs the simulation.

The approach taken for time series data values and expressions is to specify the required units explicitly where needed. Thus TSDATA has an argument to specify the required units. Similarly, the TVD Connector object has a field, **Connector units**, that is used to specify the units of the expression as a whole for the purpose of converting back into the native units that are used by the simulation engine.

It is the responsibility of the user to ensure that all numeric values used in the expression are correct for the chosen units.

Constructing time series data values

Unlike a simple numeric value, which can be entered in an expression as a literal such as 3.14159 or obtained from a field of an object in the network, a time series data value can only be created as the result of an expression that involves time series data values, or as the result of a function that constructs a time series data value using parameters that are simple values. The most important such functions are the TSDATA and TSPOINTS constructor functions. See Specialised functions for use only with time series data values for more information.

Standard operators that can be used with time series data values

Any of the standard set of arithmetic, logical or comparison operators, unary or binary, may be applied to a time series data value or pair of values. The operators are:

- Unary arithmetic:
  - - (negate)
- Unary logical:
  - NOT
  - IS NULL
  - IS NOT NULL
- Binary arithmetic:
  - + (add)
  - - (subtract)
  - * (multiply)
  - / (divide)
  - ^ (raise to the power of)
  - % (modulo)
- Binary logical:
  - AND
  - OR
- Binary comparison:
  - < (less than)
  - > (greater than)
  - = (equal)
  - <> (not equal)
  - <= (less than or equal)
  - >= (greater than or equal)

A binary operator may be applied to a time series data value and a non-time series data value. The latter is first converted to a time series data value as described under conversion rules above.

The output value of an operator acting on time series data value(s) is computed as follows:

- All the distinct times associated with all data points in all of the input time series data value(s) are identified and placed in order.
- For each such time:
  - The numeric value at that time is obtained from each of the input time series data values, applying appropriate interpolation rules and treatment of null values.
  - The numeric values thus obtained are combined using the normal rules for the operator, including the treatment of null values. For logical operators, zero input is treated as false and non-zero non-null input is treated as true. For logical and comparison operators, false output is converted to zero and true output is converted to one. If either input is null, the result is null, with the following exceptions:
    - IS NULL and IS NOT NULL work as one would expect
    - = is true if both arguments are NULL, false if only one argument is NULL
    - <> is true if only one argument is NULL, false if both arguments are NULL
    - OR is true if the non-NULL argument is non-zero, otherwise false
  - The resulting numeric value is placed in the output time series data value as a data point at that time (except that consecutive null data points are dropped if not at the end of the time domain).
- If all the input time series data values use step/extend interpolation, the result also uses step/extend. Otherwise it uses linear.

Standard functions that can be used with time series data values

A wide variety of the standard SQL functions can be used with time series data values, as described below.

If all the input time series data values for a function use extend/step interpolation, the result also uses extend/step. Otherwise it uses linear.

Simple functions taking numeric arguments

All of the standard SQL functions that take one or more numeric arguments and that yield a numeric result can also be used with time series data arguments and will yield a time series data value as a result. The rules described above for operators apply for converting arguments to time series data values and for computing the times and values of individual data points. These functions are:

- INT, FLOOR, CEIL, ABS
- All trigonometric and logarithmic functions

Functions that work with lists

The list-related functions LOOKUP, INDEX, RINDEX and MEMBER can take a time series data value as their first argument and a list as their last argument, returning a time series data value. Normally the list will be a number list. If used with date or string lists, then for each individual data point value, LOOKUP will convert the date/string to a number, MEMBER and RINDEX will always return 0 (not present) and INDEX will always return NULL.

Functions that work with dates

Functions that work with dates and have a numeric or Boolean result can also be used with time series data values. These functions are:

- YEARPART
- MONTHPART
- DAYPART
- DAYOFWEEK
- DAYOFYEAR
- DAYSINYEAR
- DATEPART
- TIMEPART
- INYEAR
- INMONTH
- INYEARS
- INMONTHS

The values in the time series are treated as being the numeric representation of dates (e.g. as set by the TSDATETIME function) and the various functions operate on these dates as documented. Note that TIMEPART is always truncated to a whole number of minutes.

Conditional functions

The standard conditional SQL functions IIF and NVL also work for time series data values. Unlike the other functions, they do not generate an output data point for every possible input time. Instead:

- IIF uses times from the second argument only during periods when the first argument is true (non-zero and non-null) and uses times from the third argument only during periods when the first argument is false (zero). This avoids introducing unnecessary times from data points in the unused time series. During periods when the first argument is null, the result is null.
- NVL uses times from the second argument only during periods when the first argument is null. This avoids introducing unnecessary times from data points in the unused time series.

Other standard functions

If standard SQL functions other than those listed above are used with time series data arguments, the result is a null value.

Specialised functions for use only with time series data values in InfoWorks ICM

The following functions can be used only to create or manipulate time series data values. They cannot be used in "normal" SQL expressions and queries.

- TSDATA
- TSPOINTS
- TSRESAMPLE
- TSAGGREGATE
- TSAPPLYPROFILE
- TSBEFORE
- TSDATETIME
- TSLAG
- TIDE

TSDATA

TSDATA (source, units)

Creates a time series from a data source that is either a single number, or a stream of observed or previously-calculated data.

The source will usually be the name of one of the TVD connector's input fields; that is, one of input_a, input_b, or input_c.

The string value in this field is then extracted and passed to the function as described below. Alternatively, the source can be specified directly as either a number or a string.

The string passed to the function must be one of:

- The id of a TVD connector
- The id of a TSD stream, prefixed by the character #
- A number in string or numeric format

If a TVD connector or TSD stream id is specified then it must be one of the ids that appear in the input fields of the TVD connector (i.e. it is not possible to use arbitrary ids from the network or TSD).

If there is ambiguity as to the meaning of the string (e.g. there is a TVD connector named "3") then the first-matching interpretation from the above list will be used.

The units must be specified as a string such as "mm" or "ft" or "" and must be specified only if the source is a TSD stream ID, in which case the specified units need to be compatible with the units (quantity) of the source.

The constructor function returns a time series data value constructed as follows for each type of source.

Examples of using TSDATA in an expression is included in the TVD Connector-SQL Expression topic.

TVD connector id

The returned value is obtained by evaluating the expression defined for the specified TVD connector. This might in turn involve the evaluation of expressions defined for other TVD connectors. Checks are made to prevent an infinite loop of expression evaluation.

TSD stream id

The returned value is obtained from the named TSD stream for the specified time domain, noting that:

- Numeric values within the time series are converted to the specified units
- Data points with null values are inserted as needed to directly represent missing periods at the start, end or between observed values (where the gap is greater than the data interval of the TSD stream)
- The interpolation rule for the time series data value is set to be the same as for the TSD stream
- If necessary, numeric values are calculated for the start and end point of the domain using the appropriate interpolation rule

Number (as numeric value or string)

The returned value has interpolation="extend" and has two points, one at the start time with numeric value equal to the specified number and one at the end time with a null value.

TSPOINTS

TSPOINTS(interval_unit, interval_multiplier, date_origin, time_of_day_origin, daylight_saving)

Constructs a time series containing data points that have a specified, repeating time interval. All data points in this series are given the same value (1.0). Intervals are not necessarily of equal duration (e.g. days adjusted for daylight saving; months; years). The created time series uses step/extend interpolation.

The main uses of such a series are to define the times for resampling of other series (using TSRESAMPLE) or to generate profiles that repeat for each interval (using TSAPPLYPROFILE).

TSPOINTS is primarily intended for use in TVD Connector expressions, where the time domain is defined by the simulation period.

The interval unit must be one of the following:

- "s" or "second"
- "m" or "minute"
- "h" or "hour"
- "d" or "day"
- "w" or "week"
- "mon" or "month"
- "y" or "year"

The chosen interval unit is multiplied by the interval multiplier to give the interval between data points. The multiplier may be fractional if the interval unit is a day or smaller and must be greater than or equal to 0.001.

If daylight saving is true, adjustments for daylight saving are made when crossing a transition between standard time and daylight saving time in the time zone of the domain. Daylight saving adjustment is only carried out if the interval unit is an hour or greater and the interval multiplier is a whole number. This can be used to create daily time stamps that are at the same local clock time every day (e.g. 09:00). Because InfoWorks ICM runs are always in local standard time, the visible time stamp is then one hour earlier during daylight saving (e.g. 08:00).

The procedure for aligning the time stamps of the data points is as follows:

- If the interval unit is hours or smaller, or date_origin is null or an empty string, the date origin is taken to be the day before the day that contains the start of the time domain. Otherwise, the string or date provided is used and must be before the date of the start of the time domain. Only the date part of the date_origin is used.
- If the time origin is null or an empty string, it is taken to be midnight ("00:00:00"). Otherwise, it must be a time-of-day string (e.g. "09:00", "00:00:00.1"). The length of time after midnight must be less than the interval.
- The date origin and time origin are combined and used as the time-stamp of the "origin" data point of the series. The time stamps of all other data points are calculated as successors to this origin using the specified interval (this is why the origin needs to be before the start of the time domain).
- The actual first data point in the created series will be the data point at or immediately before the start of the time domain, while the last point will be at or immediately after the end of the time domain.

The parameters of this function are somewhat complicated due to the wide range of possible time scales and possible variations in the length of intervals. The following examples of different intervals should help to make them clearer:

- 100 milliseconds

  TSPOINTS("s", 0.1, "", "", 0)
- 30 seconds

  TSPOINTS("s", 30, "", "", 0)

- 15 minutes

  TSPOINTS("m", 15, "", "", 0)
- 3 hours

  TSPOINTS("h", 3, "", "", 0)
- 12 hours, at 09:00 and 21:00 local clock time

  TSPOINTS("h", 12, "", "09:00", 1)
- Daily at midnight local standard time

  TSPOINTS("d", 1, "", "", 0)
- Weekly, every Wednesday at 09:00 local clock time (noting that 5 Jan 2000 was a Wednesday)

  TSPOINTS("w", 1, "2000-01-05", "09:00", 1)
- Quarterly, at midnight local clock time on the first day of the month

  TSPOINTS("mon", 3, "2000-01-01", "", 1)
- Annually, at midnight local clock time on 1 July each year

  TSPOINTS("y", 1, "2000-07-01", "", 1)

Note that it is not possible to create diary-style or specialised intervals (e.g "3rd Wednesday of every month"; "accounting month"; "financial quarter").

TSRESAMPLE

TSRESAMPLE(value_series, time_stamp_series, statistic, window_option, left_secs, right_secs, statistic_factor, output_interpolation)

Resamples the values in value_series using the time stamps of the non-null, non-zero data points in time_stamp_series, calculating the value for a window around each time stamp using the specified statistic and window_option.

Null or zero-valued data points in the time_stamp_series are used to generate output data points at the same time stamps with these points having null values. This is useful for calculating statistics of event periods where each event is represented by a non-null, non-zero data point at the start of the event and a null or zero data point at the end of the event.

The resulting time series uses the specified output_interpolation.

The window options are:

- 1 – window is the interval between this time stamp and the next time stamp
- 0 – window is from left_secs seconds before this time stamp to right_secs seconds after it
- -1 – window is the interval between the preceding time stamp and this time stamp

The statistics that may be calculated for values in the window are:

- "MEAN" - time-mean value

- "TMEAN" - triangular-weighted time-mean (used mainly for smoothing)
- "INTEGRAL" - time-integral (with time measured in seconds)
- "SDEV" - standard deviation
- "FIRST" - first non-null value
- "LAST" - last non-null value
- "DUR" - duration(length) of the time window in seconds
- "DVDT" - time derivative over the window
- "MAX" - maximum value
- "MIN" - minimum value
- "MAXTIME" - time (in seconds after start of window) at which maximum occurs
- "MINTIME" - time (in seconds after start of window) at which minimum occurs
- "COUNT" - the count of data points (including those with null values)
- "COUNTNONNULL" - the count of data points (excluding those with null values)
- "TOTAL" - the sum of the values from each data point
- "COVERAGE" - the percentage of time during which the value is non-null
- "PERCENTILE" - value of the specified percentile of individual values in the window
- "BWDPERCENTILE" - percentile calculated as prescribed by EU Bathing Water Directive

Note that:

- MEAN, TMEAN, INTEGRAL and SDEV are based on time-integration of the values in the window according to the interpolation rule for the value series. If the interpolation rule is linear or step/extend then there need not be any data points from the value series within the window. For delta interpolation, all non-null points are given a unit weighting (thus INTEGRAL is equivalent to TOTAL and MEAN is a simple average)
- PERCENTILE and BWDPERCENTILE statistics are based on the values at non-null points within the time window (i.e. no time-weighting)
- If there are periods of null value within the window, most statistics are calculated over the non-null periods, unless the entire window is null. The exceptions are:
    - DUR is calculated from window start to window end, regardless of whether either end is null.
    - DVDT is calculated from window start to window end. If either end is null, the result is null.
    - COUNT counts both null and non-null data points
    - COVERAGE is the non-null percentage of the entire window
- The window is treated as being half-open on the right – that is, a time stamp that defines the right end of the window is not treated as falling within the window. This means that:
    - The COUNT of a window which is the interval between successive data points is 1, not 2. Likewise, the TOTAL, PERCENTILE and BWDPERCENTILE and any statistics using delta

interpolation do not include the value of the second data point. This avoid double-counting of data points and their values.

  - For a extend/step series, the LAST value in a window which is the interval between two time stamps is the value from the first time stamp. This reflects the fact that a step/extend series is a discontinuous function.

- For points near the start or end of the time domain, it is possible for the time window to extend to times outside the domain. The value series might or might not have data at these times (depending on how it overlaps the time domain) and times with no data are treated as null. Thus statistics at or near end points can be different from what they would be at the same times with a wider time domain. The consequences of this for simulation are discussed in the Resampling buffer and chained (continuous) simulation section.

For most of the above statistics, the statistic parameter is simply a multiplier that is applied after calculating the statistic. This is useful for applying a scaling factor and/or converting time from seconds to some other unit (for INTEGRAL, DUR, DVDT, MINTIME or MAXTIME). A value of 1.0 should be specified if no scaling is required. For PERCENTILE and BWDPERCENTILE, the statistic parameter is the percentile to be calculated (e.g. 95, 99).

The output interpolation must be one of

- "linear"
- "extend"/"step"

Resampling buffer and chained (continuous) simulation

As noted above, resampling time windows can extend outside the time domain, which by default is from the start time to the end time of a simulation, and this can affect the resampled statistics. This has implications for chained simulations (as carried out by InfoWorks ICM) in which successive simulations start from states saved by previous simulations. Note that this can occur even when using a time window that is entirely to the 'right' (future) of each time point to be resampled, as the first such time point may lie before the start of the time domain (e.g. if created using TSPOINTS).

The principle of chaining is that the inputs and thus the end results are the same regardless of whether the calculations are made as chained simulations or as a single simulation covering the same overall period (this use of chained simulations is also known as 'continuous' simulation). If resampled input data near the start time of a chained simulation is different from what it would be in the middle of the equivalent overall simulation, there will be an inconsistency between chained and single-simulation results.

To avoid this, the user can specify a "resampling buffer", as a number of minutes, as a property of a TVD connector . This is only necessary for TVD connectors that use TSRESAMPLE in an expression. At evaluation time, the largest of the resampling buffers from all TVD connectors in the network is used to extend the time domain backwards far enough for the resampled statistics at the start time of the simulation to be the same as they would be in a longer simulation .

The size of the resampling buffer must be decided by the user, based on the time windows in all uses of the TSRESAMPLE function within the TVD connector itself and all its upstream inputs (if any). For example, TSRESAMPLE might be used exactly once to provide smoothing as a moving average with a time window of half an hour either side of a data point, as follows:

- **TSRESAMPLE($values, $points, "MEAN", 0, -1800, 1800, 1, "linear")**

For this example, the resampling buffer should be at least 30.

If the time window is asymmetric (i.e. its extent to the left/earlier is different from its extent to the right/later) then its extent to the left should be used (i.e. 30 in the above example). If TSREAMPLE is applied in series (i.e. resampling a series which is then itself resampled) the time windows should be added together. If TSRESAMPLE is applied in parallel (i.e. resampling two series independently, then combining the resampled series) the larger of the windows should be used. If the time window is ambiguous (i.e. it is the previous or next interval of an input time series that does not have a well-defined interval) then a window large enough to accommodate the largest likely interval should be assumed. If the first time point to be sampled could lie before the start of the domain (e.g. if created by TSPOINTS with an hourly interval, but the time domain starts on a quarter hour) then the largest possible gap before the start of the domain should be added (45 minutes in this example).

More generally, if use of resampling is complex, it does not hurt to specify a resampling buffer that is larger than absolutely necessary.

For simplicity, the resampling buffer is used to extend both the start and end of the time domain, although the latter is rarely an issue as the state (used to chain simulations together) is typically saved well before the end of the simulation.

Since the extension of the time domain applies to all input data, regardless of whether it is actually resampled, it is advisable not to use excessively large resampling buffers (minutes or a small number of hours would be typical; a day or longer is unlikely to be appropriate or useful).

An example of using the TSRESAMPLE in an expression is included in the TVD Connector-SQL Expression topic.

TSAGGREGATE

TSAGGREGATE(value_series, start_time, end_time, statistic, statistic_factor)

Calculates the specified statistic for values in value_series using the specified time window.

This function returns either null or a single numeric value – not a time series data value. It can be used within a TVD Connector expression , but is primarily intended for use in an SQL query.

The available statistics and the meaning of the statistic factor are exactly as described for the TSRESAMPLE function, with the following exception:

- The MAXTIME and MINTIME statistics return numbers that can be interpreted as dates (i.e. days since 00:00 30-Dec-1899) in the local clock time (i.e. including daylight saving) of the machine.

TSAPPLYPROFILE

TSAPPLYPROFILE(input_series, normalize, divide_by_duration, multiplier, output_interpolation,

number_list)

This function takes a profile, defined as a number list of N values, and applies it to every data point in the input series that has a non-zero, non-null value, as follows:

- If the normalize option is true, the profile in the list is first normalized (i.e. time-mean value set to 1.0).
- The time interval between the current data point and the next data point is divided into either N (if output_interpolation is step/extend) or N-1 (if output_interpolation is linear) equal intervals
- An initial value at each data point in the sub-divided interval (starting at the time of the original point) is computed by multiplying the value at the original point by the value for the corresponding entry in the list. If output_interpolation is linear and the subsequent value in the original series is zero or null, the N"th value in the list is applied at this time.
- If divide_by_duration is true, these initial values are then divided by the duration of the interval (in seconds). This can be used to convert an original value that is a total over the original interval (e.g. mm of rain) into an intensity (e.g. mm/s).
- Finally, each value is multiplied by the specified multiplier. This can be used to convert from, say, mm/s to mm/hr.

For each input data point that has a null or zero value, an output data point with a null value is generated. If the output uses step/extend interpolation, the time stamp for this output point will be the same as the input. If the output uses linear interpolation, there will already be an output point at this time stamp (to allow interpolation up to the end of the interval) and so the null-valued point is output with a time stamp half way between this and the next input time stamp – this prevents interpolation between periods that represent distinct events.

The resulting time series uses the specified output interpolation, which must be one of:

- "linear"
- "extend" or "step"

An example of using the TSAPPLYPROFILE in an expression is included in the TVD Connector-SQL Expression topic.

TSBEFORE

TSBEFORE(date_time)

Constructs a series that is true (1.0) before the specified date-time and false (0.0) thereafter, with interpolation="extend". The resulting time series data value will contain either 3 data points (if the specified date-time is inside the time domain) or 2 data points (if the specified date-time is outside or on a boundary of the time-domain).

An example of using the TSBEFORE in an expression is included in the TVD Connector-SQL Expression topic.

TSDATETIME

TSDATETIME(data_series, daylight_saving, output_interpolation)

Constructs a series with time stamps that match the points in the input data series and values that are the numeric equivalents of these time stamps. If daylight saving is non-zero, daylight saving adjustment is applied when converting the time stamps to numeric values.

The numeric equivalent of a time stamp is the number of days (including fractional days) since 00:00 30-Dec-1899. This is not usually of direct interest but can be used for:

- Calculating the time difference (in days) from a specific date-time, such as the run origin
- Extracting particular components of the date-time using SQL functions that work on date-times, such as YEARPART, MONTHPART, DAYPART, TIMEPART, DAYOFWEEK, DAYOFYEAR, DAYSINYEAR

Daylight saving adjustment should not be applied if the values are to be used for calculating time differences. It is intended for use in converting these values to time of day or day of week in local clock time.

The resulting time series uses the specified output interpolation, which must be one of

- "linear"
- "extend" or "step"

An example of using the TSDATETIME in an expression is included in the TVD Connector-SQL Expression topic.

TSLAG

TSLAG (input_series, lag_seconds)

Constructs a series containing the same data points as the input series, but with time stamps shifted forward (lagged) by the specified number of seconds. A negative lag may be specified to shift the time stamps backwards. When used in a TVD connector expression, the resampling buffer for the TVD connector (in minutes) should be set to be at least as large as the lag time (in seconds) so that data from before the start of the simulation can be shifted into the simulation period (otherwise, the first lag_seconds of the simulation period will contain null data).

An example of using TSLAG in an expression is included in the TVD Connector-SQL Expression topic.

TIDE

TIDE(constituent_name, amplitude, phase_lag, timestep)

Constructs a time series for a specified tidal constituent, amplitude and phase lag, using linear interpolation.

- constituent_name – a string which identifies a tidal constituent. 28 constituents, derived from the UK Admiralty Tide Tables (time zone referenced by Greenwich Mean Time), are supported and are described in the Tidal constituents table.

- amplitude – the mean amplitude of the constituent (taken over a 19-year tidal epoch) in units such as metres.
- phase_lag – the phase lag for each tidal constituent, in degrees. This is the phase lag of the constituent relative to the phase of the equilibrium tide at Greenwich, where the phase at Greenwich is calculated using local standard time (i.e. not converted to GMT).
- timestep – the time interval between output data points, in minutes. The timestep must not be less than 0.1 minutes.

The first output data point is at the start of the time domain and the last is at or immediately after the end of the time domain.

The choice of whether to reference phase lag to local standard time or GMT is a matter of convention. When using the TIDE function for time zones other than GMT, it is important to be clear that TIDE uses the convention that phase lags are referenced to local standard time. If a phase lag (G) referenced to GMT is provided, it can be converted to a phase lag (g) referenced to local standard time in time zone (UTC + N hours) as follows:

- **g = G - aN**

Where **a** is the angular speed of the constituent in degrees/hour.

Notes on using NULL in expressions

The behaviour of NULL is slightly complex:

1. Essentially NULL is an 'I have no value' value.
2. The general rule for SQL is that any operation involving anything and NULL gives the answer NULL e.g. 3 + NULL is NULL. Important exceptions to this rule are the operators OR, IS NULL and IS NOT NULL and the function NVL (OR treats NULL as false, while NVL is used to detect and replace NULL values).

Parent topic: Technical Notes