+++ title = "About expressions" weight = 10 aliases = ["/docs/sources/panels/query-a-data-source/use-expressions-to-manipulate-data/about-expressions/"] +++

About expressions

> ***Note:*** *This documentation is for a beta feature.*

Server-side expressions allow you to manipulate data returned from queries with math and other operations. Expressions create new data and do not manipulate the data returned by data sources, aside from some minor data restructuring to make the data acceptable input for expressions.

Using expressions

Expressions are primarily used by the new [Grafana 8 alerts]({{< relref "../../../alerting/unified-alerting/_index.md" >}}). The processing is done server-side, so expressions can operate without a browser session. However, expressions can also be used with backend data sources and visualization.

> ***Note:*** *Expressions do not work with legacy dashboard alerts.*

Expressions are meant to augment data sources by enabling queries from different data sources to be combined or by providing operations unavailable in a data source.

> ***Note:*** *When possible, you should do data processing inside the data source. Copying data from storage to the Grafana server for processing is inefficient, so expressions are targeted at lightweight data processing.*

Expressions work with data source queries that return time series or number data. They also operate on [multiple-dimensional data]({{< relref "../../../basics/timeseries-dimensions.md" >}}). For example, a query that returns multiple series, where each series is identified by labels or tags.

An individual expression takes one or more queries or other expressions as input and adds data to the result. Each individual expression or query is represented by a variable that is a named identifier known as its RefID (e.g., the default letter `A` or `B`).

To reference the output of an individual expression or a data source query in another expression, this identifier is used as a variable.

Types of expressions

Expressions work with two types of data.

- A collection of time series.
- A collection of numbers, where each number is an item.

Each collection is returned from a single data source query or expression and represented by the RefID. Each collection is a set, where each item in the set is uniquely identified by its dimensions which are stored as [labels]({{< relref "../../../basics/timeseries-dimensions.md#labels" >}}) or key-value pairs.

Data source queries

Server-side expressions only support data source queries for backend data sources. The data is generally assumed to be labeled time series data. In the future we intended to add an assertion of the query return type (number or time series) data so expressions can handle errors better.

Data source queries, when used with expressions, are executed by the expression engine. When it does this, it restructures data to be either one time series or one number per data frame. So for example if using a data source that returns multiple series on one frame in the table view, you might notice it looks different when executed with expressions.

Currently, the only non-time series format (number) is supported when using data frames are you have a table response that returns a data frame with no time, string columns, and one number column:

Loc	Host	Avg_CPU
MIA	A	1
NYC	B	2

The example above will produce a number that works with expressions. The string columns become labels and the number column the corresponding value. For example `{"Loc": "MIA", "Host": "A"}` with a value of 1.

Operations

You can use the following operations in expressions: math, reduce, and resample.

Math

Math is for free-form math formulas on time series or number data. Math operations take numbers and time series as input and changes them to different numbers and time series.

Data from other queries or expressions are referenced with the RefID prefixed with a dollar sign, for example `$A`. If the variable has spaces in the name, then you can use a brace syntax like `${my variable}`.

Numeric constants may be in decimal (`2.24`), octal (with a leading zero like `072`), or hex (with a leading 0x like `0x2A`). Exponentials and signs are also supported (e.g., `-0.8e-2`).

Operators

The arithmetic (`+` , binary and unary `-` , `*` , `/` , `%` , exponent `**`), relational (`<` , `>` , `==` , `!=` , `>=` , `<=`), and logical (`&&` , `||` , and unary `!`) operators are supported.

How the operation behaves with data depends on if it is a number or time series data.

With binary operations, such as `$A + $B` or `$A || $B` , the operator is applied in the following ways depending on the type of data:

- If both `$A` and `$B` are a number, then the operation is performed between the two numbers.
- If one variable is a number, and the other variable is a time series, then the operation between the value of each point in the time series and the number is performed.
- If both `$A` and `$B` are time series data, then the operation between each value in the two series is performed for each time stamp that exists in both `$A` and `$B` . The Resample operation can be used to line up time stamps. (**Note:** in the future, we plan to add options to the Math operation for different behaviors).

Summary:

- Number OP number = number
- Number OP series = series
- Series OP series = series

Because expressions work with multiple series or numbers represented by a single variable, binary operations also perform a union (join) between the two variables. This is done based on the identifying labels associated with each individual series or number.

So if you have numbers with labels like `{host=web01}` in `$A` and another number in `$B` with the same labels then the operation is performed between those two items within each variable, and the result will share the same labels. The rules for the behavior of this union are as follows:

- An item with no labels will join to anything.
- If both `$A` and `$B` each contain only one item (one series, or one number), they will join.
- If labels are exact math they will join.
- If labels are a subset of the other, for example and item in `$A` is labeled `{host=A,dc=MIA}` and and item in `$B` is labeled `{host=A}` they will join.
- Currently, if within a variable such as `$A` there are different tag *keys* for each item, the join behavior is undefined.

The relational and logical operators return 0 for false 1 for true.

Math Functions

While most functions exist in the own expression operations, the math operation does have some functions that similar to math operators or symbols. When functions can take either numbers or series, than the same type as the argument will be returned. When it is a series, the operation of performed for the value of each point in the series.

abs

abs returns the absolute value of its argument which can be a number or a series. For example `abs(-1)` or `abs($A)`.

is_inf

is_inf takes a number or a series and returns `1` for `Inf` values (negative or positive) and `0` for other values. For example `is_inf($A)`.

> ***Note:*** *If you need to specifically check for negative infinity for example, you can do a comparison like* `$A == infn()`.

is_nan

is_nan takes a number or a series and returns `1` for `NaN` values and `0` for other values. For example `is_nan($A)`. This function exists because `NaN` is not equal to `NaN`.

is_null

is_nan takes a number or a series and returns `1` for `null` values and `0` for other values. For example `is_null($A)`.

is_number

is_number takes a number or a series and returns `1` for all real number values and `0` for other values (which are `null`, `Inf+`, `Inf-`, and `NaN`). For example `is_number($A)`.

log

Log returns the natural logarithm of of its argument which can be a number or a series. If the value is less than 0, NaN is returned. For example `log(-1)` or `log($A)`.

inf, infn, nan, and null

The inf, infn, nan, and null functions all return a single value of the name. They primarily exist for testing. Example: `null()`.

round

Round returns a rounded integer value. For example, `round(3.123)` or `round($A)`. (This function should probably take an argument so it can add precision to the rounded value).

ceil

Ceil rounds the number up to the nearest integer value. For example, `ceil(3.123)` returns 4.

floor

Floor rounds the number down to the nearest integer value. For example, `floor(3.123)` returns 3.

Reduce

Reduce takes one or more time series returned from a query or an expression and turns each series into a single number. The labels of the time series are kept as labels on each outputted reduced number.

Fields:

- **Function -** The reduction function to use
- **Input -** The variable (refID (such as `A`)) to resample
- **Mode -** Allows control behavior of reduction function when a series contains non-numerical values (null, NaN, +-Inf)

Reduction Functions

Count

Count returns the number of points in each series.

Mean

Mean returns the total of all values in each series divided by the number of points in that series. In `strict` mode if any values in the series are null or nan, or if the series is empty, NaN is returned.

Min and Max

Min and Max return the smallest or largest value in the series respectively. In `strict` mode if any values in the series are null or nan, or if the series is empty, NaN is returned.

Sum

Sum returns the total of all values in the series. If series is of zero length, the sum will be 0. In `strict` mode if there are any NaN or Null values in the series, NaN is returned.

Last

Last returns the last number in the series. If the series has no values then returns NaN.

Reduction Modes

Strict

In Strict mode the input series is processed as is. If any values in the series are non-numeric (null, NaN or +-Inf), NaN is returned.

Drop Non-Numeric

In this mode all non-numeric values (null, NaN or +-Inf) in the input series are filtered out before executing the reduction function.

Replace Non-Numeric

In this mode all non-numeric values are replaced by a pre-defined value.

Resample

Resample changes the time stamps in each time series to have a consistent time interval. The main use case is so you can resample time series that do not share the same timestamps so math can be performed between them. This can be done by resample each of the two series, and then in a Math operation referencing the resampled variables.

Fields:

- **Input -** The variable of time series data (refID (such as `A`)) to resample
- **Resample to -** The duration of time to resample to, for example `10s`. Units may be `s` seconds, `m` for minutes, `h` for hours, `d` for days, `w` for weeks, and `y` of years.
- **Downsample -** The reduction function to use when there are more than one data point per window sample. See the reduction operation for behavior details.
- **Upsample -** The method to use to fill a window sample that has no data points.
  - **pad** fills with the last know value
  - **backfill** with next known value
  - **fillna** to fill empty sample windows with NaNs