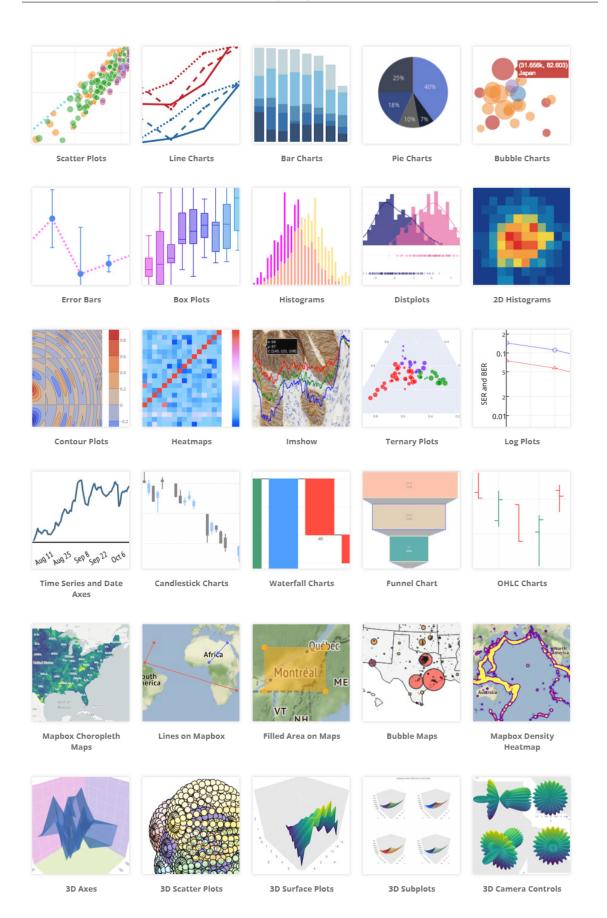
# Baudrate Data Analytics and Visualisation (DAV) Library

User Guide

# Contents

1	Introduction
1.1	Key Features
1.2	Requirements
1.3	Installation
1.4	Next Steps
	-
2	
2.1	Dav Service
2.2	Widget Data Structure    6
2.3	Attribute Editor         7
2.4	Animation
2.4.1	Format
2.5	Ord Types
2.5.1	Component Ord 11
2.5.2	History Ord
2.5.3	BQL and NEQL Ords
2.5.4	Series Transform Ord
2.6	Templates
2.7	DAV Ord Scheme
2.8	Relativization and Parameterization
2.8.1	Relativization
2.8.2	ORD Parameters
2.8.3	Attribute Parameters
2.9	Machine Learning
2.9.1	Trendlines
2.9.2	Forecasts
2.9.3	DAV Copilot
2.10	Additional resources
	Plotly.com
	Plotly Chart Studio
	Plotly.js JSON Editor
	Search Engines
	Single-Page Reference
2.10.0	5 mg/o 1 0 go 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3	Step-by-step Guide
3.1	Bar chart animated with numeric points
3.2	Scatter chart animated with numeric history

4	Attribute Reference
4.1	Trace Attributes         40
4.2	Layout Attributes
4.2.1	Frequently Used Layout Attributes
4.3	Config Attributes
4.3.1	Time Range Selector Buttons    41
4.3.2	Frequently Used Config Attributes



# 1 Introduction

A picture is worth a thousand words, and certainly when it comes to data, metrics and KPIs, nothing could be more right. Without visual tools such as charts and graphs we would soon get lost in the 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 visualisation. However, when it comes to large amounts of data and more complex data presentation along with sophisticated dashboards – Baudrate Data Analytics and Visualisation (DAV) library can help.

The library is based on plotly.js, which 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.js allows users to create beautiful interactive web-based visualisations that now can be displayed in standard Niagara PX files. Every chart type has hundreds of customisations options and almost every aspect of data visualisations 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.

Baudrate DAV library not just allow to insert plotly.js charts in Niagara PX views. It fully integrates all charts with Niagara framework with easy to use configuration UI, adds many ways to bind charts with real-time and historical data, relativize and parameterize charts, maintain consistent views with templates, set and switch time ranges for histories, introduce trend lines and forecasts, produce PDF reports, and much more.

# 1.1 Key Features

- over 40 unique chart types, including:
  - scatter: line, marker, area, bubbles
  - bars: grouped, stacked, overlay
  - pie: regular, pulled, donut
  - indicator: bullet, angular, datacard
  - heatmap: regular, contour
  - polar: bar, radar
  - sankey
  - sunburst
  - treemap
  - waterfall
  - geographical: scatter map, density map, choropleth
  - 3D: surface, scatter, mesh, volume
  - financial: OHLC, candlesticks, funnel
  - statistical: histogram, violin
- almost all chart aspects can be customized
- binding historical and real-time data
- BQL and NEQL queries
- absolute and relativized ords
- complex data queries with Series Transform
- interactive controls
- quick time range selection for historical data
- color schemes defined in templates

# 1.2 Requirements

- Tridium Niagara 4.8 or later powered device such as JACE, Supervisor or their OEM versions
- Baudrate DAV library license file

**NOTE:** If you have previously used a different version of our library, it must be removed prior to installing the DAV library. Specifically, please delete the plotly-ux.jar file from the 'modules' directory before proceeding with the DAV library installation.

# 1.3 Installation

- 1. Copy **dav-ux.jar** and **dav-doc.jar** into /modules folder of Niagara Workbench, restart Workbench and Niagara station in Application Director.
- 2. If licensed for a JACE, install the modules via **Software Manager** and restart its station.
- 3. Open the palette and add **DavService** to **Services**.
- 4. Open the **DavService**. Press import icon » near **License** property and import the provided **.license** file. Press **Save** button and make sure **Status Message** shows "License ok".
- 5. Right-click on the service and select **Create Dummy Histories** action. This action creates a few histories to provide data for DAV library palette widgets.
- 6. Drag & drop any of the widget examples from the palette to the px file. You should see the widget with populated data.

# 1.4 Next Steps

The DAV library is shipped with the palette that contains examples of charts to get you started. Each palette example is bound to the dummy data source that can be re-bound to the actual data sources.

Note: this guide is available in two formats: as a PDF document and as a part of Niagara help file. Niagara help file includes interactive chart examples.

To understand main library concepts and components it is highly recommended to read the Component Guide section.

Then, in order to learn how to build charts from scratch, see Step-by-step Guide section.

Chats could be configured with lots of attributes. For a full description of all of them see Attribute Reference section.

Status       {ok}         Status Message       License ok         License <li>(license generated="2022-11-12" expiration="never" hostId          <feature demo="false" name="bar"></feature></li>
<pre><li>license generated="2022-11-12" expiration="never" hostId</li></pre>
<pre></pre> <pre>&lt;</pre>
Default Template dark 🗸
Templates Component
History Views History Views
Playground Folder

Figure 1: Dav Service

# 2 Component Guide

This section describes main library concepts and components one should know about in order to create new charts.

# 2.1 Dav Service

The station shall contain one **DavService** component installed in **Services** container. This component is responsible for licensing, templating and other features. It also stores test points used by DAV palette widgets. **DavService** properties:

- Status {ok} if the service is enabled and licensed.
- Status Message explains the status value, useful for debugging.
- License displays license file content; press >> button to import the license file.
- Default Template a combo-box to select the default template.
- **Templates** a container with all templates, which can be applied to widgets. Please refer to Templates section.
- **Playground** a folder with a few Niagara points and history generators to provide data for palette widgets.

DavService have Create Dummy Histories action to generate a few histories stored in History/dav container.

# 2.2 Widget Data Structure

DAV library charts are described declaratively as JSON objects. Every aspect of a DAV library chart (the colors, the grids, the data, and so on) has a corresponding JSON attribute. In Niagara each widget is configured with one or multiple **Dav Bindings** and two properties: **davLayout** (also referred as **layout**) and **config**.

WebWidget	<b>∮</b> ≜↓ <b>↓</b>
Web Widge	t
davConfig	{`davServerURL`:`https://chart-studio.dav.com`,`displaylogo`:false,`show
davLayout	{`title`:{`text`:`Bar Customization`},`yaxis`:{`range`:[0,120]}}
enabled	true 🗸
js	view:dav:DavWidget
layer	▼
layout	750.0,330.0,500.0,500.0
preferredSize	500.0,500.0
visible	true
Dav Bindin	g ×
chart	Bar
trace	{`_map`:{`y[5]`:{`ord`:`station: slot:/Services/DavService/Playground/Buils
	OK Cancel

Figure 2: Niagara widget properties

- Dav Bindings are used to set widget type, data and type-specific attributes. Each binding has two properties: chart for type of the widget and trace, a set of attributes to configure various aspect of the chart. The attributes can be bound to real-time points or histories. They can be bound with various ord schemes: slot, history, bql, neql, seriesTransform etc. All attributes are described in Trace Attributes section.
- **davLayout** property defines attributes that apply to the rest of the chart, like title, xaxis, annotations. The detailed description is in Layout Attributes section.
- **davConfig** property defines high-level configuration options for the plot, such as the scroll / zoom / hover behaviour. The detailed description is in Config Attributes section. The difference between **config** and **davLayout** is that **davLayout** refers to the content of the plot, whereas **config** refers to the context in which the plot is being shown.
- enabled, js, layer, layout, preferredSize, visible are standard Niagara widget properties, please refer to Niagara documentation for more help.

# 2.3 Attribute Editor

Widget properties **trace**, **davLayout** and **davConfig** are collections of attributes describing the widget. Attributes are organized in a tree structure: one or multiple root attributes, which might have sub-attributes or children, which in turn might also have sub-attributes etc. As the total number of DAV library attributes is very high (in a range of hundreds), they are not shown 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

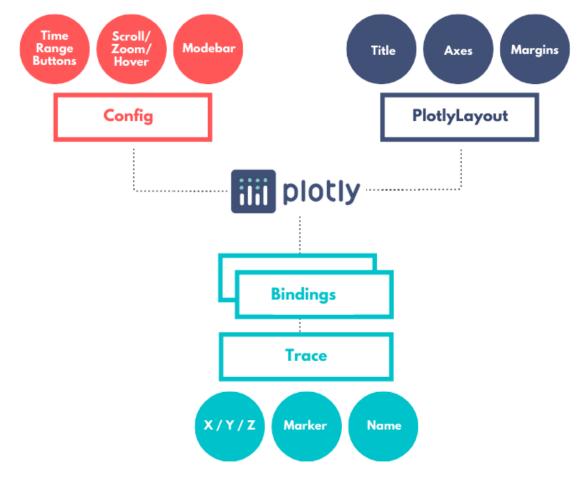


Figure 3: DAV object structure

good, only one or two attributes should be changed. By changing more attributes one can create very beautiful and interesting widgets.

When clicking on **trace**, **davLayout** or **davConfig** property, a new 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.

Note: **trace** attributes are different for every chart type, therefore **chart** property shall be defined first.

Buttons:

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

**Add Root** opens a selector with all root attributes available for this object. The bottom area of the dialog box displays information of a selected attribute: its type, default value and description. After the attribute is added, it is possible to **Edit** its value. In case of complex attributes, it might be necessary to **Add Child** first. That will open the attribute selector containing all root children. Unused attributes can be deleted with **Remove** button.

# 2.4 Animation

Attributes of **trace** property can be animated – dynamically bound to Niagara points, components and histories. Animating 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, use **Animate** button and enter **Ord** (using ord chooser or manually) and **Format**. Animated value will be displayed with a yellow background. To un-animate press on **Animate** button again.

Various types of ords and ord choosers are available:

- Component Chooser to select ord to station component, which slot value should be displayed. For more information see Component Ord section.
- History Chooser to select Niagara history, which time-series values should be displayed. In addition, the time range, rollup period and function can be specified. The history can also be reshaped into matrix. For more information see History Ord section.
- Transform Chooser to select the Series Transform graph, which allows complex transformation of Niagara histories. For more information see Series Transform Ord section.
- BQL Query Builder to run BQL query.
- NEQL Query to run NEQL query (no query builder is available, should be entered manually). For more information see BQL and NEQL Ords section.

#### 2.4.1 Format

In addition to the **Ord** property, the **Format** property is used to extract specific information from bound components and to do its processing. Format is based on a standard Niagara BFormat feature: a regular text with embedded scripts denoted by the % character. Calls within the script are mapped to an object's methods, and can be chained using the dot operator. It is also possible to embed mathematical operations, array operations and JavaScript function calls to modify values for widgets.

Examples for numeric points:

Name	Value	Ord	₽
🗆 🗽 xaxis			
Atickangle	-45	Mew 1	×
E barmode	group		
🗆 词 title		Property to Add activeshape	- 1
S text	Grouped Bar Chart	height	<u>-</u>
🗆 🕞 yaxis		hiddenlabels	
□ □ □ □ range		hidesources	
	-	hoverdistance	
	0	hoverlabel	
	130	hovermode	
		iciclecolorway	
		images legend	
		mapbox	
		margin	
		meta	
		modebar	
		newshape	
		papar brealar	Ψ.
Add Root Ad	d Child Remov	e Edit Animate Expand Colla	apse
hovermode			
	d.oneof( <b>x</b>   <b>v</b>	closest   false   x unified   y unified ).	
Default <b>x</b>	-, (, )		
Determines the n	node of hover inte	ractions. If closest, a single hoverlabel wil	
		the hoverdistance. If <b>x</b> (or <b>y</b> ), multiple	
		points at the <b>closest</b> x- (or y-) coordinate	
		he caveat that no more than one hoverlabe	
		OK Cancel	

Figure 4: Attribute editor

🐴 Anim	nate	×
Ord	<pre>station: slot:/Services/PlotlyService/Playground/Building1/H</pre>	-
Format	<pre>\$out.value\$</pre>	<u>C</u> omponent Chooser
		<u>H</u> istory Chooser
	OK Cancel	<u>T</u> ransform Chooser
		<u>B</u> ql Query Builder

Figure 5: Animate dialog box

- %out.value% returns value of the point to be used as data in widget attributes like x, y, values.
- %displayName% returns user-friendly name of the point to be used in widget attributes like labels.
- %anyName% returns value of extra property anyName, which can be added in the Niagara slot sheet.
- %out.value% \* 10 returns out.value multiplied by 10.
- %out.value% > 100 ? 'red' : 'blue' returns a red or blue text string depending on out.value. Can be used to dynamically change colors.

For Niagara tables (histories, series transform) **Format** property shall specify the name of table column:

- %timestamp% returns history timestamps to animate attributes like x in Scatter widgets.
- %value% returns values from non-rollup histories. Can be used for y attribute in Scatter widgets.
- %max%, %min%, %avg% returns max, min and average values for each period in rollup history. Useful for high and low attributes in Candlestick widgets.
- %first%, %last% returns first and last values for each period in rollup history. Useful for open and close attributes in OHLC widgets.

It is possible to apply JavaScript array functions to the whole history column:

- %value%.slice(0,10) returns 10 first values from the value column.
- %max%[0] returns the first value from the table max column as a scalar. Can be used in Indicator.
- %min%[%min%.length-1] returns the last value from the min column.

There is a special syntax of **Format** allowing to process every value in the column separately: two percentage signs instead of one  $- \frac{1}{3}$ . Example:

• (%%status%% & 8) == 8 ? 'red' : 'green' - encode the alarm flag in the history status column as a color string to animate marker.color.

# 2.5 Ord Types

#### 2.5.1 Component Ord

This is the basic ord, pointing to any Niagara station components, usually to numeric points. The widget subscribes to these components and displays their slots as specified by **Format** 

	🐕 trace X
Basic Sankey	Name Value Ord 🕫
	🗄 🍞 node
	€ orientation h
	🗆 📷 link
	E di source
81	D 🕞 value
	null station: slot:/Services/DavService/Playground/Building1/Heating
	null station: slot:/Services/DavService/Playground/Building1/Cooling
A1	Add Root Add Child Remove Edit Animate Expand
	Collapse
CI	value Type: data_array
	Default []
	A numeric value representing the flow volume value.
82	
A2	
c.	OK Cancel
	Animate X
	Ord station:/slot:/Services/DavService/Playground/Buildingl/Heat
	Format \$out.value\$ (?)

Figure 6: Slot Ord

property in real-time. For example ord station:|slot:/Folder/NumericPoint1 subscribes to Folder/NumericPoint1.

#### 2.5.2 History Ord

History ord points to Niagara history and also specifies a few extra parameters:

- **Time Range** to specify default history record time range. There is a number of time ranges that can be selected from the dropdown menu:
  - Standard ones: Today, Yesterday, This Week, Week To Date, Last Week, This Month, Month To Date, Last Month, This Year, Year To Date, Last Year, All.
  - Custom Range: fixed time period.
  - **Current** and **Previous** periods: specify a certain number of minutes, hours, days, weeks, months and years including (Current) or excluding (Previous) the current one.
  - **From** selector: the most powerful one for very complex periods, like "go back 2 months and select 3 first weeks". This selector allows to easily specify multiple periods of the same history for comparison, e.g. overlay and compare 4 last weeks of the same history on the same line chart.
- Rollup Interval specifies an interval of time used to determine what 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 can render charts cluttered and slow.

	🌆 trace		×
History Scatter with Date Slider	Name	Value	Ord 🛤
25	<b>⊿</b> ×		history:/plotly/test1?period=timeRange;start=2021-01-01T00:00:00.000
Building 1	S name	Building 1	
20 III. I IIII IIII Building 2	🗗 у		history:/plotly/test1?period=timeRange;start=2021-01-01T00:00:00.000
	👫 History C	Chooser	×
10. What is a standard that is a standard to a standard the standard to a standard to a standard to a standard	History ORD	histo	ory:/plotly/testl
	Time Range	All	<b>•</b>
-1 Jan 3 Jan 10 Jan 17 Jan 24 Jan 31	Rollup Interv	al Hourly	y 👻
2021	Rollup Functi	ion avg	*
MIANAA INANA INANA INANA IN	Advanced Pro	operties 🗙	
	Reshape to si	ize -1	
			OK Cancel

Figure 7: History Ord

- Rollup Function specifies the functional operations against each record of the rollup period. Please note that Format 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 see examples in palette candlestick or OHLC charts.
- **Reshape to Size** this property is used to transform one-dimensional history into a twodimensional matrix, which is necessary for charts like heatmap or contour. It should be equal to the size of the first dimension of the matrix.

#### 2.5.3 BQL and NEQL Ords

BQL and NEQL queries allow searching to Niagara components by name or by tags. The widget then subscribes to all found components and displays their values. This type of ords is great to make relativized universal views. For example, show a bar chart with all room temperatures on a certain floor. One widget will be able to display any number of temperature values, whether there are 5 or 50.

Both BQL and NEQL can be used for such a task. BQL queries rely on slot names and values: e.g. find all points whose names end with "Temp" and with alarm extension. Note: BQL queries should return component slotPath in the first column, to allow subscription.

NEQL queries rely on semantic tags – a more modern and flexible approach. For example query station:|slot:/BuildingA/Level01|neql:n:point will return all Niagara points stored in a folder /BuildingA/Level01.

#### 2.5.4 Series Transform Ord

Series Transform is a standard Tridium Niagara module, which provides a way to manipulate Niagara histories, in order to create new series data. Series Transform is not well-known among the integrator community, although its graphs compensate for a missing JOIN operator in the Niagara BQL language. With Series Transform one can, for example, merge two or more histories using their timestamps. The resulting query produces a set of columns, each of which can be bound to a widgets attribute.

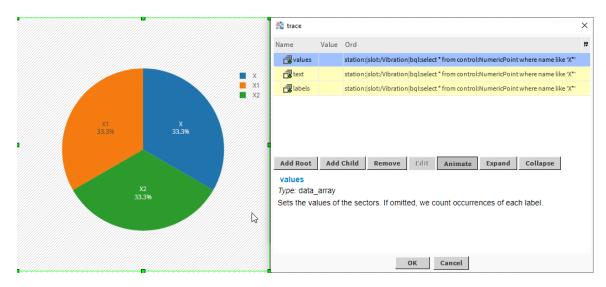


Figure 8: BQL Ord

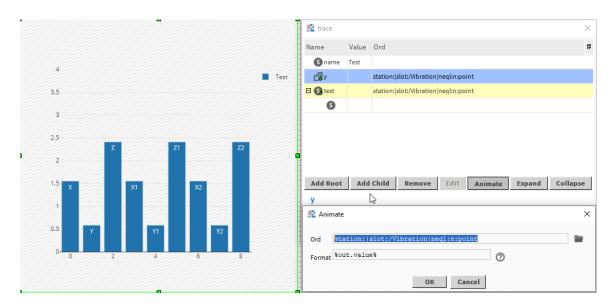


Figure 9: NEQL Ord

		📲 🌠 trace							
	Power vs Temperature	Name	Value	Ord	ţ.				
		🚩 mode	markers						
120kW		🗆 🕞 marker							
		💦 size	12						
100kW		tolor		transform:slot:/SeriesTransform/Tra	insformGraph1				
		_ <b>i</b> x		transform:slot:/SeriesTransform/Tra	insformGraph1				
80kW	ROLIN			transform:slot:/SeriesTransform/Tra	insformGraph1				
GORT		S text		transform:slot:/SeriesTransform/Tra	insformGraph1				
60kW		S hovertemplate	96{x:.1f}°C-96{y:.1f}kW@	0					
CONT		Add Root Ad	d Child Remove	Edit Animate Expand	Collapse				
40kW	•	×							
		Animate			×				
20kW									
			ot:/SeriesTransform,	/TransformGraph1	_ <b>•</b> •				
0°C	5°C 10°C 15°C 20°C	Format %temp%		0					
			ОК	Cancel					

Figure 10: Series Transform Ord

Ord example: transform:slot:/Folder/TransformGraph1 – assuming this graph combines outdoor temperature history and energy consumption history, its result can be used to draw XY scatter showing how energy consumption (Y) depends on outdoor temperature (X).

For more information about Series Transform, search for it in Tridium Niagara help.

# 2.6 Templates

Templates are powerful tools to maintain and update graphical design of charts, including their color scheme. The library is shipped with nine predefined templates and users can quickly customize them or make their own template to preserve company brand or to make fast changes to all widgets at once to suit customer wishes.

Each template is a collection of attributes to set default values for **trace**, **davLayout** and **davConfig** objects. The **Templates** property under **DavService** contains a set of named templates. Templates are editable with the standard attribute editor, which allows to add, remove, edit all chart attributes in a tree-like structure. In order to add a new template, **Duplicate** an existing template, rename and edit its attributes.

Template properties and attribute editor

Every time a new chart is rendered in a web browser, it sends its **davLayout** / **template** value to the **DavService**. Depending on the value, it receives back the contents of one template and applies its values as a default attribute. The value of **davLayout** / **template** can be:

- none no template will be applied
- template name the requested template will be applied. E.g. **red** will return a template called 'red'.
- empty or not set the template selected in **Default Template** property under **DavService** will be returned.

By default, the classic template is applied to all widgets. If **Default Template** property is changed, all widgets will be redrawn after the next page reload. Note, the template name can also be passed to PX file via hyperlink's Attribute Parameters.

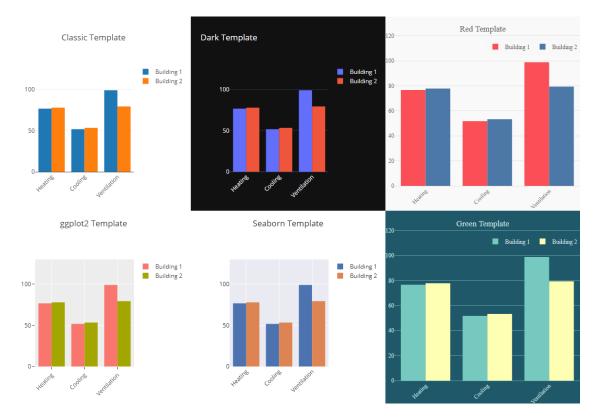


Figure 11: Template examples

Status	{ok}		🏰 Template				×			
🗎 Status Message	License ok		Name	Value	Ord		E			
	<li>clicense generated="2021-05-15" expiration="never" hostId <feature demo="false" name="bar"></feature> <feature demo="false" name="barpolar"></feature></li>		⊞ layout	PLOTLY DARK						
License	<feature demo="false" name="scatter"></feature> <feature demo="false" name="box"></feature>	»	🗄 📷 data							
License	<feature demo="false" name="candiestick"></feature> <feature demo="false" name="carpet"></feature>	//	Config							
	<pre><feature demo="false" name="choropleth"></feature> <feature demo="false" name="choroplethmanbox"></feature> </pre>		B showTips	false						
🗎 Default Template	classic v									
Templates	Component									
Classic Tem	plate		Add Root Add	d Child Rei	nove Edit	Animate Exp	and Collapse			
🔻 🍄 dark 🛛 Tem	plate		layout							
Value {"1		Type: object containing one or more of the keys listed below								
ggplot2 Tem	plate									
🕨 🍘 gray 🛛 Tem	plate									
Seaborn Tem	plate									
White Tem	plate									
🕨 🍄 red 🛛 Tem	plate									
🕨 🅐 green 👘 Tem	plate				OK Car	ncel				
🕨 🍘 blue 🛛 Tem	plate									

O Playground Folder

Figure 12: Template properties

👫 Propertie	25	×	🚰 Animate 🛛 🕹
CanvasPane	e źł	∎⊥	Binding
✿ Canvas Pa			bajaui:Value Binding [station: slot:/Services/DavService dav:layout/paper_bgcolor]
background	(default)	-	Converter Type
enabled	true	-	String To Brush
halign	Center	-	🖬 String To Brush
layer		-	
maxScaleFact	c1.00		
minScaleFact	0.50		
scale	Fit Ratio	-	
valign	Center	-	
viewSize	1000.0,800.0		
visible	true	-	
▲ Value Bind	ling	×	
ord	station: slot:/Services/DavService dav:layout/paper_bgcolor		
degradeBeha	None	-	OK Cancel
hyperlink	null		<u> </u>
summary	%displayName?typeDisplayName% = %.%		
popupEnable	true	-	
	ОК Сапсеl		

Figure 13: DAV Ord Scheme

# 2.7 DAV Ord Scheme

DAV library ord scheme **dav:** allows to use default template attributes in a regular Niagara bindings in order to change PX background when the template get changed. The ord station:|slot:/Services/DavService|dav:layout/paper\_bgcolor and converter StringTo-Brush will update the background color every time layout.paper\_bgcolor is changed in the default template.

#### 2.8 Relativization and Parameterization

Niagara allows to create one PX file and then reuse it to see various sets of data. There are a few ways how that could be accomplished for Baudrate charts.

#### 2.8.1 Relativization

Data ORDs, used for attribute animation, can be relative or absolute. The absolute path ensures that the data always resolves to a single unique component (slot:/FCU1/HeatValve) in the location that is specified by the ORD, regardless of where the PX file or the parent component is located. If the same PX file is attached to a view that belongs to a different component, the ORD will still resolve to the original component (HeatValve) because of the absolute path. However, if you make data binding relative (slot:HeatValve), then the path will resolve relative to its current parent ORD. This relative path makes the PX file resolve data bindings correctly to identically named components that reside in different locations, thus making one PX file usable in many views. So for parents slot:/FCU1, slot:/FCU2, and slot:/Floor2/FCU20, this binding will resolve slot:/FCU1/HeatValve, slot:/FCU2/HeatValve, and slot:/Floor2/FCU20/HeatValve.

To relativize PX file means to translate absolute bindings to relative ones. In order to automatically relativize ords, first create widget with absolute ords, add PX file as a view to a parent component it shall display, select the widget and choose **Relativize Ords** button in DAV Copilot toolbar. A

File Edit Search Bookma	rks Tools Wind	dow PxEditor	Help								Q	uick Search			
	C 🗗 🗉	<b>•</b> • •		G % @	i De	×	h /*	- 1	å	a P	<u>⊟</u> ⇔	<u>.</u> 5	14	Q	Q >
My Host : DESKTOP-JM2Q3AE (davTest2)	: Station (davTest2)												/		Building
• Nav	-2	(							-	Bound (	Ords				-7
N O 🛛 🕲 My Network														/	AB
_															
Templates     History View															
<ul> <li>Playground</li> </ul>															
🔻 🔘 Buildir									-	Widget	Tree				2
) 🔽 H	eating									wiuger	nee			_	12
	ooling									_		1	•		- 14
	entilation								-	Scroll					11
El	ectrical Relativize Ords											ne (Content) /idget [null]			_
so so											webw	naget (nuttj			-
Te	belore				After					Prop	erties				-7
E N	✓ station: slot:/Ser	rvices/DavService/Pla			slot:Heating					_				$\oplus$	×
▶ 🚯 B0	DQ	rvices/DavService/Pla			slot:Solar					2					10
- Palette		rvices/DavService/Pla			slot:Lightin;					_					
🖿 🗙 🔊 着 dav		rvices/DavService/Pla			slot:Electric					- 11					
		rvices/DavService/Pla			slot:Cooling					Laye	rs				-7
DavService	✓ station: slot:/Ser	rvices/DavService/Pla	iyground/Bui	lding1/Ventilatio	n slot:Ventilat	ion				Luye	15			Ð	×
DavWidget     Ear Charts														G	
Barcharts     Basic Bars			2	ок	Cancel					IS					₽
Grouped Bars			-		ounder					_					
Stacked Bars	_	11								_					_
Overlay Bars		Heating							-	Propert	ies				-7
Customized Bars		0	10	20	30	40	50		Wet	Widget			o	Å↓	∎Ļ
History Bars									<u> </u>	Veb Widget				_	^
<ul> <li>D         Business Charts     </li> <li>         Heatmaps     </li> </ul>		-			_						}	National States			
Indicators											mue	`gridcolor`:	white		
Maps									r .	DAV Co					
Misc										scribe the v		p 1		A-B	
Pie Charts		G									-	$\sim$			F
Polar Charts															
<ul> <li>Sankey Charts</li> <li>Scatter Charts</li> </ul>															
Scatter Unarts	· · · · · ·														

Figure 14: DAV Toolbar

dialog box will show all ords, that will be changed. After that the PX file can be used in all similar components.

#### 2.8.2 ORD Parameters

Another way to reuse one PX file to visualize multiple components is to use ORD parameters in hyperlinks. Any part of ORD could be replaced with \$(var) parameter, where var is a parameter name, e.g. \$(ord1), \$(ordA), \$(ahu). Then the value for these parameters could be passed in a hyperlink.

For example, ORD history:/demo/\$(ord)?timeRange=yesterday becomes

#### history:/demo/HeatValve?timeRange=yesterday

if the hyperlink to the view contains parameter **ord = HeatValve**. Every PX file can have multiple ORD parameters.

Note, there is a special parameter case: (ord). It can be used as described above, but also serves as a placeholder for an ORD, which is added when the chart is added by drag & dropping points into PX file and selecting charts in palette via Make Widget menu. The (ord) parameter will be replaced with an absolute ord of the selected component.

#### 2.8.3 Attribute Parameters

Hyperlink parameters can also be used to set chart attribute values in layout, config and trace object. For example **layout.template** = **blue** parameter changes the name of template in all widgets on a linked PX screen, so all charts on the screen will be displayed with **blue** template, independently of the template set in **DavService** / **Default Template** property.

# 2.9 Machine Learning

Baudrate library includes multiple machine learning (ML) models to produce advanced analytical representations: trendlines and forecasts. They are fully calculated in browser, so no configuration shall be done on backend (station). All ML models are embedded into widgets using **postprocessing** attribute in **trace** object. This attribute can contain one or multiple lines in JavaScript to fit the model and calculate points to visualize it.

#### 2.9.1 Trendlines

Trendlines are useful tool to calculate analytical dependency between variables.

A good example is how energy consumption in a building depends on degree days – a metric of how much (degrees) and for how long (days) the outside temperature sits below a certain level. To prepare data for scatter chart with trendline, one can use Series Transform component to merge energy and degree days histories and bind them to scatter chart attributes, so  $\mathbf{x}$  axis represents degree days and  $\mathbf{y}$  axis represent energy.

Then add another scatter trace and bind it to the same SeriesTransform component. Add **postprocessing** attribute to the **trace** and enter [%trendX%, %trendY%, %trendF%] = Trendline.fit\_predict(%degreedays%, %energy%, 'poly', 2). This function calculates polynomial trendline of 2nd degree using degreedays column as independent variable and energy column as dependent variable. The function returns three sets of values: trendline X coordinates, trendline Y coordinates and trendline formula. In this example these values are saved in %trendX%, %trendY%, and %trendF% variables in the history, so now one can draw trendline by binding %trendX% to x attribute, and %trendY% to y attribute using Format field.

Types of trendlines:

- linear linear curve Y = a \* X + b
- poly polynomial curve  $Y = a1 * X + a2 * X^2 + a3 * X^3 + \ldots + b$
- power power curve  $Y = a * X^b$
- exponent exponential  $Y = a * e^{(X * b)}$

#### 2.9.2 Forecasts

Forecasts are used to predict future values of time series, i.e. timestamped historical data. The model used for forecasting works well with periodical data: outdoor temperature for the next week, energy consumption for the next year, or outdoor light level for the next day.

The model predicts one future period and requires at least two full periods of data and the number of data samples per period. The more periods of data are supplied, the better will be the forecast. The number of data samples should be calculated as the number of historical values for one repetitive cycle.

For example, if we collect 15 minute history for energy consumption, and we need to forecast consumption for tomorrow, then one cycle is 24 hours and the number of samples is 24  $^{*}$  60 / 15

🏰 Propertie	5						$\times$
BoundLabel					o.	AZ↓	∎↓
Bound Lab	el						
background	(default)						-
blink	false						-
border	0.0 none black						
enabled	true						-
font	(default)						
	5000						
4 hyperlini	c						×
station: s	slot:/FCU					- 1	(De
layout.templa	te=blue;ord=FCU1	» · •					
	-		- I				
mouseOver	None	ОК	Cancel				
padding	0.0	🐴 Config	ure View Q	uery			×
	FCU01	Key		Value			Ę
text	4.00	layout.te	emplate	blue	-		
textlconGap		ord		FCU1			_
textToIconAlig							
valign	Center						
visible	true						
wordWrapEna	false						
▲ Bound Lab	el Binding	1		(+	×	Ŀ	-
ord	null	]	0	K Cai	ncel		
degradeBehav	None	1					
hyperlink	station: slot:/FCU viev	v:?layout.tem	plate=blue	ord=FCU1			
summary	%displayName?typeD	isplayName%	6 = 96.96				
popupEnable	true						-
statusEffect	None						-
		ОК	Cancel				

Figure 15: Hyperlink parameters

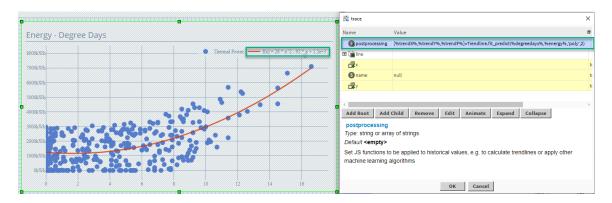


Figure 16: Trendline

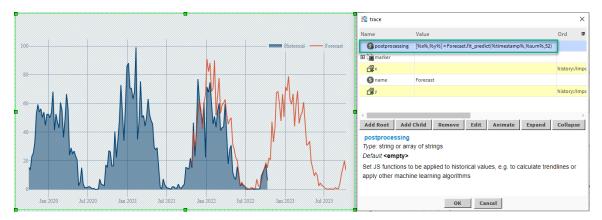


Figure 17: Forecast

= 96. In this case the model will match daily fluctuation of energy. If we need to predict yearly fluctuations, then we can use rollup function to reduce the amount of data to one per day (or one per week, one per month) and the number of samples will be 365 (days in a year). Note, the model requires at least two full cycles, so there shall be at least 365 \* 2 = 730 samples available.

The chart configuration is done very similarly to trendlines, except the data shall be timestamped. The **postprocessing** attribute value should be equal to [%x%, %y%] = Forecast.fit\_predict(%timestamp%, %avg%, 52), where timestamp is history timestamp column, avg is history data column, 52 – number of samples per period, x and y – variable names for forecasted values. These values can be displayed on the chart by binding to x and y attribute using Format field.

#### 2.9.3 DAV Copilot

The DAV Copilot is an AI-based assistant that helps generate charts from textual descriptions. It's an efficient tool for converting your ideas into visual representations with just a few keystrokes. To begin using the DAV Copilot, you'll need to subscribe to an OpenAI paid plan and obtain an OpenAI API key. Follow these steps to set up your DAV Copilot:

- 1. Sign up for OpenAI's paid plan: Visit OpenAI's subscription page and follow the prompts to complete your subscription.
- 2. **Obtain your OpenAI API Key**: After signing up, generate your API key. The **OpenAI Dashboard** will guide you through this process. Remember to keep this key confidential and secure.

42 Options		×
X General	DAV Options	
🌲 Alarm Console	🚯 DAV Options	
🌲 Alarm Portal	Api KeymyApiKey	
FIPS Options	Temperature 1.0 [0.0-2.0]	
🖸 Text Editor	( remperature 110 (in 10)	
🔂 DAV Options		
A Lexicon Editor		
Distributed Sche		
Security Dist Sch		
Q Wb Fox Analyzer		
Bajadoc		
J CI Options		
KNX Import Serv		
C Station Copier O		
TT Platform Conner		
Code Signing Op		
Px Editor		
Template Optior		
Wire Sheet		
$\leftarrow$		
	OK Cancel	

Figure 18: DAV Copilot

3. Input your API Key in the DAV Copilot: After obtaining your API key, open your Workbench application. Navigate to the top menu and select Tools > Options > DAV Options. Enter your OpenAI API key in the 'API Key' field.

Your DAV Copilot setup is now complete!

Here's a simple guide to get you started:

- 1. Begin by opening your PX file.
- 2. In the text input area, describe the widget you'd like to create. For instance, type "Create a pie chart for weekly energy consumption," and then press Enter.
- 3. The DAV Copilot will generate the corresponding chart using its AI capabilities and add it to your PX file automatically.

If you need to modify an existing chart, follow these steps:

- 1. Select the chart you want to modify.
- 2. Enter your modification instructions in the DAV Copilot menu.
- 3. The DAV Copilot will update the chart based on your text description.

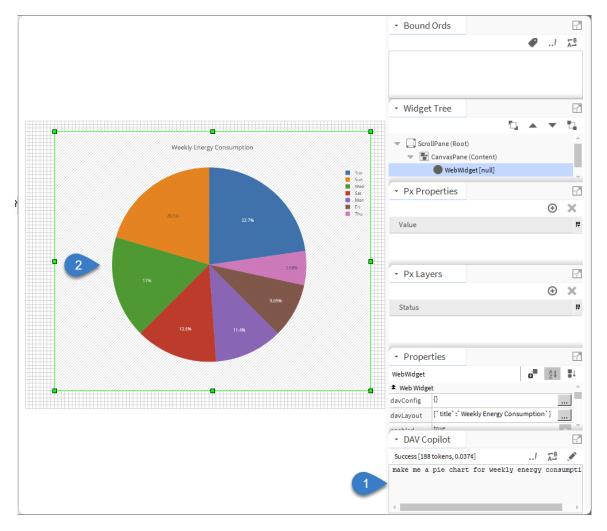


Figure 19: DAV Copilot



Figure 20: Plotly Chart Studio

# 2.10 Additional resources

As the library is based on a popular plotly.js charts, there is a lot of online resources available to answer questions and get inspiration of how to build good-looking charts. Here we describe some of them.

#### 2.10.1 Plotly.com

Plotly.com website provides tons of examples and in-depth descriptions of various plotly.js aspects. Almost everything shown there could be implemented in Baudrate library.

#### 2.10.2 Plotly Chart Studio

Plotly Chart Studio is an online repository of thousands of plotly.js charts, searchable by keywords, chart type, popularity. Every chart can be viewed and edited in online chart editor (see its user guide), the great tool to see how various attribute values affect the chart.

Every chart in Plotly Chart Studio can be exported into file (press **Export** button and select **json** format) and then imported into Niagara (select **Import JSON** in the DAV Copilot toolbar and find the exported file). The chart appears in Niagara exactly as seen online, the only thing left is to bind real-time or historical data to it.

#### 2.10.3 Plotly.js JSON Editor

Another Plotly.js JSON Editor also allows to see and edit multiple chart examples.

#### 2.10.4 Search Engines

Due to plotly.js popularity it is often very easy to find an answer to chart attribute related questions online. For example, googling "plotly.js scatter how to change grid color" returns multiple relevant

<ul> <li>Structure</li> </ul>	~ =	Ŷ		Trace 0,	Trace 0, y; Trace 1, y; Trace 2, y; Trace 3, y; Trace 4, y	Trace 1, «	Trace 2,*	Trace 3, #	Trace 4, x 💡	G
	Type	E Bar	1 v	0.29	Sri Lanka	0.52	0.05	0.06	0.079999999999999985	
Traces			2 🔻	0.17	Italy	0.28	0.33	0.15	0.06999999999999999	
Subplots	x	Trace 0, x 🗙 🗙 🗙	× З т	0.21	South Africa	0.32	0.16	0.18	0.129999999999999999	
			4 🔻	0.07	France	0.17	0.3	0.3	0.159999999999999999	
Transforms	Y	Trace 0, y; Trace 1, y; Trace 2, y; Trace 3, y; Trace	5 -	0.11	England	0.25	0.25	0.28	0.1099999999999999999	
Theme		4, y								
Style										
-	Orientation	Vertical Horizontal								
Style Annotate	Orientation Axes To Use				Agree strongly				Disagree strongly	,
Annotate	Axes To Use			inland Japan	Agree strongly				Disagree strongly	1
Annotate Analyze			- F	Finland Japan Jorway	Agree strongly				Disagree strongly	ſ
Annotate Analyze	Axes To Use X Axis		+ N Ge S	Finland Japan Iorway Frmany Weden	Agree strongly				Disagree strongly	ſ
Annotate Analyze Export	Axes To Use		+ N Ge S	Finland Japan Jorway ermany	Agree strongly				Disagree strongly	1
Annotate Analyze Export	Axes To Use X Axis Y Axis	k            x	+ N Ge S	Finland Japan Jorway ermany weden enmark Israel ustralia	Agree strongly				Disagree strongly	ſ
Annotate Analyze Export JSON	Axes To Use X Axis Y Axis	X:        Y:        e and position your axes in the	+ N Ge S + De	Finland Japan Iorway ermany weden weden Israel Israel Jstralia Brazil	Agree strongly				Disagree strongly	1
Annotate	Axes To Use X Axis Y Axis You can style	X:        Y:        e and position your axes in the	+ N Ge S + De	Finland Japan Jorway ermany weden enmark Israel Israel ustralia Brazil ortugal Ireland	Agree strongly				Disagree strongly	1
Annotate Analyze Export JSON	Axes To Use X Axis Y Axis You can style	X:        Y:        4       a and position your axes in the net.	+ N Ge S S A A E E	Finland Japan Jorway ermany weden enmark Israel Jsrael Jstralia Brazil ortugal	Agree strongly				Disagree strongly	1

Figure 21: Plotly Chart Editor

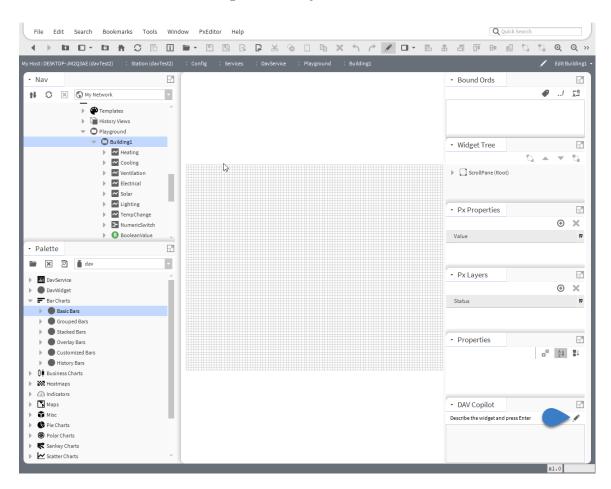


Figure 22: JSON import

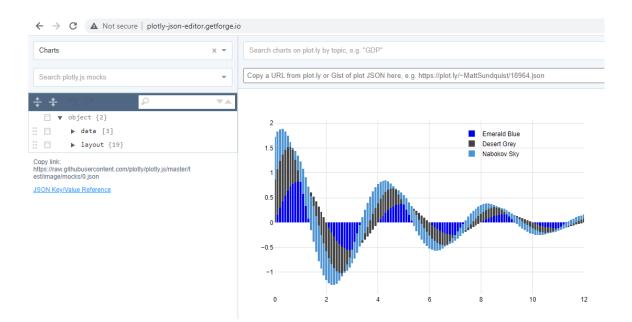
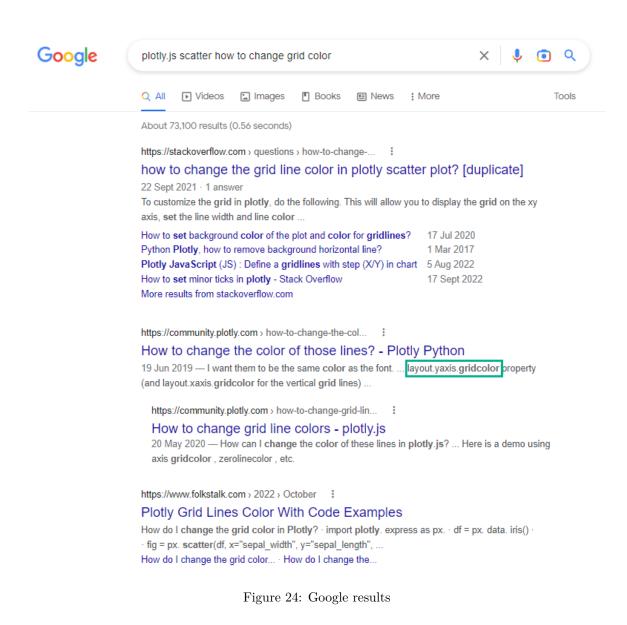


Figure 23: Plotly alternative editor

links to community.plotly.com and stackoverflow.com, providing layout.yaxis.gridcolor attribute as the right answer. It can be modified under **davLayout** property.

#### 2.10.5 Single-Page Reference

Plotly.js Single-Page Reference is a list describing all plotly.js attributes for every type of chart.



# 3 Step-by-step Guide

# 3.1 Bar chart animated with numeric points

In this example we will build a simple bar chart and animate it using numeric points.

- 1. Drag & drop **EmptyWidget** from the DAV palette to the px file.
- 2. Click on Add Binding button and select Dav Binding in the dialog box. Click OK.

😤 Propertie	5	×	🐕 Add Binding 🛛 🗙
WebWidget	1 of 24	∎L.	workbench:Wb View Binding
* Web Widge	t	-	bajaui:Value Binding
davConfig	0	2	dav:Dav Binding
davLavout	0		vykonPro:View Name Binding
	true		vykonPro:User Name Binding
enabled		× 1	vykonPro:Tool Tip Binding
js	view:dav:DavWidget		vykonPro:Enhanced Popup Binding
layer		-	vykonPro:Enhanced Hyperlink Binding
layout	70.0,20.0,500.0,500.0		vykonPro:Enhanced Value Binding
			kitPx:Spectrum Setpoint Binding
preferredSize	,		kitPx:Spectrum Binding
visible	true	-	kitPx:Set Point Binding
			kitPx:Mouse Over Binding
	OK Cancel		kitPx:Popup Binding
			kitPx:Action Binding
			bcsv3:Bcp Command Binding
			analytics:Analytic Web Rollup Binding
			analytics:Analytic Web Chart Binding
			analytics:Analytic Rollup Binding
			analytics:Analytic Value Binding
			analytics:Analytic Table Binding

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

🐴 Propertie	s	×
WebWidget	o Ži	∎↓
✿ Web Widge	t	
davConfig	0	
davLayout	0	
enabled	true	-
js	view:dav:DavWidget	
layer		-
layout	70.0,20.0,500.0,500.0	
preferredSize	500.0,500.0	
visible	true	•
▲ Dav Bindin	g	×
chart	Bar	-
trace	{`_path`:`traces.bar.attributes`}	
	OK Cancel	

4. Press on **trace** property to open a dialog box. This property might contain a very complex tree-like set of attributes.

🏰 Propertie	S	×
WebWidget	o <sup>™</sup> Ž↓	∎↓
✿ Web Widge	t	
davConfig	0	
davLayout	0	
enabled	true	-
js	view:dav:DavWidget	
layer		-
layout	70.0,20.0,500.0,500.0	
preferredSize	500.0,500.0	
visible	true	-
▲ Dav Bindin	g	×
chart	Bar	-
trace	{`_path`:`traces.bar.attributes`}	
	-	
	OK Cancel	

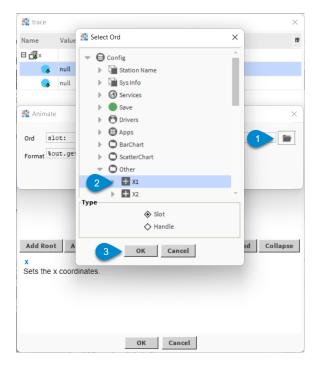
5. Click Add Roots and add  $\mathbf{x}$  axis attribute.

🏰 trace	$\times$
Name Value	9
Mew New	×
Property to Add alignmentgroup	1
uid	
OK uirevision unselected	
visible	
width	
2 ×	
x0	
xaxis	
xcalendar	
xperiod	
xperiod0 xperiodalignment	
y	
Add Root 1 hild Remove y0	Expand Collapse
X vovie	
Sets the x coordinates.	
OK Cancel	

- 6. Click Add Child to add a number of points you are looking to add to the chart.
- 7. While child attribute is selected click **Animate** button. From drop-down menu select and click **Component Chooser**.

Image: Second								
Image: Second Secon	lame	Value	Ord					t
Component Chooser Format Bout, value B OK Cancel Bql Query Builder	×							
Animate Ord alot: Component Chooser Hotory Chooser Hotory Chooser Transform Chooser Bqt Query Builder	•	null						
Ord alot: Format Boot.value8 OK Cancel Bal Query Builder	•	null						
	Ord	alot:	ue%	ОК	•	2 - 3 Component C History Choos	ser	
	Add Roo	t Ad	d Child R	emove	Expand (	Bql Query Bul		
X Tuno: data, accav	x			emove	Expand (	Bql Query Bul		
X Type: data_array Sets the x coordinates.	x Type: da	ta_array		emove	Expand	Bql Query Bul		
Type: data_array	x Type: da	ta_array		emove	Expand	Bql Query Bul		
Type: data_array	x Type: da	ta_array		emove	Expand	Bql Query Bul		
Type: data_array	x Type: da	ta_array		emove	Expand	Bql Query Bul		

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



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

🏰 trace		×	
Name	Value	Ord	,
🗆 과			
•	null		
•	null		
👫 Animate		×	
Ord st	ation:	slot:/Other/X1	
Format			
Format	roprayi		
		2 OK Cancel	
			ſ
Add Root	Ad	d Child Remove Edit Animate Expand Collapse	ı
X	Au	a cinta Remove Lait Annuale Expana Cottapse	1
Sets the x	coordi	nates.	
		OK Cancel	

- 10. Click **Add Roots** and add  $\mathbf{y}$  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 point in Niagara station. Click OK.
- 14. Enter %out.value% 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.

- 16. Click **OK** to close **trace** property dialog box.
- 17. Click **davLayout** to open the dialog box.

4 Propertie	s	$\times$
WebWidget	o <sup>™</sup> ≜1	∎↓
Web Widge	t	
davConfig	8	
davLayout	0	
enabled	true	-
js	view:dav:DavWidget	
layer		-
layout	70.0,20.0,500.0,500.0	
preferredSize	500.0,500.0	
visible	true	-
★ Dav Bindin	g	×
chart	Bar	-
trace	{`_path`:`traces.bar.attributes`}	
	OK Cancel	

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

🚰 davLayout	×
Name Value Ord	Ę
👫 New X	
Property to Add title	
OK Cancel	
Add Roothild Remove Edit Animate Expand Coll	apse
Type: object containing one or more of the keys listed below	
OK Cancel	

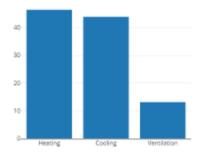
18. Select **title** attribute and click **Add Child** and add **text** attribute.

Name Value						₽.
📷 title						
		Property to Add	d font font pad text x xanch xref y yanch yref			
Add Root	Add Child	<b>1</b> e	Edit	Animate	Expand	Collapse
	ed to be defi	that before the				
		ОК	Car	ncel		

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

Name	Value	t₽
🗆 🍺 title		
🔇 t	text null	
	🐔 Edit 🛛 🕹	
	Property Value Energy Chart	
	ОК Сапсеі	
Add Roo	Add Child Remove Edit Animate Expand Coll	apse
contents	plot's title. Note that before the existence of <code>title.text</code> , the title's used to be defined as the <code>title</code> attribute itself. This behavior has precated.	;
	OK Cancel	

- 20. Click  $\mathbf{OK}$  to close  $\mathbf{davLayout}$  property dialog box.
- 21. Press  $\mathbf{OK}$  to close the widget and see the result.



# 3.2 Scatter chart animated with numeric history

In this example we will build a simple scatter chart.

- 1. Drag and drop **EmptyWidget** from the **dav** palette to the px file.
- 2. Click on Add Binding button and select Dav 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.

Name Value		ţ
	🐒 New X	
	Property to Add cliponaxis	
	stream	
	OK t	
	text	
	textfont	
	textposition	
	texttemplate transforms	
	uid	
	uirevision	
	unselected	
Add Root	Id Remove Visible Expand C	ollapse
x	x0	
Sets the x coordinates	s. <sub>xaxis</sub>	
	veolop.dar	
	OK Cancel	

6. While **x** attribute is selected click **Animate** button. Enter %timestamp% in Format field to display history timestamps on x axis. From drop-down menu select and click **History Chooser**.

🌆 trac	ie.		×
Name	Value	Ord	æ
đ×			
Add I X Type:	teot	Add Child Remove 1 Animate Expand Collapse	
		OK Cancel	

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

🏰 trace		×
Name	Value Ord	ŧ
<b>a</b> ⊀		
😤 An	nimate	×
Ord	History ORD	
	ОК Сапсеі	
Add Roo X Sets the	oot Add Child Remove Edit Animate Expand	Collapse
	OK Cancel	

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

History ORD	history: 🖀 -	
Time Range	Time Range	
Rollup Interval	? to ? Disabled	
Rollup Function	»	
Advanced Prope	ties ¥	
	OK Cancel	

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.

History ORD	history:	<b>·</b>
Time Range	Time Range	
Rollup Interval	Daily	
Rollup Function		>
Advanced Proper	rties ¥	
	OK Cancel	

10. If **Rollup Interval** is used click on **Rollup Function** and select the necessary function. For this example let's select **avg**. Click **OK**. Click **OK** to exit a dialog box.

4 ord	X
History ORI	Chooser X
	vavg 2
Time Range	count
	first
Rollup Inter	max
Rollup Fund	min 🔪 🔪 🚺
Advanced F	last
	sum
	OK Cancel
	OK Cancel

- 11. Click Add Root and add y axis attribute.
- 12. While **y** attribute is selected click **Animate** button to open the dialog box. Enter **%avg%** in **Format** field (if you have not used rollup function enter "%value%") to display history values on **y** axis. Repeats steps 7-8.
- 13. Click **Add Root** and add **name** attribute.
- 14. Double-click on name attribute and enter a series name e.g. "Building 1".

Name	Value	ţ,
<b>a</b> ×		
Ду		
🔇 name	null 1	
	-	
ſ		
	🚰 Edit	×
	Prope 2 Building 1	
	OK Cancel	
Add Root	t Add Child Remove Edit Animate	Expand Collapse
name		
Sets the t	trace name. The trace name appear as the legend	item and on hover.
	OK Cancel	

- 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 select copy for the existing **trace** properties, then select paste for a newly created **trace** property.

😤 Propertie	\$	🗙 🕺 Properti	es X
WebWidget	o" 21 5	WebWidget	o <sup>#</sup> 21 ≣↓
* Web Widge	t	* Web Widg	et
davConfig	{`displaylogo`:false,`showEditinChartStudio`:true,`modeBarButtonsToAdc	davCon 3	{`displaylogo`:false,`showEditInChartStudio`:true,`modeBarDuttonsToAdc
davLayout	("title":("text":"Basic Sankey"),"font":("size":10))	. davLayout	(`title`:(`text`:`Basic Sankey`),`font`:(`size`:10))
enabled	true	enabled	true 🗸
js	view:dav:DavWidget		view:dav:DavWidget
layer		layer	
layout	0.0,120.0,470.0,620.0	layout	0.0,120.0,470.0,620.0
preferredSize	500.0,500.0	preferred Size	
visible	true	visible	true 🗸
* Day Bindin	l.	× ± Dav Bindi	ng X
chart	Sankey	chart	Sankey
	("node"d"nad":15,"color":["blue","blue","blue","blue","blue","blue","blue",	. trace	('node's['pad':15,'color's['blue','blue','blue','blue','blue','blue','blue'],
± Davi		× ± Dav Bindi	ng X
chart Uni	ink	chart	Sankey
trace Cop		trace	(* gath'; traces.sankey.attributes')
_	OK Cancel	Unlin	k
		Сору	
		Paste	Cancel

- 19. Click **davLayout** to open the dialog box.
- 20. Click **Add Root** and add **title** attribute.

Name Value	51
Mew 1	×
Property to Add	title
	polar
	radialaxis
	scene
	selectdirection
	selectionrevision
	separators
	shapes
	showlegend
	sliders
Add Root hild Remov	spikedistance Expand Collapse
title	sunburstcolorway
Add a new child to the object or se	template its description
	ternary
2	title
-	transition
	OK Cancel

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

Name Value					ţ
title					
	Mew		×		
	Property to Add	font			
	ОК	pad			
	2	text x			
		xanchor			
		xref			
		У			
		yanchor yref			
Add Root Add Chil	d ve		nimate	Expand	Collapse
text Sets the plot's title. No contents used to be de been deprecated.	te that before the	existence le attrib	of titl ute itself.	e.text , th This behav	e title's ior has
	ОК	Cancel	L		

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

Name	Value					5
🗆 🍓 title						
🔇 tex	t null < 1					
	🐴 Edit					×
	Prope 2	Building 1	Annual B	Energy Prof:	ile	
		ОК	Ca	ncel		
Add Root	Add Child	Remove	Edit	Animate	Expand	Collapse
	lot's title. Note t sed to be defin ecated.					
		ОК	Car	ncel		

- 23. Click **OK** to close **davLayout** property dialog box.
- 24. Press  $\mathbf{OK}$  to close the widget and see the result.



# 4 Attribute Reference

# 4.1 Trace Attributes

The **trace** objects attributes control aspects of a chart itself: the colors, the grids, the data etc. The full list of attributes can be found Trace Reference online.

# 4.2 Layout Attributes

The **davLayout** object contains attributes that control positioning and configuration of non-datarelated 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 DAV library functions when interacted with by a user: updatemenus and sliders

#### 4.2.1 Frequently Used Layout Attributes

- title.text figure title.
- title.font.size figure title size.
- title.font.color figure title color.
- showlegend enable/disable 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 in Layout Reference online.

# 4.3 Config Attributes

The **config** object's attributes control behaviours, which are not considered part of the figure itself, e.g. the behaviour of the **modebar**, or time range selector buttons, or how the figure reacts to mouse actions like scrolling.

Descriptions of config attributes can be found in Configuration Options online.

4 Attribute Reference

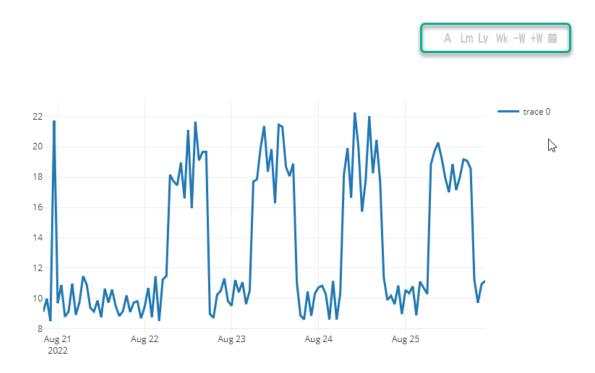


Figure 25: Time range buttons

#### 4.3.1 Time Range Selector Buttons

Chart time range can be changed dynamically via buttons in the mode bar.

To display time range buttons add any combination of the following values as children to config.modeBarButtonsToAdd or as grandchildren to config.modeBarButtons:

- histAuto, histToday, histYesterday, histWeekToDate, histLastWeek, histMonthToDate, histLastMonth, histYearToDate, histLastYear standard Niagara time selectors
- prevDay, prevWeek, prevMonth, prevYear select previous day / week / month / year relative to the displayed period.
- nextDay, nextWeek, nextMonth, nextYear select next day / week / month / year relative to the displayed period.

#### 4.3.2 Frequently Used Config Attributes

- 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.