Plotly Overview

User Guide

Contents

1 Introduction . . . 2
1.1 Key Features . . . 2
1.2 Plotly Charts Examples . . . 2
1.3 Plotly Structure . . . 4
1.4 Quick Start . . . 6
1.5 Requirements . . . 6
1.6 Installation . . . 7

2 Value Binding . . . 7
2.1 Quick Start . . . 7
2.2 Property Description . . . 12
2.2.1 Chart . . . 12
2.2.2 Trace . . . 12

3 History Binding . . . 13
3.1 Quick Start . . . 14
3.2 Property Description . . . 20
3.2.1 Chart . . . 20
3.2.2 Trace . . . 21

4 Layout Options . . . 23
4.1 Frequently Used Parameters . . . 24

5 Configuration Options . . . 24
5.1 Time range selector buttons . . . 24
5.2 Frequently Used Parameters . . . 24

6 Trace Options . . . 25

1 Introduction

A picture is worth a thousand words, and, certainly when it comes to data, metrics and KPIs, nothing could be more apt. Without visual tools such as charts and graphs we would soon get lost in the swathers of numbers being presented to us. In Building Automation, visual presentation of data is a first-class citizen. With data collection and use continuing to increase exponentially, the need to visualize this data is becoming more important. Engineers seek to consolidate thousands of database records into beautiful charts and dashboards that humans can quickly and intuitively interpret.

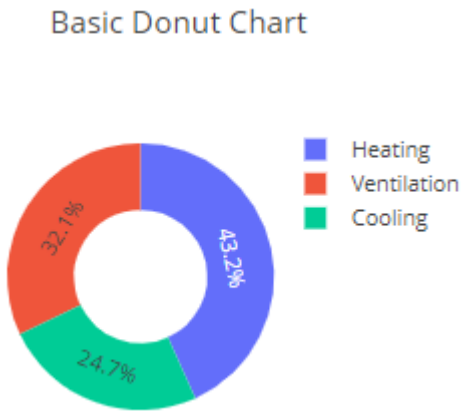
Tridium Niagara offers a great platform to collect the data, and it comes with a few easy to use chart types that are sufficient to do a basic visualision. However, when it comes to large amounts of data and more complex data presentation along with sophisticated dashboards – Plotly library can help.

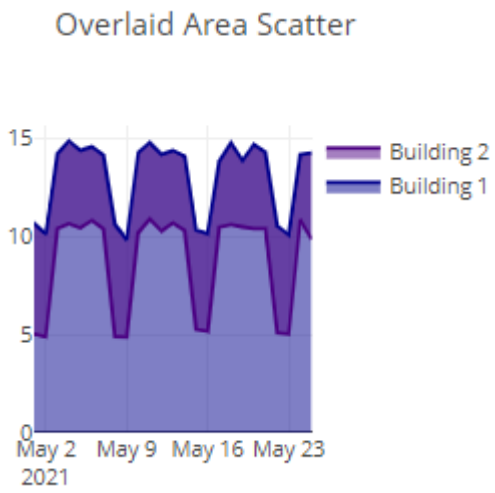
Plotly is an interactive plotting library that supports over 40 unique chart types covering a wide range of statistical, financial, geographic, scientific, and 3-dimensional use-cases. Built on top of d3.js and stack.gl, Plotly enables Niagara users to create beautiful interactive web-based visualizations that can be displayed in standard Niagara px files. Every chart type has hundreds of customisations options and almost every aspect of data visualizations can be modified to suit the needs. Plotly offers advanced charting capabilities which are used by leading data science and statistical companies. It allows engineers to put complex data analytics in the hands of business decision makers and operators.

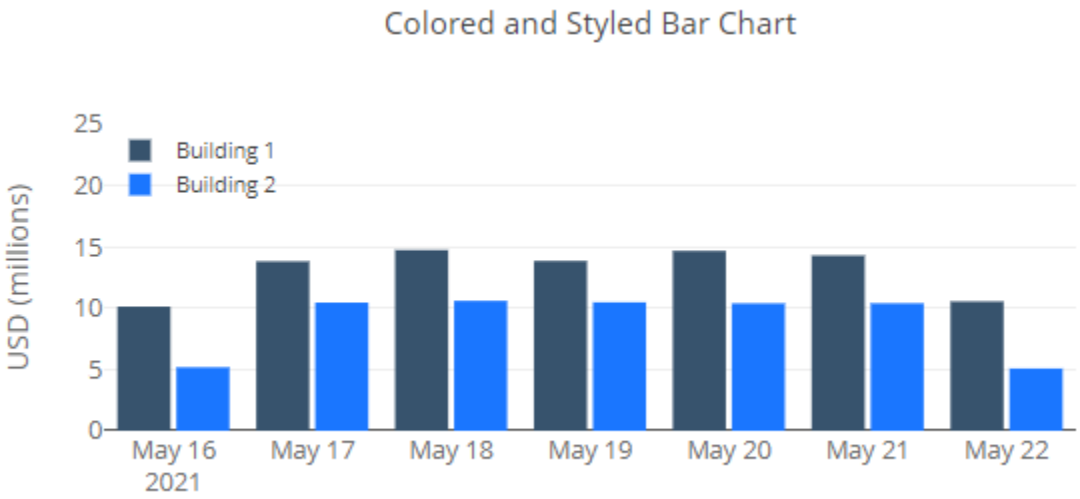
1.1 Key Features

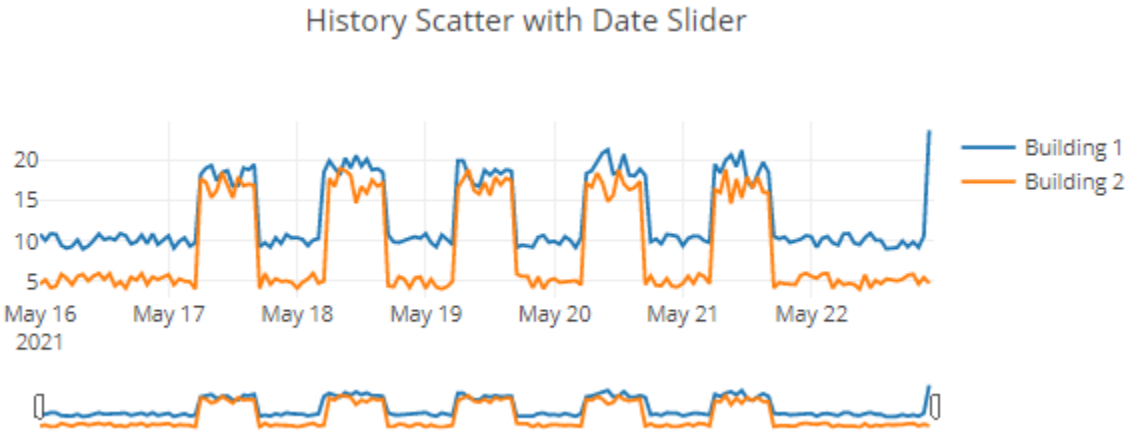
- over 40 unique chart types
- most of the chart properties can be costumized
- both historical and real-time data
- interactive controls
- quick time range selection for historical data

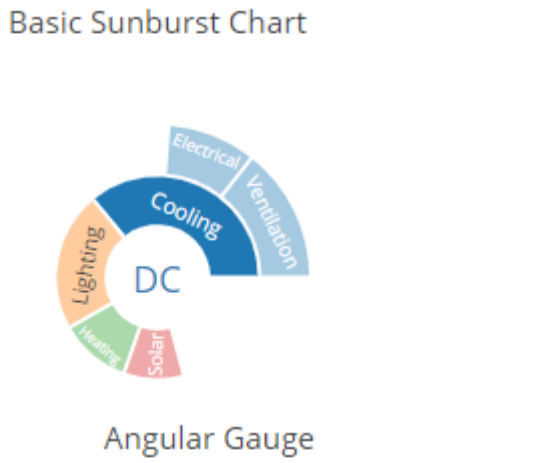
1.2 Plotly Charts Examples



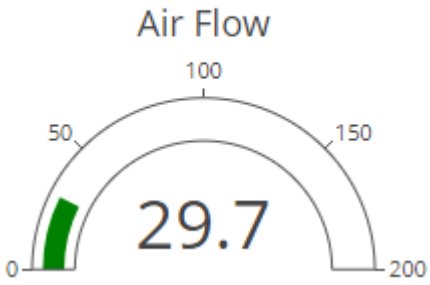

Overlaid Area Scatter
15
10
5
0
May 2
2021
May 9
May 16
May 23
Building 2
Building 1


Colored and Styled Bar Chart
USD (millions)
25
20
15
10
5
0
Building 1
Building 2
May 16
2021
May 17
May 18
May 19
May 20
May 21
May 22

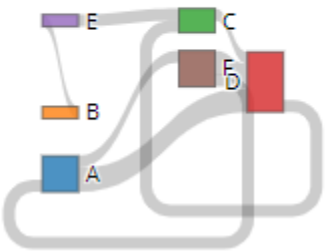

History Scatter with Date Slider
20
15
10
5
May 16
2021
May 17
May 18
May 19
May 20
May 21
May 22
Building 1
Building 2



Angular Gauge



Sankey with manually positioned node

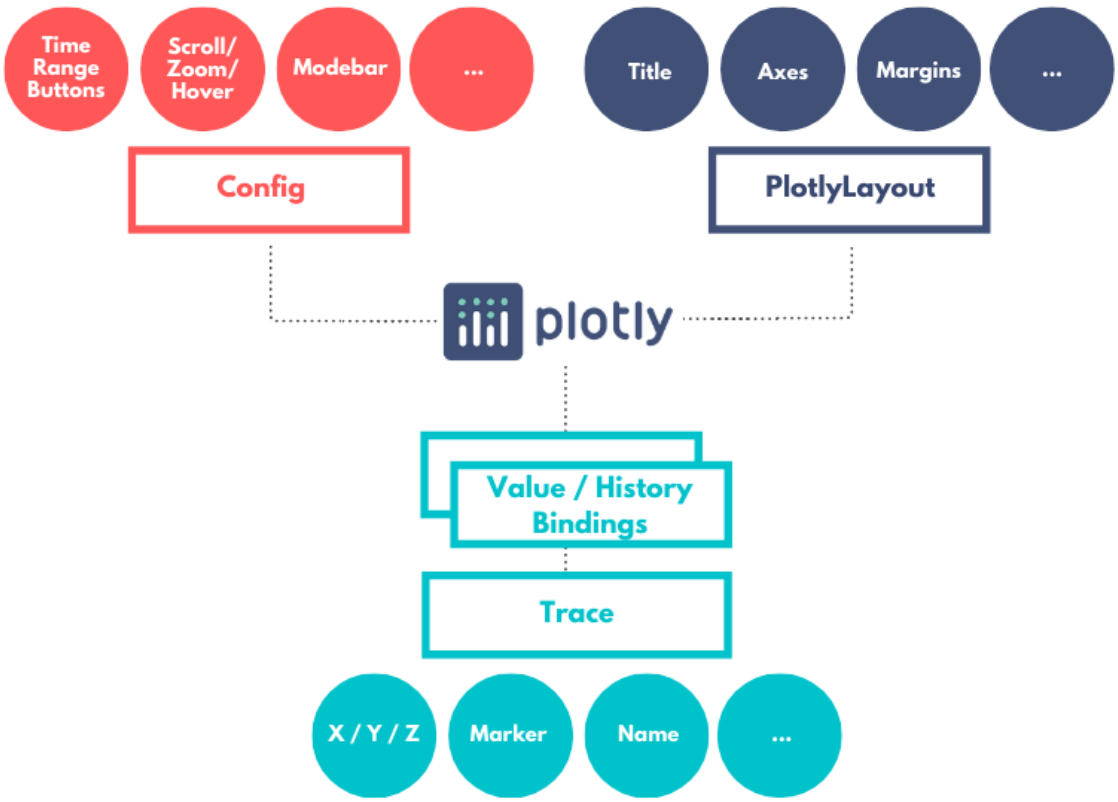


1.3 Plotly Structure

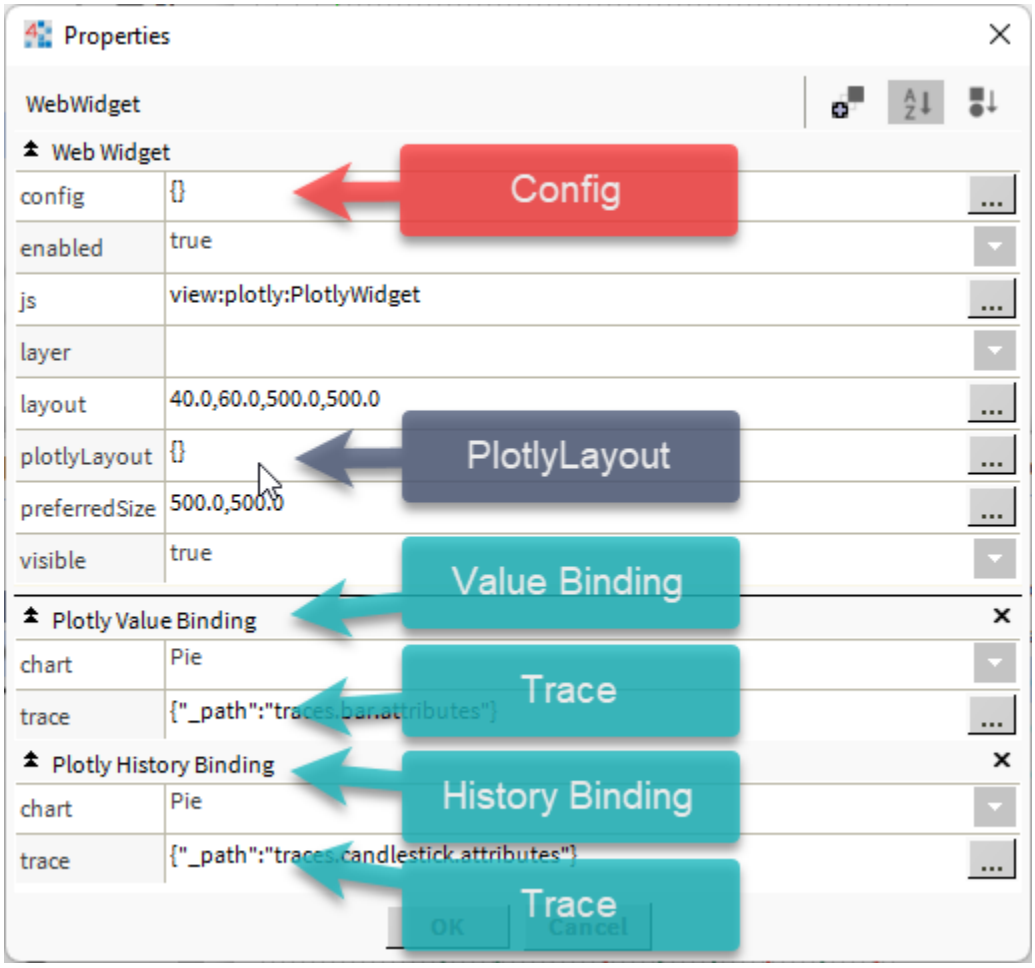
Plotly consist of the following main properties:

- **Value / History Bindings** are used to map a data the charts. Widget can contain two types of bindings: value and history. Each binding will be represented as a single data series. Value Binding is used to animate a Niagara numeric points and History Binding is used to animate histories.
- **Config** property defines high-level configuration options for the plot, such as the scroll / zoom / hover behaviour. Examples are documented here. The difference between **config** and **plotlyLayout** is that layout relates to the content of the plot, whereas config relates to the context in which the plot is being shown.

- **PlotlyLayout** object defines features that are not related to **traces** (like title, axis titles, and so on). We can also use the layout to add annotations and shapes to the chart. Please see full list of layout options here.
- **Trace** - each binding type will contain a trace property. Each trace is a dictionary type object that holds the values to be drawn. Each element of the chart is identified by a trace. A trace consists of a collection of data and the type of this data. Examples are documented here.



Same structure is maintained in Niagara:



- **JS, Layer, Layout, Enabled, Preferred Size, Visible** are standard Niagara properties, please refer to Niagara documentation for more help.

1.4 Quick Start

Plotly is shipped with the palette that contains examples of charts to get you started. Each palette example is linked to the dummy data source that can be relinked to the necessary data sources. Palette example previews are available here (only available in the Niagara Worbench help file).

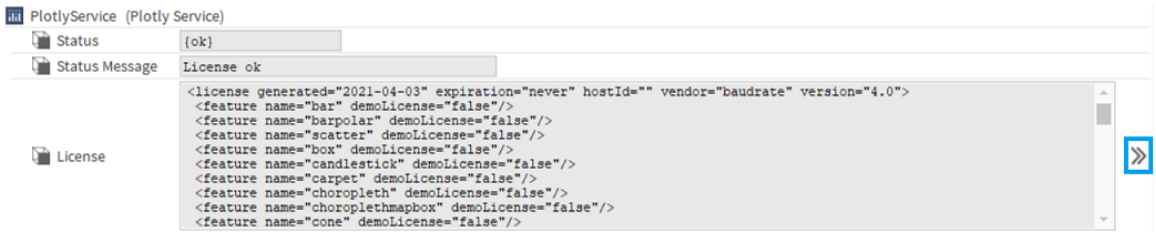
In order to learn how to build charts from scratch please refer to the value binding and history binding examples.

1.5 Requirements

- Niagara N4 4.8 or later powered device such as Jace, Supervisor or their OEM versions
- Plotly widget license file

1.6 Installation

1. Install plotly-ux.jar and plotly-doc.jar and all dependent modules via Software Manager.
2. Add **PlotlyService** to Services.
3. Open the **PlotlyService**. Press import icon and import the ***.license** file. Press **Save** button.
4. Right click on a service and press **Create Dummy Histories** action in order to create histories for palette examples (action will take up to a minute).
5. Drag'n'drop any of the widget examples from the palette to the px file.



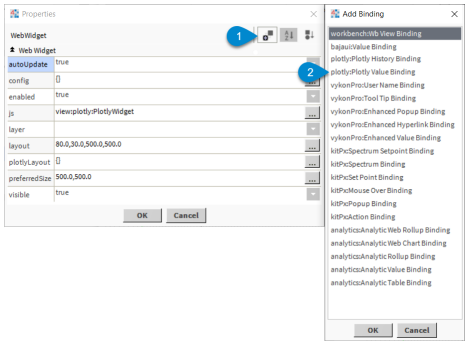
2 Value Binding

The binding represents a *trace*, a set of related graphical marks in a figure. Each trace must have a type attribute which defines the other allowable attributes. Value binding is also used to map single or several Niagara numeric points to the Plotly charts.

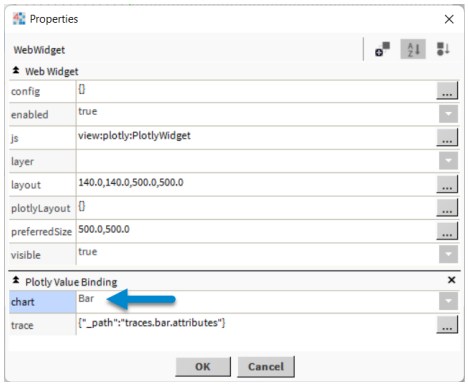
2.1 Quick Start

In this example we will build a simple bar chart.

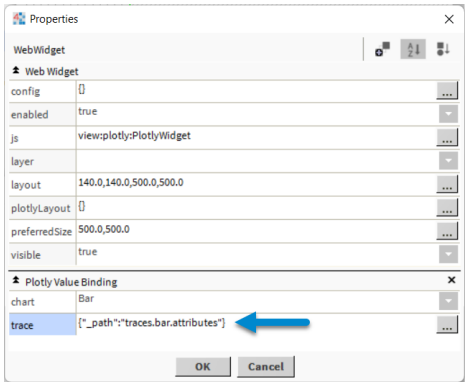
1. Drag and drop *PlotlyWidget* from the *Plotly* palette to the px file.
2. Click on *Add Binding* button and select *Plotly Value Binding* in the dialog box. Click *OK*.



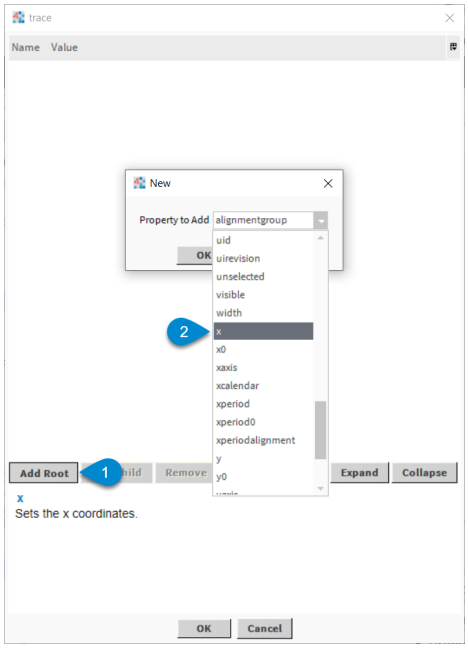
3. Select *Bar* chart type in *chart* property.



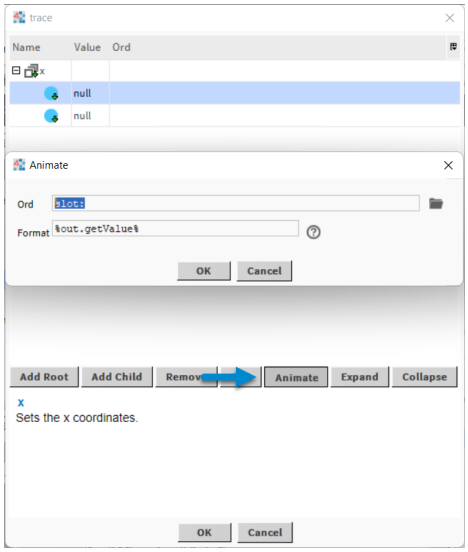
4. Press on *trace* property to open the dialog box.



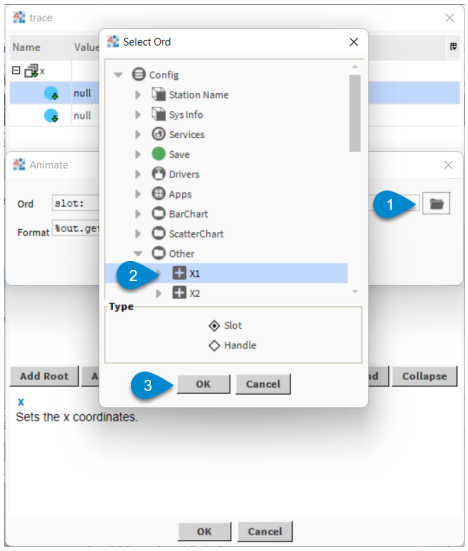
5. Click *Add Roots* and add x axis attribute.



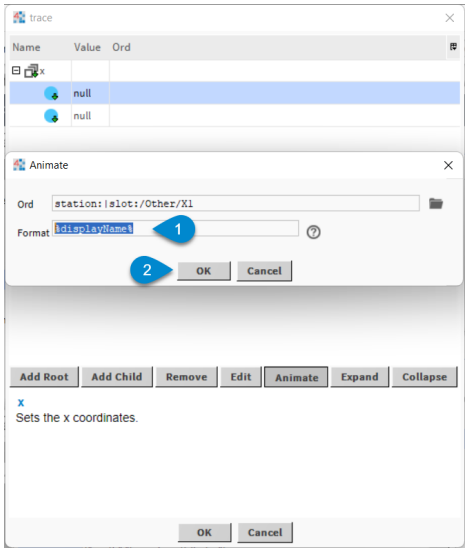
6. By clicking *Add Child* add the necessary number of points you are looking to add to the chart.
7. While child attribute is selected click *Animate* button.



8. Click on dialog box and select numeric points in Niagara station. Click *OK*.



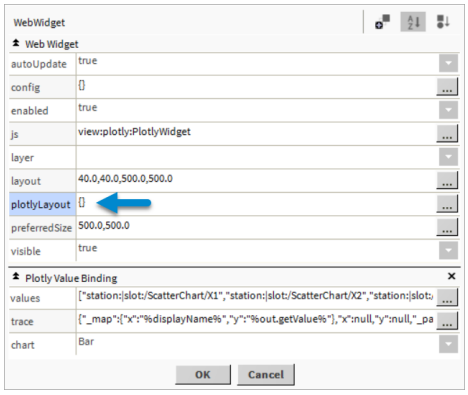
9. Enter *%displayName%* in *Format* field to display point display names on x axis. Click *OK*.



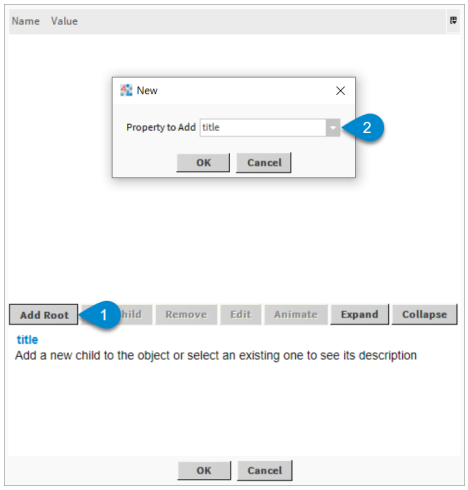
10. Click *Add Roots* and add *y* axis attribute.
11. By clicking *Add Child* add the necessary number of points you are looking to add to the chart.
12. While child attribute is selected click *Animate* button.
13. Click on dialog box and select numeric points in Niagara station. Click *OK*.
14. Enter *%out.getValue%* to display point *out* property on *y* axis Click *OK*.

    NOTE: For all attributes that are currently animated the background color will be changed to yellow.

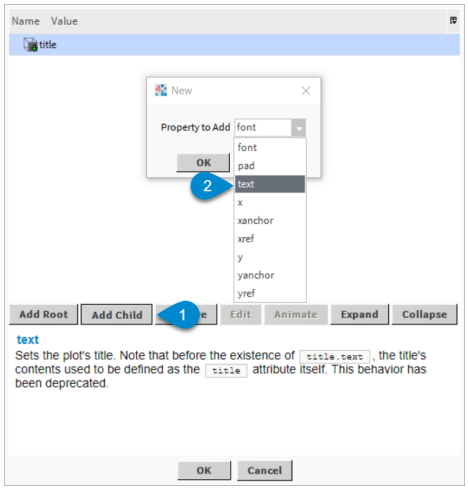
15. Click *OK* to close *trace* property dialog box.
16. Click *plotlyLayout* to open the dialog box.



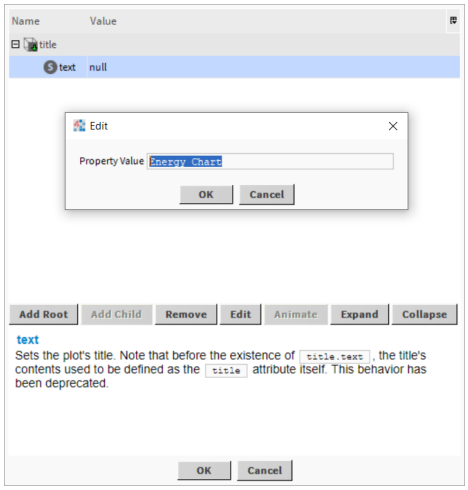
17. Click *Add Roots* and add *title* attribute.



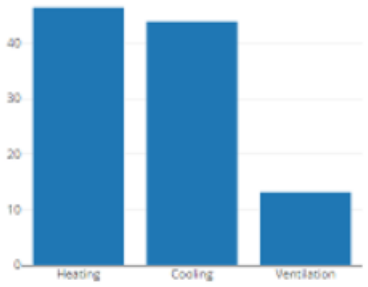
18. Select *title* attribute and click *Add Child* and add *text* attribute.



19. Double-click *text* attribute and enter the desired chart name.



20. Click *OK* to close *plotlyLayout* property dialog box.
21. Press *OK* to close the widget and see the result.



2.2 Property Description

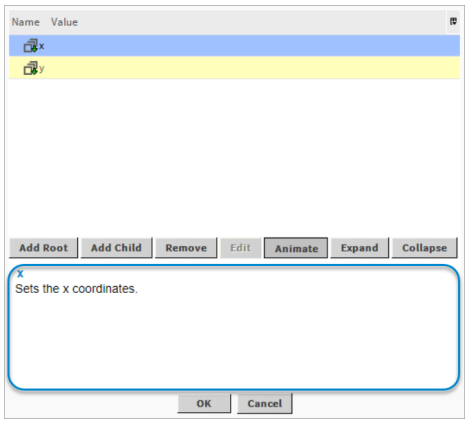
2.2.1 Chart

chart property is used to define the chart type. List of *trace* attributes depends on the *chart* type selected, so please always define the *chart* type first.

2.2.2 Trace

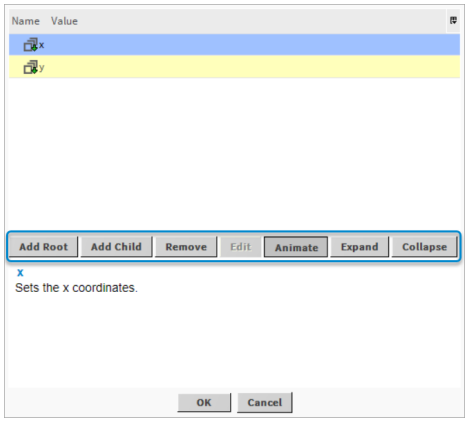
A *trace* is a collection of static and dynamic (animated) data and all other attributes describing the chart. Trace attributes are organized in a tree structure: root attributes might have sub-attributes or `children`, which in turn might also have sub-attributes etc. As the total number of possible attributes is very high (in a range of hundreds), they are not listed initially. Instead one has to add the attributes, which should be modified, and then set their values. It is not necessary to know each widget attribute. In fact, nobody knows that – there are just too many. It is also not needed, as all attributes by default have suitable values. So for a simple widget, which still will look good, only one or two attributes shall be changed. By changing more attributes one can create very beautiful and interesting widgets.

When clicking on *trace* property, a dialog box appears. Its top area contains added attributes, middle area contains buttons and bottom area has dynamic text, which describes selected attributes. This text helps to understand attribute purpose and possible values.



Buttons

1. Add Root [Ctrl+Insert] - to add trace root attributes – the ones on the top of a tree;
2. Add Child [Insert] - to add sub-attributes to root attributes;
3. Remove [Delete] - to delete attributes;
4. Edit [Ctrl+E] - to modify attribute value;
5. Animate [Ctrl+A] - to animate attribute value using ord values;
6. Expand [Ctrl+Down] - expands all root properties so that sub-attributes are visible;
7. Collapse [Ctrl+Up] - collapses all root properties and their sub-attributes;



Animate

Animating a property in Niagara means to link a widget property to a bound data component value, so that the widget can display any change in value as it occurs.

In order to animate press on *Animate* button and enter the desired BFormat in *Format* property and select data point in *Ord* property. Animated value will be displayed with a yellow background. To un-animate press on *Animate* button again. To edit *BFormat* double click on the property. Most common animation examples:

Trace Attributes	BFormat	Description
x,y	%displayName%	Bounds component display name
x,y,values	%out.getValue%	Bounds component out slot value
x,y,values	%in#.getValue%	Bounds component in# (replace # with number) slot value

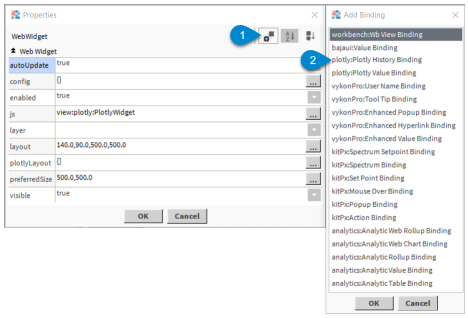
3 History Binding

The binding represents a *trace*, a set of related graphical marks in a figure. Each trace must have a type attribute which defines the other allowable attributes. History binding is also used to map Niagara history records to the Plotly charts.

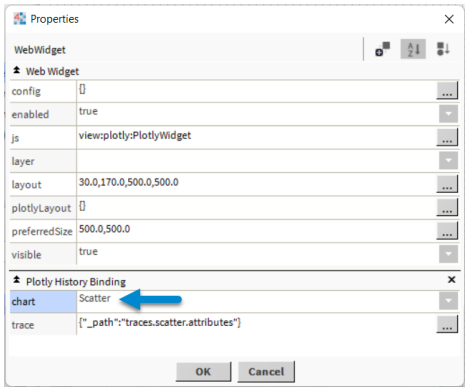
3.1 Quick Start

In this example we will build a simple scatter chart.

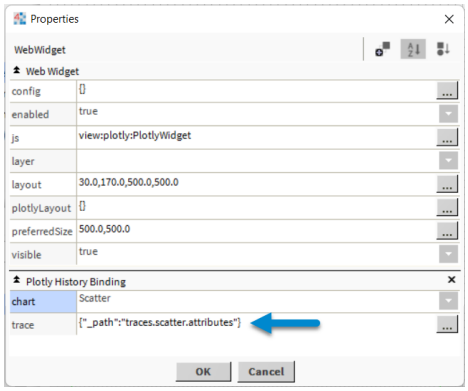
1. Drag and drop *PlotlyWidget* from the *Plotly* palette to the px file.
2. Click on *Add Binding* button and select *Plotly History Binding* in the dialog box. Click *OK*.



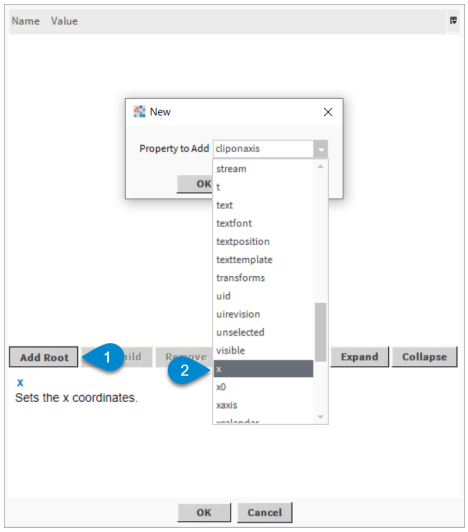
3. Select *Scatter* chart type in *chart* property.



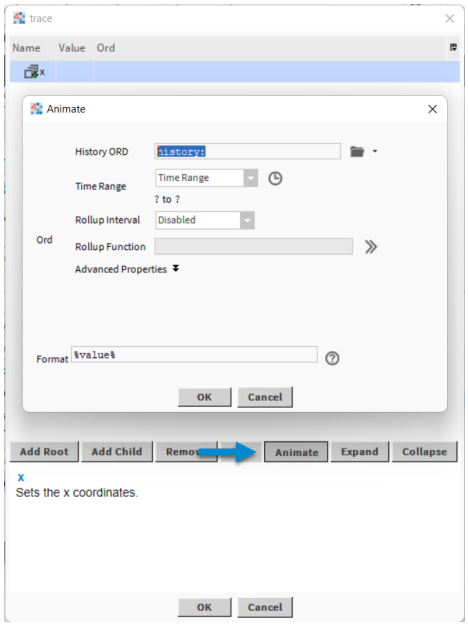
4. Press on *trace* property to open the dialog box.



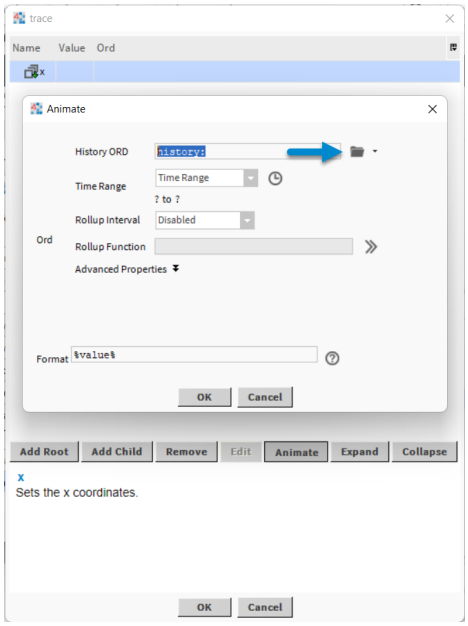
5. Click *Add Root* and add x axis attribute.



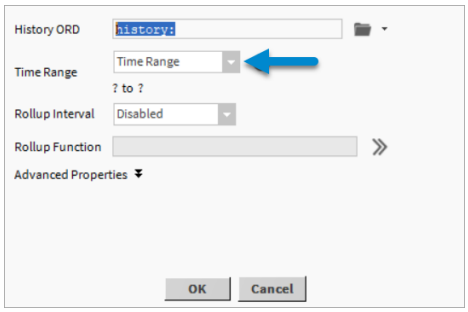
6. While x attribute is selected click *Animate* button to open the dialog box.



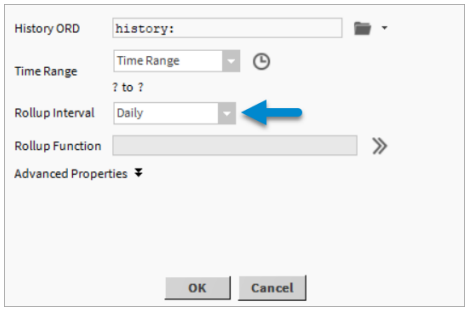
7. Click *History Ord Chooser* button and select history to be bound.



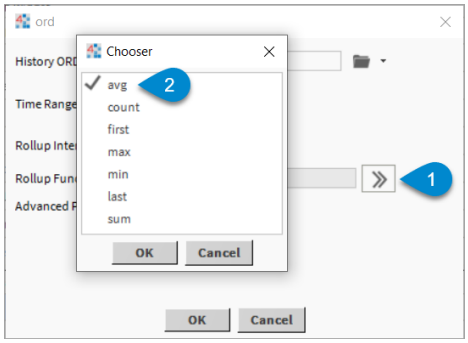
8. Select *Time Range* without specifying the period, that will show the full history length.



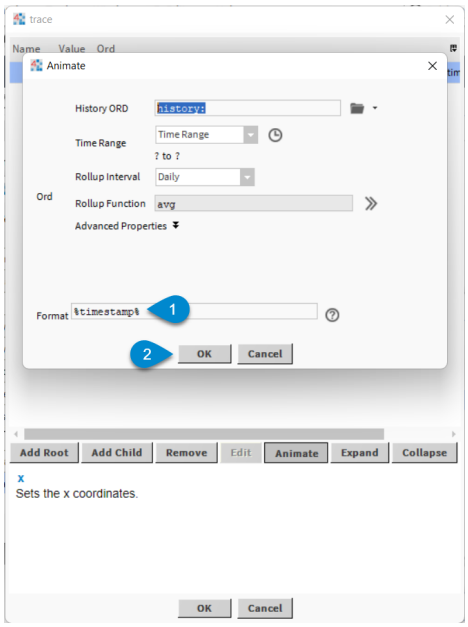
9. Optional Item. In order to roll up the history values select the desired *Rollup Interval*. For example if you have annual history you might want to select *Daily* interval.



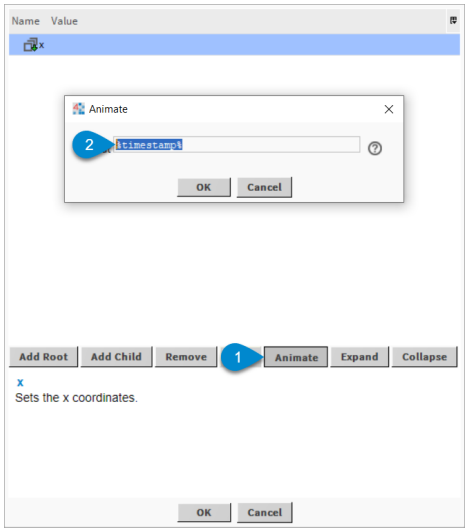
10. If *Rollup Interval* is used click on *Rollup Function* and select the necessary function. For this example let‘s select *avg*.



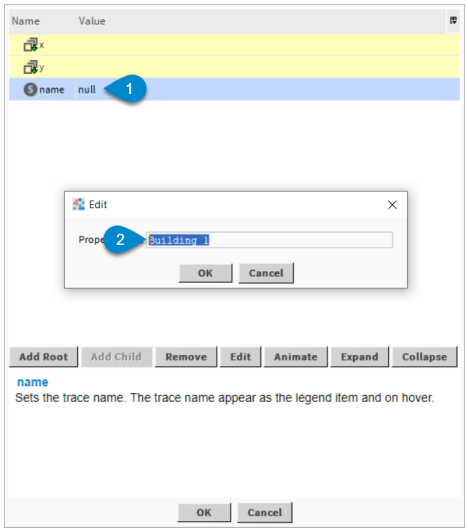
11. Enter *%timestamp%* in *Format* field to display history timestamps on x axis. Click *OK*.



NOTE: For all attributes that are currently animated the background color will be changed to yellow.

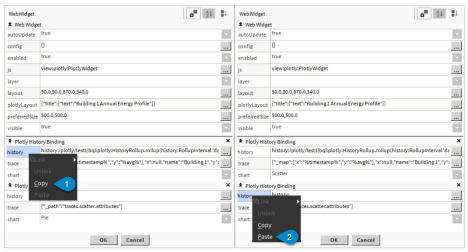


12. Click *Add Root* and add *y* axis attribute.
13. While *y* attribute is selected click *Animate* button to open the dialog box. Repeats steps 7-10.
14. Enter *%avg%* in *Format* field (if you have not used rollup function enter "%value%") to display history values on *y* axis.
15. Click *Add Root* and add *name* attribute.
16. Double-click on *name* attribute and enter a series name e.g. "Building 1".

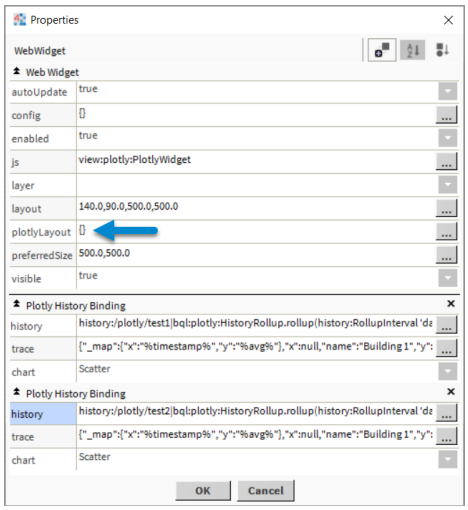


17. Click *OK* to close *trace* property dialog box.
18. Repeat steps 2 to 17 to add another series to the chart.

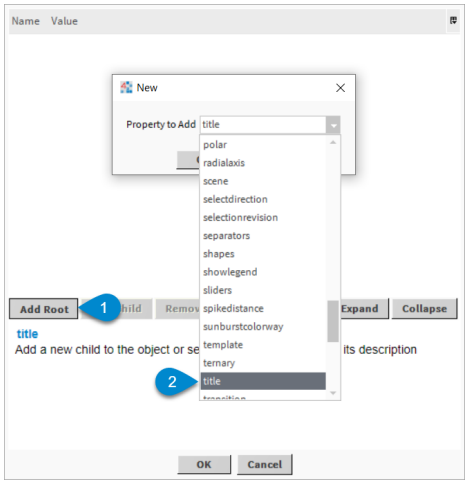
> ***NOTE:*** You can copy any of the properties. For example you can right click and press copy on the existing *history* properties. After that you can press paste to newly created *history* property.



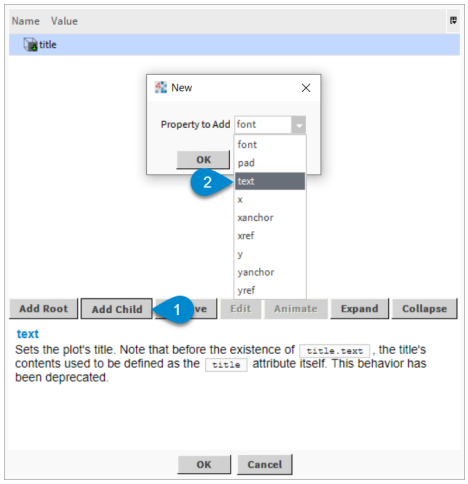
19. Click *plotlyLayout* to open the dialog box.



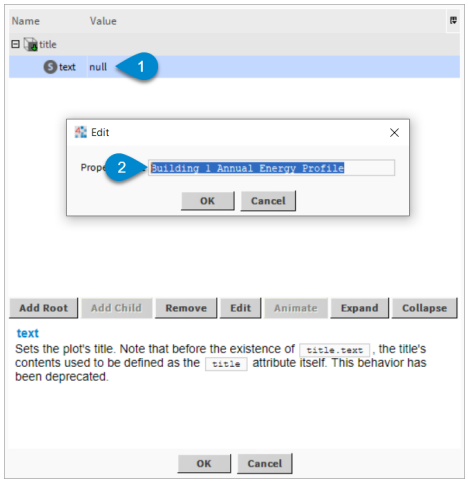
20. Click *Add Root* and add *title* attribute.



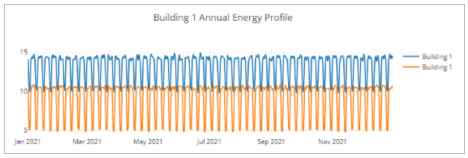
21. Select *title* attribute and click *Add Child* and add *text* attribute.



22. Double-click *text* attribute and enter the desired chart name.



23. Click *OK* to close *plotlyLayout* property dialog box.
24. Press *OK* to close the widget and see the result.



3.2 Property Description

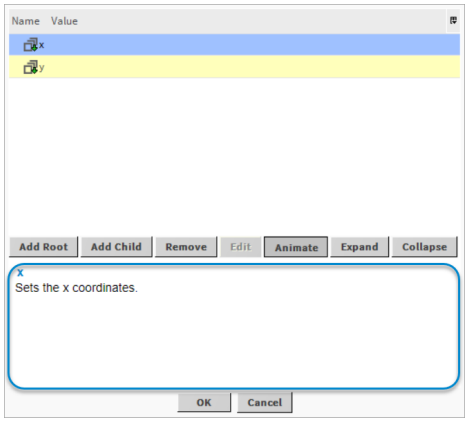
3.2.1 Chart

chart property is used to define the chart type. List of *trace* attributes depends on the *chart* type selected, so please always define the *chart* type first.

3.2.2 Trace

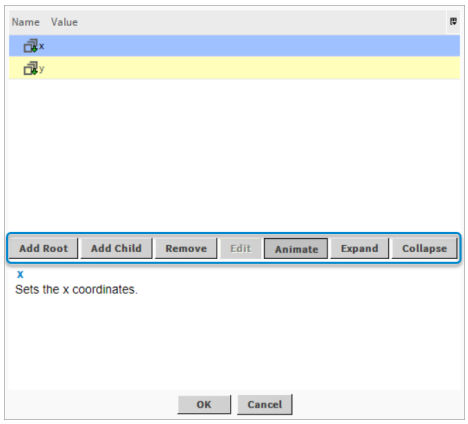
A *trace* is a collection of static and dynamic (animated) data and all other attributes describing the chart. Trace attributes are organized in a tree structure: root attributes might have sub-attributes or `children`, which in turn might also have sub-attributes etc. As the total number of possible attributes is very high (in a range of hundreds), they are not listed initially. Instead one has to add the attributes, which should be modified, and then set their values. It is not necessary to know each widget attribute. In fact, nobody knows that – there are just too many. It is also not needed, as all attributes by default have suitable values. So for a simple widget, which still will look good, only one or two attributes shall be changed. By changing more attributes one can create very beautiful and interesting widgets.

When clicking on *trace* property, a dialog box appears. Its top area contains added attributes, middle area contains buttons and bottom area has dynamic text, which describes selected attributes. This text helps to understand attribute purpose and possible values.



Buttons

1. Add Root [Ctrl+Insert] - to add trace root attributes – the ones on the top of a tree;
2. Add Child [Insert] - to add sub-attributes to root attributes;
3. Remove [Delete] - to delete attributes;
4. Edit [Ctrl+E] - to modify attribute value;
5. Animate [Ctrl+A] - to animate attribute value using ord values;
6. Expand [Ctrl+Down] - expands all root properties so that sub-attributes are visible;
7. Collapse [Ctrl+Up] - collapses all root properties and their sub-attributes;

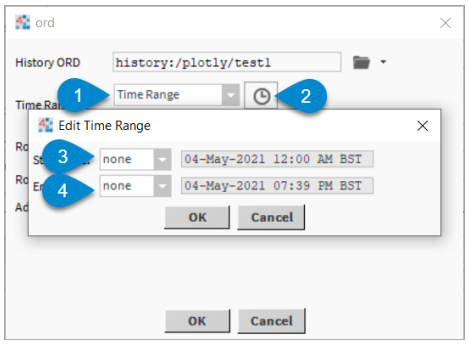


Animate

Animating a property in Niagara means to link a widget property to a bound data component value, so that the widget can display any change in value as it occurs.

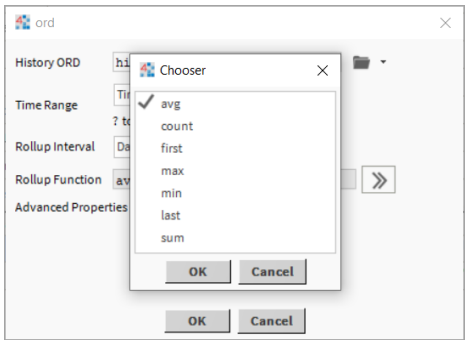
Animate dialog box consist of the following parameters:

1. History ORD - to select Niagara history record.
2. Time Range - to specify default history record time range. There are a number of predefined time ranges that can be selected from the dropdown menu. Also it is possible to select a custom time period by selecting *Time Range* from the dropdown, clicking a clock symbol and selecting the desired period.

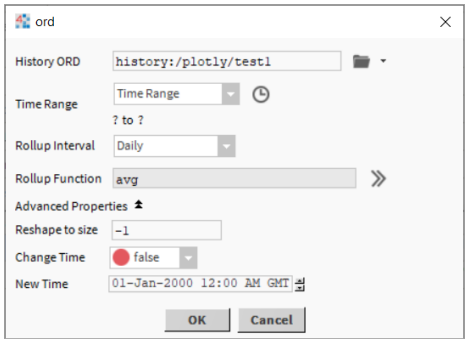


3. Rollup Interval - specifies an interval of time used to determine what (and how) data is presented in your chart. Each displayed point represents a designated time interval before the specified plot time. A rollup value of 1 hour will present data at a granularity level of every one hour, while a rollup value of 15 minutes will show data for every 15 minutes of logged data. Rollups are very useful to reduce granularity – you can collect data very often (e.g. once every minute), but still display fast and clean charts for a long time period (e.g. a year). Without rollups there would be too many data points, which could render charts cluttered and slow.

4. Rollup Function - specifies the functional operations against each record of the rollup period. Please note that BFormat used for animation will depend on the selected function. For example if you selected *avg* and *count* as your rollup functions you will be able to use them as *%avg%* and *%count%* to animate trace attributes. It is possible to use multiple rollup functions on the same chart – try *candlestick* or *ohlc* charts.



Advanced Properties



5. Reshape to size - this property is used to transform one-dimensional history into two-dimensional matrix, which is necessary for charts like *heatmap* or *contour*. It should be equal to the size of the first dimension of the matrix.

6. Change Time, New Time - allows to recalculate timestamps of the history by enabling *Change Time* and providing new starting time in *New Time* property. Useful to compare multiple time ranges of the same history. For example, it is possible to add two scatter traces, set one time range to *today* and the other to *yesterday* and change time for both – then the scatter widget will overlay both traces, which is very useful for comparison. Note: similar functionality could be achieved with multiple *xaxis* attributes.

7. Format - desired value *BFormat*. Only certain BFormat combinations are currently supported for he history binding:

Trace Attributes	BFormat	Description
x, y, z	%timestamp%	Bounds history timestamps
x, y, z	%value%	Bounds history values
x, y, z	%avg%	*Avg* rollup function needs to be selected in *history* property. Bounds history rollup calculated average values. All other rollup functions can be bound using the same approach.

4 Layout Options

The *plotlyLayout* attribute, whose value is referred below as *the layout*, containing attributes that control positioning and configuration of non-data-related parts of the figure such as:

- Dimensions and margins, which define the bounds of *paper coordinates*
- Figure-wide defaults: fonts, colors, hover-label and modebar defaults
- Title and legend
- Color axes and associated color bars
- Subplots of various types on which can be drawn multiple traces: xaxis, yaxis, scene, ternary, polar, geo, mapbox subplots
- Non-data marks: annotations, shapes, images
- Controls which can trigger Plotly.js functions when interacted with by a user: upatemenus and sliders

4.1 Frequently Used Parameters

- title.text - figure title.
- title.font.size - figure title size.
- title.font.color - figure title color.
- showlegend - enable/disable figure legend.
- margin - figure margins
- barmode - determines how bars are displayed on the graph. With "stack", the bars are stacked on top of one another. With "relative", the bars are stacked on top of one another. With "group", the bars are plotted next to one another centered around the shared location. With "overlay", the bars are plotted over one another, you might need to an "opacity" to see multiple bars.
- colorscale - represent a mapping between the range 0 to 1 and some color domain within which colors are to be interpolated (unlike discrete color sequences which are never interpolated).
- colorway - sets the default trace colors.
- modebar - list of configurations for modebar.

Full list of plotly attributes can be found here.

5 Configuration Options

In Plotly widgets it is also possible to control certain figure behaviours which are not considered part of the figure itself, i.e. the behaviour of the "modebar" and time range selector buttons, how the figure relates to mouse actions like scrolling etc. It is done using the *config* attribute.

Please note that Plotly Niagara manual for his secion includes interactive examples.

5.1 Time range selector buttons

Chart time range can be changed dynamically via buttons shown on the image below. Please note that the selected time range will not be saved and will be set to the default on the next chart reload. See *history binding* section for explanation how to set a default time range.

In order to add time range buttons add the following *modeBarButtons* attributes under *config* attribute:

Attribute	Value
modeBarButtonsToAdd	auto | timeRange | today | yesterday | weekToDate | lastWeek | monthToDate | lastMonth | yearToDate | lastYear

5.2 Frequently Used Parameters

- plotlyServerURL - use value `https://chart-studio.plotly.com` in order to export the chart to the chart studio
- modeBarButtonsToAdd.downloadJSon - allows to export the chart as JSON file
- displaylogo - to enable/disable plotly logo

6 Trace Options

Every aspect of a plotly chart (the colors, the grids, the data, and so on) has a corresponding attribute. Full list of plotly attributes can be found here.