+++ title = "Reference: Transformation functions" aliases = ["/docs/grafana/latest/panels/transformations/types-options/", "docs/sources/panels/reference-transformation-functions/"] weight = 1000 +++

Reference: Transformation functions

You can perform the following transformations on your data.

Add field from calculation

Use this transformation to add a new field calculated from two other fields. Each transformation allows you to add one new field.

- **Mode -** Select a mode:
  - **Reduce row -** Apply selected calculation on each row of selected fields independently.
  - **Binary option -** Apply basic math operation(sum, multiply, etc) on values in a single row from two selected fields.
- **Field name -** Select the names of fields you want to use in the calculation for the new field.
- **Calculation -** If you select **Reduce row** mode, then the **Calculation** field appears. Click in the field to see a list of calculation choices you can use to create the new field. For information about available calculations, refer to [Reference: Calculations]({{< relref "./reference-calculation-types.md" >}}).
- **Operation -** If you select **Binary option** mode, then the **Operation** fields appear. These fields allow you to do basic math operations on values in a single row from two selected fields. You can also use numerical values for binary operations.
- **Alias -** (Optional) Enter the name of your new field. If you leave this blank, then the field will be named to match the calculation.
- **Replace all fields -** (Optional) Select this option if you want to hide all other fields and display only your calculated field in the visualization.

In the example below, I added two fields together and named them Sum.

{{< figure src="/static/img/docs/transformations/add-field-from-calc-stat-example-7-0.png" class="docs-image--no-shadow" max-width= "1100px" >}}

Concatenate fields

This transformation combines all fields from all frames into one result. Consider:

Query A:

Temp	Uptime
15.4	1230233

Query B:

AQI	Errors
3.2	5

After you concatenate the fields, the data frame would be:

Temp	Uptime	AQI	Errors

15.4	1230233	3.2	5

Config from query results

This transformation allow you to select one query and from it extract standard options like **Min**, **Max**, **Unit** and **Thresholds** and apply it to other query results. This enables dynamic query driven visualization configuration.

If you want to extract a unique config for every row in the config query result then try the rows to fields transformation.

Options

- **Config query**: Select the query that returns the data you want to use as configuration.
- **Apply to**: Select what fields or series to apply the configuration to.
- **Apply to options**: Usually a field type or field name regex depending on what option you selected in **Apply to**.

Convert field type

This transformation changes the field type of the specified field.

- **Field -** Select from available fields
- **as -** Select the FieldType to convert to
  - **Numeric -** attempts to make the values numbers
  - **String -** will make the values strings
  - **Time -** attempts to parse the values as time
    - Will show an option to specify a DateFormat as input by a string like yyyy-mm-dd or DD MM YYYY hh:mm:ss
  - **Boolean -** will make the values booleans

For example the following query could be modified by selecting the time field, as Time, and Date Format as YYYY.

Time	Mark	Value
2017-07-01	above	25
2018-08-02	below	22
2019-09-02	below	29
2020-10-04	above	22

The result:

Time	Mark	Value
2017-01-01 00:00:00	above	25
2018-01-01 00:00:00	below	22
2019-01-01 00:00:00	below	29
2020-01-01 00:00:00	above	22

Filter data by name

Use this transformation to remove portions of the query results.

Grafana displays the **Identifier** field, followed by the fields returned by your query.

You can apply filters in one of two ways:

- Enter a regex expression.
- Click a field to toggle filtering on that field. Filtered fields are displayed with dark gray text, unfiltered fields have white text.

In the example below, I removed the Min field from the results.

Here is the original query table. (This is streaming data, so numbers change over time and between screenshots.)

{{< figure src="/static/img/docs/transformations/filter-name-table-before-7-0.png" class="docs-image--no-shadow" max-width= "1100px" >}}

Here is the table after I applied the transformation to remove the Min field.

{{< figure src="/static/img/docs/transformations/filter-name-table-after-7-0.png" class="docs-image--no-shadow" max-width= "1100px" >}}

Here is the same query using a Stat visualization.

{{< figure src="/static/img/docs/transformations/filter-name-stat-after-7-0.png" class="docs-image--no-shadow" max-width= "1100px" >}}

Filter data by query

Use this transformation in panels that have multiple queries, if you want to hide one or more of the queries.

Grafana displays the query identification letters in dark gray text. Click a query identifier to toggle filtering. If the query letter is white, then the results are displayed. If the query letter is dark, then the results are hidden.

In the example below, the panel has three queries (A, B, C). I removed the B query from the visualization.

{{< figure src="/static/img/docs/transformations/filter-by-query-stat-example-7-0.png" class="docs-image--no-shadow" max-width= "1100px" >}}

> ***Note:*** *This transformation is not available for Graphite because this data source does not support correlating returned data with queries.*

Filter data by value

This transformation allows you to filter your data directly in Grafana and remove some data points from your query result. You have the option to include or exclude data that match one or more conditions you define. The conditions are applied on a selected field.

This transformation is very useful if your data source does not natively filter by values. You might also use this to narrow values to display if you are using a shared query.

The available conditions for all fields are:

- **Regex:** Match a regex expression
- **Is Null:** Match if the value is null

- **Is Not Null:** Match if the value is not null
- **Equal:** Match if the value is equal to the specified value
- **Different:** match if the value is different than the specified value

The available conditions for number fields are:

- **Greater:** Match if the value is greater than the specified value
- **Lower:** Match if the value is lower than the specified value
- **Greater or equal:** Match if the value is greater or equal
- **Lower or equal:** Match if the value is lower or equal
- **Range:** Match a range between a specified minimum and maximum, min and max included

Consider the following data set:

Time	Temperature	Altitude
2020-07-07 11:34:23	32	101
2020-07-07 11:34:22	28	125
2020-07-07 11:34:21	26	110
2020-07-07 11:34:20	23	98
2020-07-07 10:32:24	31	95
2020-07-07 10:31:22	20	85
2020-07-07 09:30:57	19	101

If you **Include** the data points that have a temperature below 30°C, the configuration will look as follows:

- Filter Type: `Include`
- Condition: Rows where `Temperature` matches `Lower Than` `30`

And you will get the following result, where only the temperatures below 30°C are included:

Time	Temperature	Altitude
2020-07-07 11:34:22	28	125
2020-07-07 11:34:21	26	110
2020-07-07 11:34:20	23	98
2020-07-07 10:31:22	20	85
2020-07-07 09:30:57	19	101

You can add more than one condition to the filter. For example, you might want to include the data only if the altitude is greater than 100. To do so, add that condition to the following configuration:

- Filter type: `Include` rows that `Match All` conditions
- Condition 1: Rows where `Temperature` matches `Lower` than `30`
- Condition 2: Rows where `Altitude` matches `Greater` than `100`

When you have more than one condition, you can choose if you want the action (include / exclude) to be applied on rows that **Match all** conditions or **Match any** of the conditions you added.

In the example above we chose **Match all** because we wanted to include the rows that have a temperature lower than 30 *AND* an altitude higher than 100. If we wanted to include the rows that have a temperature lower than 30 *OR* an altitude higher than 100 instead, then we would select **Match any**. This would include the first row in the original data, which has a temperature of 32°C (does not match the first condition) but an altitude of 101 (which matches the second condition), so it is included.

Conditions that are invalid or incompletely configured are ignored.

Group by

This transformation groups the data by a specified field (column) value and processes calculations on each group. Click to see a list of calculation choices. For information about available calculations, refer to the [List of calculations]({{< relref "./reference-calculation-types.md" >}}).

Here's an example of original data.

Time	Server ID	CPU Temperature	Server Status
2020-07-07 11:34:20	server 1	80	Shutdown
2020-07-07 11:34:20	server 3	62	OK
2020-07-07 10:32:20	server 2	90	Overload
2020-07-07 10:31:22	server 3	55	OK
2020-07-07 09:30:57	server 3	62	Rebooting
2020-07-07 09:30:05	server 2	88	OK
2020-07-07 09:28:06	server 1	80	OK
2020-07-07 09:25:05	server 2	88	OK
2020-07-07 09:23:07	server 1	86	OK

This transformation goes in two steps. First you specify one or multiple fields to group the data by. This will group all the same values of those fields together, as if you sorted them. For instance if we group by the Server ID field, then it would group the data this way:

Time	Server ID	CPU Temperature	Server Status
2020-07-07 11:34:20	**server 1**	80	Shutdown
2020-07-07 09:28:06	**server 1**	80	OK
2020-07-07 09:23:07	**server 1**	86	OK
2020-07-07 10:32:20	server 2	90	Overload
2020-07-07 09:30:05	server 2	88	OK
2020-07-07 09:25:05	server 2	88	OK
2020-07-07 11:34:20	***server 3***	62	OK
2020-07-07 10:31:22	***server 3***	55	OK

2020-07-07 09:30:57	***server 3***	62	Rebooting

All rows with the same value of Server ID are grouped together.

After choosing which field you want to group your data by, you can add various calculations on the other fields, and apply the calculation to each group of rows. For instance, we could want to calculate the average CPU temperature for each of those servers. So we can add the *mean* calculation applied on the CPU Temperature field to get the following:

Server ID	CPU Temperature (mean)
server 1	82
server 2	88.6
server 3	59.6

And we can add more than one calculation. For instance:

- For field Time, we can calculate the *Last* value, to know when the last data point was received for each server
- For field Server Status, we can calculate the *Last* value to know what is the last state value for each server
- For field Temperature, we can also calculate the *Last* value to know what is the latest monitored temperature for each server

We would then get :

Server ID	CPU Temperature (mean)	CPU Temperature (last)	Time (last)	Server Status (last)
server 1	82	80	2020-07-07 11:34:20	Shutdown
server 2	88.6	90	2020-07-07 10:32:20	Overload
server 3	59.6	62	2020-07-07 11:34:20	OK

This transformation allows you to extract some key information out of your time series and display them in a convenient way.

Join by field (outer join)

Use this transformation to join multiple time series from a result set by field.

This transformation is especially useful if you want to combine queries so that you can calculate results from the fields.

In the example below, I have a template query displaying time series data from multiple servers in a table visualization. I can only view the results of one query at a time.

{{< figure src="/static/img/docs/transformations/join-fields-before-7-0.png" class="docs-image--no-shadow" max-width= "1100px" >}}

I applied a transformation to join the query results using the time field. Now I can run calculations, combine, and organize the results in this new table.

{{< figure src="/static/img/docs/transformations/join-fields-after-7-0.png" class="docs-image--no-shadow" max-width= "1100px" >}}

Labels to fields

This transformation changes time series results that include labels or tags into a table where each label keys and values are included in the table result. The labels can be displayed either as columns or as row values.

Given a query result of two time series:

- Series 1: labels Server=Server A, Datacenter=EU
- Series 2: labels Server=Server B, Datacenter=EU

In "Columns" mode, the result looks like this:

Time	Server	Datacenter	Value
2020-07-07 11:34:20	Server A	EU	1
2020-07-07 11:34:20	Server B	EU	2

In "Rows" mode, the result has a table for each series and show each label value like this:

label	value
Server	Server A
Datacenter	EU

label	value
Server	Server B
Datacenter	EU

Value field name

If you selected Server as the **Value field name**, then you would get one field for every value of the Server label.

Time	Datacenter	Server A	Server B
2020-07-07 11:34:20	EU	1	2

Merging behavior

The labels to fields transformer is internally two separate transformations. The first acts on single series and extracts labels to fields. The second is the merge transformation that joins all the results into a single table. The merge transformation tries to join on all matching fields. This merge step is required and cannot be turned off.

To illustrate this, here is an example where you have two queries that return time series with no overlapping labels.

- Series 1: labels Server=ServerA

- Series 2: labels Datacenter=EU

This will first result in these two tables:

Time	Server	Value
2020-07-07 11:34:20	ServerA	10

Time	Datacenter	Value
2020-07-07 11:34:20	EU	20

After merge:

Time	Server	Value	Datacenter
2020-07-07 11:34:20	ServerA	10	
2020-07-07 11:34:20		20	EU

Merge

Use this transformation to combine the result from multiple queries into one single result. This is helpful when using the table panel visualization. Values that can be merged are combined into the same row. Values are mergeable if the shared fields contain the same data. For information, refer to [Table panel]({{< relref "../visualizations/table/_index.md" >}}).

In the example below, we have two queries returning table data. It is visualized as two separate tables before applying the transformation.

Query A:

Time	Job	Uptime
2020-07-07 11:34:20	node	25260122
2020-07-07 11:24:20	postgre	123001233

Query B:

Time	Job	Errors
2020-07-07 11:34:20	node	15
2020-07-07 11:24:20	postgre	5

Here is the result after applying the Merge transformation.

Time	Job	Errors	Uptime
2020-07-07 11:34:20	node	15	25260122
2020-07-07 11:24:20	postgre	5	123001233

Organize fields

Use this transformation to rename, reorder, or hide fields returned by the query.

> ***Note:*** *This transformation only works in panels with a single query. If your panel has multiple queries, then you must either apply an Outer join transformation or remove the extra queries.*

Grafana displays a list of fields returned by the query. You can:

- Change field order by hovering your cursor over a field. The cursor turns into a hand and then you can drag the field to its new place.
- Hide or show a field by clicking the eye icon next to the field name.
- Rename fields by typing a new name in the **Rename** box.

In the example below, I hid the value field and renamed Max and Min.

{{< figure src="/static/img/docs/transformations/organize-fields-stat-example-7-0.png" class="docs-image--no-shadow" max-width= "1100px" >}}

Reduce

The *Reduce* transformation applies a calculation to each field in the frame and return a single value. Time fields are removed when applying this transformation.

Consider the input:

Query A:

Time	Temp	Uptime
2020-07-07 11:34:20	12.3	256122
2020-07-07 11:24:20	15.4	1230233

Query B:

Time	AQI	Errors
2020-07-07 11:34:20	6.5	15
2020-07-07 11:24:20	3.2	5

The reduce transformer has two modes:

- **Series to rows -** Creates a row for each field and a column for each calculation.
- **Reduce fields -** Keeps the existing frame structure, but collapses each field into a single value.

For example, if you used the **First** and **Last** calculation with a **Series to rows** transformation, then the result would be:

Field	First	Last
Temp	12.3	15.4
Uptime	256122	1230233

AQI	6.5	3.2
Errors	15	5

The **Reduce fields** with the **Last** calculation, results in two frames, each with one row:

Query A:

Temp	Uptime
15.4	1230233

Query B:

AQI	Errors
3.2	5

Rename by regex

Use this transformation to rename parts of the query results using a regular expression and replacement pattern.

You can specify a regular expression, which is only applied to matches, along with a replacement pattern that support back references. For example, let's imagine you're visualizing CPU usage per host and you want to remove the domain name. You could set the regex to `([^\.]+)\..+` and the replacement pattern to `$1`, `web-01.example.com` would become `web-01`.

In the following example, we are stripping the prefix from event types. In the before image, you can see everything is prefixed with `system.`

{{< figure src="/static/img/docs/transformations/rename-by-regex-before-7-3.png" class="docs-image--no-shadow" max-width= "1100px" >}}

With the transformation applied, you can see we are left with just the remainder of the string.

{{< figure src="/static/img/docs/transformations/rename-by-regex-after-7-3.png" class="docs-image--no-shadow" max-width= "1100px" >}}

Rows to fields

The rows to fields transformation converts rows into separate fields. This can be useful as fields can be styled and configured individually. It can also use additional fields as sources for dynamic field configuration or map them to field labels. The additional labels can then be used to define better display names for the resulting fields.

This transformation includes a field table which lists all fields in the data returned by the config query. This table gives you control over what field should be mapped to each config property (the *Use as** option). You can also choose which value to select if there are multiple rows in the returned data.

This transformation requires:

- One field to use as the source of field names.

  By default, the transform uses the first string field as the source. You can override this default setting by selecting **Field name** in the **Use as** column for the field you want to use instead.

- One field to use as the source of values.

  By default, the transform uses the first number field as the source. But you can override this default setting by selecting **Field value** in the **Use as** column for the field you want to use instead.

Useful when visualizing data in:

- Gauge
- Stat
- Pie chart

Map extra fields to labels

If a field does not map to config property Grafana will automatically use it as source for a label on the output field-

Example:

Name	DataCenter	Value
ServerA	US	100
ServerB	EU	200

Output:

ServerA (labels: DataCenter: US)	ServerB (labels: DataCenter: EU)
10	20

The extra labels can now be used in the field display name provide more complete field names.

If you want to extract config from one query and appply it to another you should use the config from query results transformation.

Example

Input:

Name	Value	Max
ServerA	10	100
ServerB	20	200
ServerC	30	300

Output:

ServerA (config: max=100)	ServerB (config: max=200)	ServerC (config: max=300)
10	20	30

As you can see each row in the source data becomes a separate field. Each field now also has a max config option set. Options like **Min**, **Max**, **Unit** and **Thresholds** are all part of field configuration and if set like this will be used by the visualization instead of any options manually configured in the panel editor options pane.

Prepare time series

> ***Note:*** *This transformation is available in Grafana 7.5.10+ and Grafana 8.0.6+.*

Prepare time series transformation is useful when a data source returns time series data in a format that isn't supported by the panel you want to use. For more information about data frame formats, refer to [Data frames]({{< relref "../developers/plugins/data-frames.md" >}}).

This transformation helps you resolve this issue by converting the time series data from either the wide format to the long format or the other way around.

Select the `Multi-frame time series` option to transform the time series data frame from the wide to the long format.

Select the `Wide time series` option to transform the time series data frame from the long to the wide format.

Series to rows

> ***Note:*** *This transformation is available in Grafana 7.1+.*

Use this transformation to combine the result from multiple time series data queries into one single result. This is helpful when using the table panel visualization.

The result from this transformation will contain three columns: Time, Metric, and Value. The Metric column is added so you easily can see from which query the metric originates from. Customize this value by defining Label on the source query.

In the example below, we have two queries returning time series data. It is visualized as two separate tables before applying the transformation.

Query A:

Time	Temperature
2020-07-07 11:34:20	25
2020-07-07 10:31:22	22
2020-07-07 09:30:05	19

Query B:

Time	Humidity
2020-07-07 11:34:20	24
2020-07-07 10:32:20	29
2020-07-07 09:30:57	33

Here is the result after applying the Series to rows transformation.

Time	Metric	Value
2020-07-07 11:34:20	Temperature	25

2020-07-07 11:34:20	Humidity	22
2020-07-07 10:32:20	Humidity	29
2020-07-07 10:31:22	Temperature	22
2020-07-07 09:30:57	Humidity	33
2020-07-07 09:30:05	Temperature	19

Sort by

This transformation will sort each frame by the configured field, When `reverse` is checked, the values will return in the opposite order.