INTERFACE DEFINITION DOCUMENT

NETRONIC HTML5
Visual Scheduling Widget - Standard Edition
(VSW SE)

Version: **1.0** | as of 11 April 2019

NETRONIC Software GmbH
Pascalstr. 15
52076 Aachen
Germany
Tel: +49 (2408) 141 0
Fax: +49 (2408) 141 33
Web: www.netronic.com

Contents

1   Changelog

Version	Description of changes
1.0	Initial release

2   Enumerations

The following enumerations are provided:

2.1   ActivityBarDragModes

```
NETRONIC.VSWidget.ActivityBarDragModes = {

    // Note: Values are flags,
    //       i.e. they can be combined by using bitwise OR operators.

    None: 0,
    DragStart: 1,
    DragEnd: 2,
    DragHorizontally: 4,
    DragVertically: 8,
    DragAutoHorOrVer: 16,
};
```

2.2   ActivityBarShape

```
NETRONIC.VSWidget.ActivityBarShape = {
    Regular: 0,
    Summary: 1,
    Diamond: 2,
};
```

2.3   AllocationBarDragModes

```
NETRONIC.VSWidget.AllocationBarDragModes = {

    // Note: Values are flags,
    //       i.e. they can be combined by using bitwise OR operators.

    None: 0,
    DragStart: 1,
    DragEnd: 2,
    DragHorizontally: 4,
    DragVertically: 8,
    DragAutoHorOrVer: 16,
};
```

2.4   CollapseState

```
NETRONIC.VSWidget.CollapseState = {
    Unchanged: -1,
    Expanded: 0,
    Collapsed: 1
```

```
};
```

2.5   ObjectType

```
NETRONIC.VSWidget.ObjectType = {
    TimeArea: -2,
    Timescale: -1,
    Activity: 1,
    Allocation: 2,
    Resource: 5,
    Link: 6,
    Curve: 7,
    Entity: 13
};
```

2.6   ViewType

```
NETRONIC.VSWidget.ViewType = {
    Activities: 0,
    Resources: 1
};
```

3   Data Model

The data model of the NETRONIC HTML5 Visual Scheduling Widget, Standard Edition (VSW SE) is designed for resource planning in general, but is extended to cover activities view and resources view.

The model is extensible on every object. When created by JavaScript code, the objects do not require a special constructor, so they can be created easily with or without using the new keyword.

A note regarding the dates in attributes:
Browsers did not handle date strings consistently in the past. So it is recommended to use the simplified ISO 8601 standard see http://www.ecma-international.org/ecma-262/5.1/#sec-15.9.1.15 for defining unambiguously: Examples: 2019-05-03T**08**:13:28**Z** (UTC) or 2019-05-03T**10**:13:28**+02:00** (MEST) for the same time point. Using date objects in the object is recommended, since then the creation can be done on several ways and internally the dates can be used immediately without conversion.

3.1   Resource

A Resource object defines the properties of a single resource.

Property Name	Type	Description
ID	string	**Required** – Identifier of the resource
Name	string	**Optional, default: undefined** – Name of the resource (freely usable), text here is displayed in the table row
CalendarID	string	**Optional, default: undefined** – Corresponding calendar. If undefined, then the calendar specified by the option defaultCalendarID will be used.

CapacityCurveID	string	**Optional, default: undefined** – Identifier of any curve representing the capacity of this resource. If the identifier references a curve stack, then the summed curve is shown with the color settings of the curve stack.
LoadCurveID	string	**Optional, default: undefined** – Identifier of any curve representing the load of this resource. If the identifier references a curve stack, then all curves within the curve stack are shown with their individual color settings as a stack.
ParentID	string	**Optional, default: undefined** – Identifier of a parent resource this resource is assigned to. If this property is defined, the parent resource will become a resource group (if not yet a resource group) and it will keep its role as a resource with a capacity of its own. If this property is undefined the current resource will be considered as a root node of the resource hierarchy.
PM_CollapseState	number	**Optional, default: -1** – Specifies whether the row of the resource should be expanded or collapsed when displayed the very first time. -1: do not change the way the resource row is displayed 0: display resource row in an expanded way 1: display resource row in a collapsed way See enum CollapseState in the Enumerations chapter for details.
PM_TableColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color for the table row. If undefined, a default value of the widget will be used.
PM_TableTextColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color for the table row texts. If undefined, a default value of the widget will be used.
PM_RowSymbolIDs	string[]	**Optional, default: undefined** – Array of identifiers of the symbols to be shown in the table symbol cell of the beginning of the table row. The symbols will be arranged one below the other. However, if the cell is not high enough to hold all symbols, then the remaining symbols are also arranged side-by-side. If this still does not fit, an additional "show more" symbol will be displayed. An empty string ("") will cause an "empty" symbol to be displayed. By this placeholder, you can reserve space for a symbol that may be shown at a later time. Please note:

		Each symbol will be resized to an image with a width and height of 16 pixels each at a zoom level of 100%.

3.2   Calendars

3.2.1   Calendar

A Calendar object defines working and non-working times to be used with resources.

Property Name	Type	Description
ID	string	**Required** – Identifier of the calendar
Entries	CalendarEntry[]	**Optional, default: undefined** – Array of calendar entry objects. The order of the entries inside the array is important!

3.2.2   CalendarEntry

A CalendarEntry object defines a single time period. It has to be referenced in the Entries array of a Calendar object. If several calendar entries describe the same time period, then the last entry wins.

Property Name	Type	Description
Start	Date|string	**Optional, default: undefined** – Start of the working time period. If data type is *String*, then the value should be formatted this way: "YYYY-MM-DDThh:mm:ssZ" (this implies that the date is specified in UTC). Since the browsers do not interpret every formatted date string in a standardized way, one has to be careful about it.
End	Date|string	**Optional, default: undefined** – End of the working time period. If data type is *String*, then the value should be formatted this way: "YYYY-MM-DDThh:mm:ssZ" (this implies that the date is specified in UTC). Since the browsers do not interpret every formatted date string in a standardized way, one has to be careful about it.
TimeType	number	**Optional, default: 1** 0: NonworkingTime, 1: WorkingTime

3.3   Activity

An Activity object defines the properties of a single activity.

Property Name	Type	Description
ID	string	**Required** – Identifier of the activity.

Name	string	**Optional, default: undefined** – Name of the activity (freely usable), text here is displayed in the table row
ParentID	string	**Optional, default: undefined** – Identifier of the parent of the activity. This serves for setting up a hierarchy of activities. If this property is undefined the current activity will be considered as a root node of the activity hierarchy.
CalendarID	string	**Optional, default: undefined** – Corresponding calendar. If undefined, then the calendar specified by the option defaultCalendarID will be used. See also option pm_activityCalendarsEnabled
Start	Date|string	**Optional, default: undefined** – Start date of the activity. If data type is *String*, then the value should be formatted this way: "YYYY-MM-DDThh:mm:ssZ" (this implies that the date is specified in UTC). Since the browsers do not interpret every formatted date string in a standardized way, one has to be careful about it.
End	Date|string	**Optional, default: undefined** – End date of the activity. If data type is *String*, then the value should be formatted this way: "YYYY-MM-DDThh:mm:ssZ" (this implies that the date is specified in UTC). Since the browsers do not interpret every formatted date string in a standardized way, one has to be careful about it.
Progress	number (floating point; in percent; $\geq 0, \leq 100$)	**Optional, default: 0.0** – Used to display a completion layer.
Editable	boolean	**Optional, default: true** If set to false, then neither this activity nor any allocation in which this activity is involved can be changed by user interactions.

PM_AllowedBarDragModes	number (see enum [ActivityBarDragModes](#))	**Optional, default: value of option pm_defaultAllowedActivityBarDragModes** – This option determines the allowed bar drag modes for this activity in the activities view (these can be overwritten using the callback canDrag).
PM_Color	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** Color for the working time periods of the bar. The nonworking time periods of the bar will be colored with the same color as long as the property PM_NonworkingTimeColor is undefined or set to "calculated". If undefined, a default value of the widget will be used.
PM_NonworkingTimeColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red" or "calculated")	**Optional, default: undefined** Color for the nonworking time periods of the bar. If undefined, a default value of the widget will be used. If set to "calculated", a color will be calculated using the color defined by the PM_Color property.
PM_BorderColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red" or "calculated")	**Optional, default: undefined** Color for the border of the bar. If undefined, a default value of the widget will be used. If set to "calculated", a color will be calculated using the color defined by the PM_Color property. This can be useful in situations where two bars are positioned next to each other and a graphical indicator is needed to visually distinguish the two bars.
PM_TextColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** Color for the texts of the bar. If undefined, a default value of the widget will be used.
PM_ProgressColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** Color for the working time periods of the progress bar. The nonworking time periods of the bar will be colored with the same color as long as the property PM_ProgressNonworkingTimeCol

		or is undefined or set to "calculated". If undefined, a default value of the widget will be used.
PM_ProgressNonworkingTimeColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red" or "calculated")	**Optional, default: undefined** Color for the nonworking time periods of the progress bar. If undefined, a default value of the widget will be used. If set to "calculated", a color will be calculated using the color defined by the PM_ProgressColor property.
PM_TableColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** Color for the table row. If undefined, a default value of the widget will be used.
PM_TableTextColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color for the table row texts. If undefined, a default value of the widget will be used.
PM_CollapseState	number	**Optional, default: -1** – Specifies whether the row of the activity should be expanded or collapsed when displayed the very first time. -1: do not change the way the activity row is displayed 0: display activity row in an expanded way 1: display activity row in a collapsed way See enum CollapseState in the Enumerations chapter for details.
PM_TopLeftBarSymbolID	string	**Optional, default: undefined** – Identifier of the symbol to be shown at the top left side of the activity bar. Please note: A symbol will be resized to an image with a width and height of 12 pixels each at a zoom level of 100%.
PM_TopRightBarSymbolID	string	**Optional, default: undefined** – Identifier of the symbol to be shown at the top right side of the activity bar. Please note: A symbol will be resized to an image with a width

		and height of 12 pixels each at a zoom level of 100%.
PM_RowSymbolIDs	string[]	**Optional, default: undefined** – Array of identifiers of the symbols to be shown in the table symbol cell of the beginning of the table row. The symbols will be arranged one below the other. However, if the cell is not high enough to hold all symbols, then the remaining symbols are also arranged side-by-side. If this still does not fit, an additional "show more" symbol will be displayed. An empty string ("") will cause an "empty" symbol to be displayed. By this placeholder, you can reserve space for a symbol that may be shown at a later time. Please note: Each symbol will be resized to an image with a width and height of 16 pixels each at a zoom level of 100%.
PM_BarShape	number (see enum ActivityBarShape)	**Optional, default: value in option pm_defaultActivityBarShape** – This option defines which shape should be used by default for the visualization activity bars.

3.4 Allocations

3.4.1 Allocation
An Allocation object defines an allocation of one activity to one resource.

Property Name	Type	Description
ID	string	**Required** – Identifier of the allocation
ActivityID	string	**Optional, default: undefined** – Identifier of an Activity
ResourceID	string	**Optional, default: undefined** – Identifier of a Resource
Entries	AllocationEntry[]	**Optional, default: undefined** – array of allocation entries.
Progress	number (floating point; in percent; ≥ 0, ≤ 100)	**Optional, default: 0.0** – Used to display a completion layer.

PM_AllowedBarDragMod es	number (see enum [AllocationBarDragMode s](#))	**Optional, default: value of option pm_defaultAllowedAllocationBarDragMo des** – This option determines the allowed bar drag modes for this allocation in the resources view (these can be overwritten using the callback canDrag).
PM_BorderColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red" or "calculated")	**Optional, default: undefined** – Color for the border of the bar. If undefined, the value of the corresponding activity, if available, will be used. If set to "calculated", a color will be calculated using the color defined by the PM_Color property. This can be useful in situations where two bars are positioned next to each other and a graphical indicator is needed to visually distinguish the two bars.
PM_TextColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color for the texts of the bar. If undefined, the value of the corresponding activity, if available, will be used.
PM_TopLeftBarSymbolID	string	**Optional, default: undefined** – Identifier of the symbol to be shown at the top left side of the allocation bar. Please note: A symbol will be resized to an image with a width and height of 12 pixels each at a zoom level of 100%.
PM_TopRightBarSymbolI D	string	**Optional, default: undefined** – Identifier of the symbol to be shown at the top right side of the allocation bar. Please note: A symbol will be resized to an image with a width and height of 12 pixels each at a zoom level of 100%.

3.4.2   AllocationEntry

Property Name	Type	Description
Start	Date|string	**Optional, default: undefined** – Start date of the allocation entry. If data type is *String*, then the value has to be formatted this way: "%d-%m-%yT%H:%M:%SZ" (this implies that the date is specified in UTC).
End	Date|string	**Optional, default: undefined** – End date of the allocation entry. This date itself is not(!) part of the interval described by this entry.

		If data type is *String*, then the value has to be formatted this way: "%d-%m-%yT%H:%M:%SZ" (this implies that the date is specified in UTC).
PM_Color	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color for the working time periods of the bar. If undefined, the value of the corresponding allocation, if available, will be used. When the property Allocation.PM_BarDesign is not set to the value GroupedEntries: The nonworking time periods of the bar will be colored with the same color as long as the property PM_NonworkingTimeColor of the appropriate allocation is undefined or set to "calculated". If undefined, the value of the corresponding activity, if available, will be used.
PM_NonworkingTimeColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red" or "calculated")	**Optional, default: undefined** Color for the nonworking time periods of the bar. If undefined, the value of the corresponding allocation, if available, will be used. If that one is also undefined, then the nonworking time periods of the bar will be colored with the same color as the working times (see PM_Color property). If set to "calculated", a color will be calculated using the color defined by the PM_Color property.

3.5   Curves

3.5.1   Curve

Property Name	Type	Description
ID	string	**Required** – Identifier of the stacked curve
Type	number	**Required** – Type of the curve: 0: PointCurve 3: StackedCurve
CurvePointEntries	CurvePointEntry[]	**Optional, default: undefined** – Array of point entries (in case of PointCurve only)
CurveIDs	string[]	**Optional, default: undefined** – Array of curve IDs (in case of StackedCurve only)
PM_FillColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color of the area below the curve
PM_StrokeColor	string (CSS color value, e.g. "#ff0000",	**Optional, default: undefined** – Color of the curve line itself

	"rgb(255, 0, 0)", or "red")	
PM_OverloadColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Used, when the curve is used as the load curve that referenced directly by the property LoadCurveID at the Resource object. Then the area above the capacity curve will be colored by this color when the load is higher than the capacity.

3.5.2   **CurvePointEntry**

Property Name	Type	Description
PointInTime	Date|string	**Required** – This property serves as an identifier of the point entry. If data type is *String*, then the value has to be formatted this way: "YYYY-MM-DDThh:mm:ssZ" (this implies that the date is specified in UTC).
Value	number (floating point)	**Optional, default: 0.0** – Value of the curve at the given point in time.

3.6   Link

A Link object defines the properties of a single link between activities.

Property Name	Type	Description
ID	string	**Required** – Identifier of this link
SourceActivityID	string	**Required** – Identifier of the source activity
TargetActivityID	string	**Required** – Identifier of the target activity
RelationType	number	**Optional, default: 0** – The relation type is used for drawing: 0: Finish-Start, 1: Finish-Finish, 2: Start-Start
PM_Color	string (CSS color value, e.g. "#ff0000", "rgb(255,0,0)", or "red")	**Optional, default: undefined** – Color for the link. If undefined, a default value will be used.

3.7   Entity

An Entity object defines the properties of a single entity. Entities are shown in a separate table on the right side.

Property Name	Type	Description
ID	string	**Required** – Identifier of this entity
Name	string	**Optional, default: undefined** – Name of the entity (freely usable)
ParentID	string	**Optional, default: undefined** – Description of the entity (freely usable)
PM_CollapseState	number (see enum [CollapseState](#))	**Optional, default: -1** – Specifies whether the row of the entity should be expanded or collapsed when displayed the very first time.
PM_TableColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color for the table row. If undefined, a default value of the widget will be used.
PM_TableTextColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color for the table row texts. If undefined, a default value of the widget will be used.
PM_RowSymbolIDs	string[]	**Optional, default: undefined** – Array of identifiers of the symbols to be shown in the table symbol cell of the beginning of the table row. The symbols will be arranged one below the other. However, if the cell is not high enough to hold all symbols, then the remaining symbols are also arranged side-by-side. If this still does not fit, an additional "show more" symbol will be displayed. An empty string ("") will cause an "empty" symbol to be displayed. By this placeholder, you can reserve space for a symbol that may be shown at a later time. Please note: Each symbol will be resized to an image with a width and height of 16 pixels each at a zoom level of 100%.

3.8 Symbol

A Symbol object defines the properties of a single symbol. Symbols are used by resources, activities, and allocations. They can be displayed at different locations inside the table and the diagram area.

Please note: The symbols will be resized to an image with an appropriate width and height depending on their application. Therefore, when designing the symbols, you should ensure that they are clearly recognizable and visually distinguishable. For more details regarding the size, please see the descriptions of the properties related to symbols.

For some users maybe it is not possible to use paths in the property URL at all, but instead you have the possibility to use 'data URIs', that can be created using an online service (e.g.

https://websemantics.uk/tools/image-to-data-uri-converter/) to convert your SVG file to a string containing the SVG.

Property Name	Type	Description
ID	string	**Required** – Identifier of this symbol
URL	string	**Required** – URL of a SVG image containing the symbol. Two types of URLs are allowed: • absolute URL (e.g. "https://www.aaazzz.com/symbol.svg") • relative URL (e.g. "images/symbol.svg") – In this case, the anchor path for the symbol directory is the application directory. • Data URI (e.g. 'data:image/svg+xml;base64,…'). See https://en.wikipedia.org/wiki/Data_URI_scheme)

4   Widget

This is the central object that an application talks to. Here are methods to add, update and remove the data objects meant above and there also are many options and callbacks to refine the appearance of the widget. Technically the widget is based on the widget factory of jQuery UI. Please see https://learn.jquery.com/jquery-ui/ in order to learn how to work with jQuery and jQuery UI widgets in general.

At first the widget has to be instantiated using a call like $("#ganttDiv").VSW(options), where 'options' is an optional object containing first settings if needed (otherwise it can be left undefined). After that you can set additional options and use the provided methods.

4.1   Options

The following options are settable and gettable by using the jQuery UI Widget command "option" at any time within a session.

Option Name	Type	Description
titleText	string	**Optional, default: undefined** – This text will be shown in the table header, when additionally the flag hasColumnHeaders is not set in the callback onDetermineColumnDefinitions (see there).
start	Date	**Required** – Start of the considered time area.
end	Date	**Required** – End date of the considered time area.
workDate	Date	**Optional, default: undefined** – Date on which the work date line will be displayed.

viewType	number (see enum ViewType)	**Optional, default: ViewType.Resources**
defaultCalendarID	string	**Optional, default: undefined** – Specifies a default calendar to be used in the widget. If calendars are defined on activities or resource they will override this calendar. If there is no calendar defined on an activity or a resource and if this default calendar ID is undefined, then the calendar is assumed to be one with constantly non-working time only.
tableViewWidth	number	**Optional, default: null** – This setting defines the width of the table view.
tableWidth	number	**Optional, default: 400** – This setting defines the width of the table. It is advisable to set this option to a value equal to or greater than the maximum sum of the column widths defined in the column definitions for the Gantt table (see onDetermineColumnDefinitions).
nonWorkingTimeVisible	boolean	**Optional, default: true** – This option defines whether the common non-working time is visible. The common time is calculated by all calendar information that are relevant to the visualization. Therefore, in task mode the calendars of the activities, in resource mode the calendars of the resources are used.
editable	boolean	**Optional, default: true** – If set to false, nothing can be edited.
entitiesTableViewWidth	number	**Optional, default: null** – This setting defines the width of the entities table view.
entitiesTableVisible	boolean	**Optional, default: false** – This option lets appear/disappear the entities table on the right side.
entitiesTableWidth	number	**Optional, default: 200** – This setting defines the width of the entities table. It is advisable to set this option to a value equal to or greater than the maximum sum of the column widths defined in the column definitions for the entities table (see onDetermineColumnDefinitions).
entitiesTitleText	string	**Optional, default: undefined** – This text will be shown in the table header of the entities table.
pm_symbolColumnVisible	boolean	**Optional, default: false** – If set to true, a special column at the left of the table will be displayed to show the row symbols of the activities in the Activities view and of

		the resources in the Resources or Loads view.
pm_symbolColumnWidth	number	**Optional, default: 22** – Width of the symbol column in the Activities, Resources and Loads view. If set to a value less than the default, it will be set to the default automatically.
pm_entitiesTableSymbolColumnVisible	boolean	**Optional, default: false** – If set to true, a special column at the left of the entities table will be displayed to show the row symbols of the entities.
pm_entitiesTableSymbolColumnWidth	number	**Optional, default: 22** – Width of the symbol column in the entities table. If set to a value less than the default, it will be set to the default automatically.

4.2 Callbacks

For simplicity reasons, we have implemented callbacks instead of events. They can be set in the same way as all other "regular" options.

Option Name	Type	Description

compareObjects	Function	**Optional, default: undefined** – This function is called when an object is added or its parent is changed on updating it. The result will determine the sorting of the rows in the view. The comparison always only is made between siblings.
		
		Profile:
		`function (args)`
		`args = {`
		  `"objectType" : ObjectType,`
		  `"objectA" : Object,`
		  `"objectB" : Object,`
		  `"isALowerThanB": Boolean //[in/out]`
		`}`
		
		The function should compare objectA and objectB and write the result into isALowerThanB: true, when A is lower than B and false, when A is greater than B. A cannot be equal to B.

onShowTooltip	Function	**Optional, default: undefined** – This function is called when a tooltip can appear (i.e. when the mouse cursor hovers over an object). The tooltip itself is to be shown by the application, if the property innerHTML is not set on return.
		
		Profile:
		`function (args)`
		`args = {`

		``
		  "objectType" : ObjectType,
		  "object" : Object,
		  "visualType" : VisualType,
		  "event" : DOMEvent,
		  "date"¹ : Date // date at mouse cursor,
		  "capacity"¹ : number,
		  "load"¹ : number,
		  "singleLoads"² : Object,
		  "entry"³ :
		AllocationEntry|CalendarGridEntry,
		  "entryIndex"³ : number,
		  "innerHTML"⁴ : string [out]
		}
		``

onShowContextMenu	Function	**Optional, default: undefined** – This function is called when a context menu can appear. If the function sets a jQuery Promise object (see http://api.jquery.com/promise/) at event.result, then the widget will internally hold the state of a context menu being open until the promise is resolved or rejected. Possible items are resources, activities, allocations, allocation entries, links, timescale and empty time area.
		
		Profile:
		``
		function (args)
		args = {
		  "objectType" : ObjectType,
		  "object" : Object,
		  "visualType" : VisualType,
		  "date" : Date,
		  "event" : DOMEvent or jQuery.Event,
		  "entry"³ :
		AllocationEntry|CalendarGridEntry,
		  "entryIndex"³ : number,
		  "promise" : Promise [out]
		}
		``

onCloseContextMenu	Function	**Optional, default: undefined** – When a context menu is visible in the application and the user starts a new action elsewhere in the widget, the widget sends this event in order to close the open context menu. Profile:

---

¹ Available only if objectType == ObjectType.Resource and the mouse cursor hovers over a curve area.
² Available only if objectType == ObjectType.Resource and the mouse cursor hovers over a curve area. This object has properties where the names are the IDs of the underlying curves of a curve stack and the values represent the current values of these curves at the current date.
³ Available only if objectType == ObjectType.Allocation or if visualType == VisualType.CalendarGrid.
⁴ Text to be displayed inside a tooltip window. This text has to be formatted compliant to the formatting rules for the contents of HTML <div> elements. **Line breaks** can be inserted by adding a <br> tag to the text. Embracing substrings by and tags will show **bold texts**. The same way you can use the <table> and the corresponding <tr> and <td> tags to **tabulate** the tooltip contents.
If your original text contains the symbols "<" or ">" - i.e. those symbols should be displayed as they are and must not be interpreted as parts of HTML tag – then you have to replace the symbols by escape sequence codes (replace "<" by "<" and ">" by ">").

		```function ()```

onDrop	Function	**Optional, default: undefined** – This function is called when an activity/allocation/entity is dropped by the user after dragging it (but only when dropping was allowed by the last triggered onDrag callback). When the function sets a jQuery Promise object into event.result, then the widget disables dragging of the dropped bar until the promise is resolved or rejected. It is also possible to cancel the interaction.
		
		Profile:
		```function (args)```
		```args = {```
		``` "objectType" : ObjectType,```
		``` "object" : Object,```
		``` "visualType" : VisualType,```
		``` "entry"³ : AllocationEntry,```
		``` "entryIndex"³ : number,```
		``` "dragMode" : ActivityBarDragModes```
		```|AllocationBarDragModes,```
		``` "newRowObjectType" : ObjectType,```
		``` "newRowObject" : Object,```
		``` "newStart" : Date,```
		``` "newEnd" : Date,```
		``` "cancel" : boolean [out]```
		```}```
		
		If the promise is resolved, then it is possible to call it with an arguments object, which offers cancel the interaction at last:
		```args = {```
		``` "cancel" : boolean```
		```}```
		
		When using a promise, then the application should ensure that it will be resolved/rejected later in any way, since the drag action lasts active until then. Maybe there should be a timer for time out.

onDoubleClicked	Function	**Optional, default: undefined** – This function is called when an object is double-clicked by the user.
		
		Profile:
		```function (args)```
		```args = {```
		``` "objectType" : ObjectType,```
		``` "object" : Object,```
		``` "visualType" : VisualType,```
		``` "date" : Date,```
		``` "entry"³ :```
		```AllocationEntry|CalendarGridEntry,```
		``` "entryIndex"³ : number```
		```}```

		On time area and timescale, the object is null.

onSelectionChanged	Function	**Optional, default: undefined** – This function is called when the user selects/deselects an object solely or in addition.
		
		Profile:
		``
		function (args)
		args = {
		  "objectType" : ObjectType|0,
		  "selectedObjects" : Object[],
		  "visualType" : VisualType,
		  "object" : Object|null,
		  "event": DOMEvent,
		  "cancel": Boolean /* [in/out],
		                          Default: false */
		}
		``

onCollapseStateChanged	Function	**Optional, default: undefined** – This function is called when a group was expanded or collapsed either in the table of the Gantt diagram or of the entities table. This callback can be triggered:
		• by the user clicking on the appropriate symbol in the group row
		• by automatic row expansion when dragging objects
		• by using the method scrollToObject
		• by setting the attribute PM_CollapseState either on a resource or on an activity object
		
		Profile:
		``
		function (args)
		args = {
		  "objectType" : ObjectType,
		  "object" : Object or null,
		  "newCollapseState" : number,
		  "interactively" : boolean,
		  "promise" : Promise [out]
		}
		``
		
		If the application sets the promise attribute, then the update of the DOM is delayed until the promise is resolved.

onCurveCollapseStateChanged	Function	**Optional, default: undefined** – This function is called when a curves pane was expanded or collapsed table of the Gantt diagram. This callback is triggered by the user clicking on the appropriate symbol in the resource row
		
		Profile:
		``
		function (args)
		args = {
		  "objectType" : ObjectType,
		  "object" : Object,
		  "newCollapseState" : boolean,
		  "promise" : Promise [out]
		}
		``

			If the application sets the promise attribute, then the update of the DOM is delayed until the promise is resolved.

4.3   Methods

The following methods are callable in two ways:

-   $("#ganttDiv").VSW("methodName", param1, param2, …)
-   $("#ganttDiv").VSW("instance").methodName(param1, param2, …)

The first way is the classical one for jQuery UI Widgets. The second way is more object-oriented and faster, when the instance object is hold in its own variable within the application.

Method Name	Result Type	Parameters	Description
addResources	-	resources : Resource[]	Adds resources. [5]
updateResources	-	resources : Resource[], updateMode : UpdateModes	Updates resources. Allowed changes are modification of all attributes besides ID. [5] updateMode is optional. See enum UpdateModes in the Enumerations chapter for details.
removeResources	-	resourcesOrIDs : string[]|Resource[]	Removes resources. [5]
addCalendars	-	calendars : Calendar[]	Adds calendars. [5]
updateCalendars	-	calendars: Calendar[], updateMode : UpdateModes	Updates calendars visually. Allowed changes are modification of all attributes besides ID. [5] updateMode is optional. See enum UpdateModes in the Enumerations chapter of API description for details.
removeCalendars	-	calendarsOrIDs : string[]|Calendar[]	Removes calendars. [5]
addCurves	-	curves : Curve[]	Adds curves. [5]
updateCurves	-	curves : Curve[], updateMode : UpdateModes	Updates curves. Allowed changes are modification of all attributes but ID and Type. [5] updateMode is optional. See enum UpdateModes in the Enumerations chapter for details.
removeCurves	-	curvesOrIDs : string[]|Curve[]	Removes curves. Resources have to be unused to be removable. [5]

---

[5] After changing the data model, the changes will not become visible until the method "render" is called. These calls should be made after all changes are made once. If forgotten, there is a timeout which calls the method "render" automatically, but this eventually leads to flickering within the Widget's visualization.

Method Name	Result Type	Parameters	Description
addActivities	-	activities : Activity[]	Adds activities. [5]
updateActivities	-	activities : Activity[], updateMode : UpdateModes	Update activities. Allowed changes are modification of all attributes besides ID. [5] updateMode is optional. See enum UpdateModes in the Enumerations chapter for details.
removeActivities	-	activitiesOrIDs : string[]|Activity[]	Removes activities. [5]
addAllocations	-	allocations : Allocation[]	Adds allocations. [5]
updateAllocations	-	allocations : Allocation[], updateMode : UpdateModes	Updates allocations. Allowed changes are modification of all attributes besides ID. [5] updateMode is optional. See enum UpdateModes in the Enumerations chapter for details.
removeAllocations	-	allocationsOrIDs : string[]|Allocation[]	Removes allocations. [5]
addLinks	-	links : Link[]	Adds links. [5]
updateLinks	-	links : Link[], updateMode : UpdateModes	Updates links. Allowed changes are modification of all attributes besides ID. [5] updateMode is optional. See enum UpdateModes in the Enumerations chapter for details.
removeLinks	-	linksOrIDs : string[]|Link[]	Removes links. [5]
addEntities	-	entities : Entity[]	Adds entities. [5]
updateEntities	-	entities : Entity[], updateMode : UpdateModes	Update entities. Allowed changes are modification of all attributes besides ID. [5] updateMode is optional. See enum UpdateModes in the Enumerations chapter for details.
removeEntities	-	entitiesOrIDs : string[]|Entity[]	Removes entities. [5]
addSymbols	-	symbols : Symbol []	Adds symbols. [5]
updateSymbols	-	symbols : Symbol[], updateMode : UpdateModes	Updates symbols. Allowed changes are modification of all attributes besides ID. [5] updateMode is optional. See enum UpdateModes in the Enumerations chapter for details.

Method Name	Result Type	Parameters	Description
removeSymbols	-	symbolsOrIDs : string[]|Symbol[]	Removes symbols. [5]
scrollToObject	-	objectType: ObjectType, object: object	Scrolls to the object (activity/allocation/entity/resource). If the object is not visible, the corresponding rows are expanded automatically.
scrollToDate	-	Date	Scrolls to the given date.
selectObjects	-	objectType : ObjectType, objectsOrIDs : string[]|object[], visualType: VisualType	Selects the given objects or the objects addressed by the given IDs. In the activity mode only activities can be selected. In the resource mode only resources and allocations can be selected. The parameter visualType is only required in the activity mode if objects of type Activity are to be selected. In this case you can define whether the activity rows (VisualType.Row) or the activity bars (VisualType.Bar) should be selected. It is possible to select objects that are hidden in the collapsed parent object. The selectionChanged callback (see options) is not called by the widget.
fitTimeAreaIntoView	-	start : Date|undefined, end : Date|undefined	Fits the time area into the visible area. If start and/or end dates are given, then only the time between these are fitted into the visible area. Not given dates are internally replaced by start and end date of the complete time area.
about	-	-	Opens a popup dialog that shows the licenses of all libraries used.