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

Valid for the VSW SE as of version 4.0.0
2021.03.01-10:53

Contents

1 Changelog

Version	Description of changes
4.0.0	MAJOR: To be treated as a bug fix, the property dragMode in the callback onDrop now contains the dragMode of the interaction that took place and not all allowed drag modes on the object! MINOR: New options multipleBarDraggingEnabled, pm_forcedActivityAllowedBarDragModes, pm_forcedAllocationAllowedBarDragModes. New properties coupledObjects and startsAndEndsOfCoupledObjects in callback onDrop. New property selectedObjects in callback canDrag. MINOR: World view implemented. See options worldViewVisible, worldViewPosition, worldViewExtent. MINOR: Improved loading performance. MINOR: New options loggingEnabled and interactiveActivationOfLoggingEnabled. MINOR: New property SymbolIDSource in TableCellDefinition object. MINOR: New property newRowObjectIsSuitableResource for callbackArgs of callback onDrag. MINOR: When dropping a date line interactively, the resulting date is rounded to the best possible date that is represented by the X coordinate the line phantom is shown on. MINOR: In the world view now date lines are drawn additionally. MINOR: New callback "visibilityFilter" triggered for sorting row objects of types Activity, Entity, Resource. MINOR: Additional parameters for method scrollToObject and new options pm_scrollToObjectAnimationEnabled, pm_scrollToObjectHighlightFlashingEnabled, and pm_scrollToObjectHighlightingColor. MINOR: New property HorizontalTitleAlignment in TableCellDefinition object. MINOR: New properties PM_BarTextPrefixSymbolID/Height/Width, PM_Left/RightBarSymbolID, PM_Left/RightBarSymbolWidth, PM_Left/RightBarSymbolHeight for Allocation and Activity objects. MINOR: Support for Polish (pl) and Portuguese (pt = pt-pt; pt-br) locales added. MINOR: New option pm_ignoreCalendarOnAllocationBarInteractions. MINOR: Option pm_commonViewAreaVisible renamed to pm_mainViewAreaVisible. PATCH: Many bug fixes.
3.2.1	PATCH: A click on a curve now triggers the callback onClick again.
3.2.0	MINOR: New options cursorDateLineVisible, pm_timeAreaPanningMode, pm_timescaleInteractionsEnabled, and pm_curvePanesCollapsibleInResourcesView. MINOR: New options currentDate, pm_pastBackgroundFillColor/LineColor/LineWidth/LineDashArray. MINOR: New option timeZone.

Version	Description of changes
	MINOR: New date line properties CaptionOrientation, CaptionPosition, InFrontOfBars, and Draggable. MINOR: New activity and allocation properties PM_Status4Color and PM_Status4Visible. MINOR: New allocation property SuitableResourceIDs and new options pm_suitableResourcesOverlayColor/pm_unsuitableResourcesOverlayColor. MINOR: New object types PeriodHighlighter/PeriodHighlighterEntry and new methods add/update/removePeriodHighlighters. New property PM_PeriodHighlighterID on activity and resource objects. New VisualType property PeriodHighlighter.
3.1.3	PATCH: Texts in first scrollable table column (in left table and in entities table) was clipped too much on the right side. PATCH: In some cases the SVG content was drawn over the horizontal scrollbars. PATCH: It is now allowed to drag bars even when they are drawn inside a visible collapsed row and belong to a hidden row.
3.1.2	PATCH: Updates to calendar and curve objects now updates also the activities view.
3.1.1	PATCH: Performance issue and memory leaks removed.
3.1	MINOR: New options pm_topRowMarginInTimeArea, pm_bottomRowMarginInTimeArea, pm_subRowDistanceInTimeArea, pm_topBarSymbolsVisible. MINOR: New option pm_linksVisibleInActivitiesView MINOR: New option timescaleNavigationMode MINOR: New link property PM_RoutingType and new option pm_defaultLinkRoutingType MINOR: New option pm_selectionColor MINOR: New option pm_splitterHighlightingColor
3.0	MINOR: New objects TooltipTemplate, TableRowDefinition/TableCellDefinition, DateLine including add/update/remove methods and properties named PM_(Bar/Curve)TooltipTemplateID and PM_TableRowDefinitionID on several objects. MINOR: New properties like PM_RowSelectable/PM_BarSelectable, PM_RowCollapsible on several objects. MINOR: New property PM_ViewArea on Resource objects. MINOR: New properties BaselineStart/BaseLineEnd, DueDate, ReleaseDate plus color properties on Activity objects. MINOR: New properties PM_BarHeight, PM_BarTextWrapMode, PM_EndIsSnapTarget/ PM_StartIsSnapTarget, PM_SnapTargetsForStart/ PM_SnapTargetsForEnd on Activity and Allocation objects. MINOR: New properties PM_CollapsedRowDesign/ PM_ExpandedRowDesign, PM_CollapseState/PM_CurveCollapseState, PM_MinimumRowHeight on Activity and Resource objects.

Version	Description of changes
	MINOR: New properties EarliestEnd/EarliestStart, LatestEnd/LatestStart, MustEndOn/MustStartOn plus color properties, and PM_EarliestDragStart/ PM_LatestDragEnd on Activity and Allocation objects. MINOR: New method setTimeResolutionForView. MINOR: Many new color options e.g. for coloring the timescale. MINOR: New callbacks onClicked, onCollapseStateChanged/ onCurveCollapseStateChanged, onTableCellDefinitionWidthChanged, onTimeAreaViewParametersChanged, onVerticalScrollOffsetChanged. MINOR: And some more object properties and options.
2.1	MINOR: New method about. MINOR: New message boxes for invalid, expiring, expired, not existing license.
2.0	MAJOR: Now the setting of a license key is mandatory. MINOR: New method removeAll. MINOR: New option locale. MINOR: New allocation properties PM_ProgressColor and PM_ProgressNonworkingColor. PATCH: Activity property Editable now marked as deprecated. MINOR: New option pm_linksVisibleInResourcesView.
1.0	Initial release.

2 System Requirements

2.1 Supported Browsers and Versions

Google Chrome (current version at delivery date of library)
Mozilla Firefox (current version at delivery date of library)
Apple Safari (current version at delivery date of library)
Microsoft Edge (current version >= 80 at delivery date of library*)
Microsoft Internet Explorer 11 (with limited support and reduced performance)

* The versions below 80 of Microsoft Edge have glitches in SVG support!

2.2 Needed 3rd Party Libraries and Versions

Library	Supported Versions	Comment
jQuery	2.x.x/3.x.x	Required. Needed for HTML handling. Versions 2.x.x support older Internet Explorer versions (but these are not supported by VSW Base!). URL: https://jquery.com/

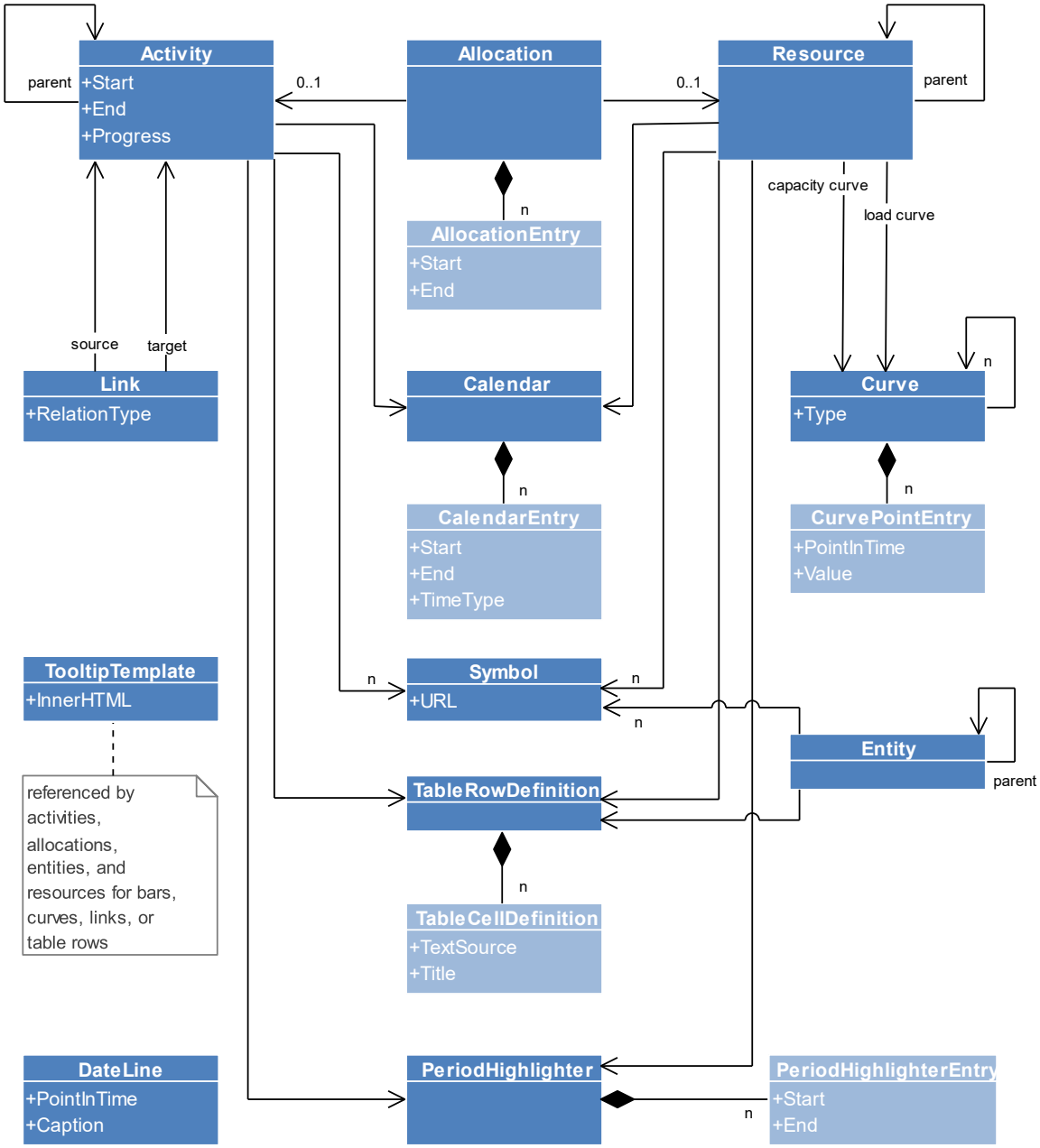
Library	Supported Versions	Comment
jQuery UI	1.11.x/1.12.x	Required. Needed as widget factory. URL: https://jqueryui.com/
D3.js	3.x/4.x/5.x	Required. Needed for SVG handling. Versions 3.x are not modular. Beginning with version 5.0.0 Internet Explorer is not supported anymore. URL: https://d3js.org/
Hammer.js	2.0.8	Required. Needed for touch and mouse gesture handling. URL: https://hammerjs.github.io/
TinyColor	1.4.1	Required. Needed for calculating derived colors e.g. for coloring non-working times. URL: https://bgrins.github.io/TinyColor/
Moment.js/ Moment.Timezone	2.x.x/ 0.x.x	Optional. Needed only, when using option "timeZone". The developer can decide, which data to serve with Moment Timezone. URL: https://momentjs.com/

The jQuery plug-in jquery.mousewheel that was required until VSW SE 3.1 is not needed for VSW SE 3.2 and up anymore.

3 Overview

The following diagram summarizes all object types described in this document and their relationships using the UML class diagram notation. Only those object properties are listed that are essential for understanding the concept of this data model.

The most important types (Activity, Allocation, Resource, Link, and Calendar) that encapsulate the core of a business logic are placed at the top of the diagram. Objects of any type other than AllocationEntry, CalendarEntry, CurvePointEntry, and TableCellDefinition (see the pale blue shapes) can be managed by calling methods of the widget (see add…, update…, and remove…).



4 Object Model

The object model of the Visual Scheduling Widget Base is designed for resource planning in general, but is extended to cover presentations all views, activities view, resources view, and loads 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.

4.1 Activity

An Activity object defines the properties of a single activity.

Activity Property Name	Type	Description
BarText	string	**Optional, default: undefined** – Text to display in the bar.
BaselineEnd	Date|string	**Optional, default: undefined** – Baseline 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.
BaselineStart	Date|string	**Optional, default: undefined** – Baseline 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.
CalendarID	string	**Optional, default: undefined** – Corresponding calendar. If undefined, then the default calendar specified by the option defaultCalendarID will be used. See also option pm_activityCalendarsEnabled.
DueDate	Date|string	**Optional, default: undefined** – Due 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

Activity Property Name	Type	Description
		standardized way, one has to be careful about it. See also option pm_releaseDueDateConnectionsVisible, if you want the widget to draw a connection line between a due date and a release date.
EarliestEnd	Date|string	**Optional, default: undefined** – If defined, an additional symbol will be displayed to indicate this date. 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.
EarliestStart	Date|string	**Optional, default: undefined** – If defined, an additional symbol will be displayed to indicate this date. 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.
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.
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.

Activity Property Name	Type	Description
ID	string	**Required** – Identifier of the activity.
LatestEnd	Date|string	**Optional, default: undefined** – If defined, an additional symbol will be displayed to indicate this date. 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.
LatestStart	Date|string	**Optional, default: undefined** – If defined, an additional symbol will be displayed to indicate this date. 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.
MustEndOn	Date|string	**Optional, default: undefined** – If defined, an additional symbol will be displayed to indicate this date. 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.
MustStartOn	Date|string	**Optional, default: undefined** – If defined, an additional symbol will be displayed to indicate this date. 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

Activity Property Name	Type	Description
		formatted date string in a standardized way, one has to be careful about it.
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.
PM_AllowedBarDragModes	number (see enum ActivityBarDragModes)	**Optional, default: value of option pm_defaultActivityAllowedBarDragModes** – This option determines the allowed bar drag modes for this activity in the activities view (these can be overwritten using the callback canDrag).
PM_BarHeight	number (≥ 0, ≤ 1000)	**Optional, default: value in option pm_defaultActivityBarHeight** – Height of the bars in pixels. This attribute is useful, when more than one line of text is shown inside (see attribute BarText). Proposal: For one line take 22, for two lines 38, for three lines 54, and so on. When no progress bar is needed, then you can subtract 4 from the value.
PM_BarSelectable	boolean	**Optional, default: value of option pm_defaultActivityBarSelectable** – If set to true, then the bar representing this activity will be selectable.
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.
PM_BarTextPrefixSymbolHeight	number	**Optional, default: 12** – Height of the bar symbol before the text (see property PM_BarTextSymbolSymbolID) in pixels at a zoom factor of 100%.
PM_BarTextPrefixSymbolID	string	**Optional, default: undefined** – Identifier of the symbol to be

Activity Property Name	Type	Description
		shown before the text inside of the activity bar. The symbol will be shown vertically centered inside the bar.
PM_BarTextPrefixSymbolWidth	number	**Optional, default: 12** – Width of the bar symbol before the text (see property PM_BarTextPrefixSymbolID) in pixels at a zoom factor of 100%.
PM_BarTextWrapMode	number (see enum TextWrapMode)	**Optional, default: TextWrapMode.None** – Specifies whether the text inside the bar is wrapped.
PM_BarTooltipTemplateID	string	**Optional, default: undefined** – ID of a tooltip template. The template is used for tooltips that appear on the activity bars.
PM_BaselineBorderColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color for the border of the baseline bar. If undefined, a default value of the widget will be used.
PM_BaselineColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color for the working time periods of the baseline bar. The nonworking time periods of the bar will be colored with the same color as long as the property PM_BaselineNonworkingTimeCol or is undefined or set to "calculated". If undefined, a default value of the widget will be used.
PM_BaselineNonworkingTimeColor	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 baseline 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_BaselineColor 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

Activity Property Name	Type	Description
		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_CollapsedRowDesign	number (see enum RowDesigns)	**Optional, default: value in option pm_defaultActivityCollapsedRowDesign** – Specifies how the time area is filled when the row is collapsed and visible. See enum RowDesigns in the Enumerations chapter for details.
PM_CollapseState	number (see enum CollapseState)	**Optional, default: -1** – Specifies whether the row of the activity should be expanded or collapsed when displayed. See also callback onCollapseStateChanged. -1: no change 0: display activity row in an expanded way 1: display activity row in a collapsed way
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. If undefined, a default value of the widget will be used.
PM_CurveCollapseState	number (see enum CollapseState)	**Optional, default: -1** – Specifies whether the curves in a activity row should be expanded or collapsed when displayed (only applicable, when option curvePanesVisibleInActivitiesView is set). See also callback onCurveCollapseStateChanged. -1: no change 0: display curves 1: hide curves
PM_DueDateColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** Color for the due date symbol. If undefined, a default value of the widget will be used.

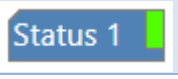
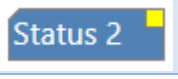
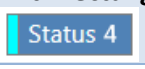
Activity Property Name	Type	Description
PM_EarliestDragStart	Date|string	**Optional, default: undefined** – If set, then the time before the given date is grayed, when beginning to drag the activity bar. If the option pm_dragDatesLimitingInteraction is set to true, then the bar itself cannot be dragged before the date.
PM_EarliestEndColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: value in option pm_defaultActivityConstraintSymbolColor** – Color for the EarliestEnd constraint symbol.
PM_EarliestStartColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: value in option pm_defaultActivityConstraintSymbolColor** – Color for the EarliestStart constraint symbol.
PM_ExpandedRowDesign	number (see enum RowDesigns)	**Optional, default: value in option pm_defaultActivityExpandedRowDesign** – Specifies how the time area is filled when the row is expanded and visible. See enum RowDesigns in the Enumerations chapter for details.
PM_HasChildren	boolean	**Optional, default: false** – If set to true, then the row representing this activity will be collapsible/expandable even when there are no children defined. This serves for lazy loading.
PM_LatestDragEnd	Date|string	**Optional, default: undefined** – If set, then the time after the given date is grayed, when beginning to drag the activity bar. If the option pm_dragDatesLimitingInteraction is set to true, then the bar itself cannot be dragged after the date.
PM_LatestEndColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: value in option pm_defaultActivityConstraintSymbolColor** – Color for the LatestEnd constraint symbol.
PM_LatestStartColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: value in option pm_defaultActivityConstraintSymbolColor** – Color for the LatestStart constraint symbol.

Activity Property Name	Type	Description
PM_LeftBarSymbolHeight	number	**Optional, default: 12** – Height of the left bar symbol (see property PM_LeftBarSymbolID) in pixels at a zoom factor of 100%.
PM_LeftBarSymbolID	string	**Optional, default: undefined** – Identifier of the symbol to be shown at the left side of the activity bar. The symbol will be shown vertically centered inside the bar. See also PM_RightBarSymbolID, PM_LeftBarSymbolHeight, and PM_LeftBarSymbolWidth.
PM_LeftBarSymbolWidth	number	**Optional, default: 12** – Width of the left bar symbol (see property PM_LeftBarSymbolID) in pixels at a zoom factor of 100%.
PM_MinimumRowHeight	number	**Optional, default: value in option pm_defaultMinimumActivityRowHeight** – Minimum height of the activity row in pixels. This attribute is useful, when more than one line of text is shown inside the table cells. Proposal: For one line take 36*, for two lines 52, for three lines 68, and so on. In order to have the same height also, when no bar is placed in the row, take the maximum bar height adding 20 (f.e. 42) as minimum. For using word wrapping in table cells, it is necessary to use a table row definition by setting the property PM_TableRowDefinitionID and setting the property WrapMode in a contained table cell definition.
PM_MustEndOnColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: value in option pm_defaultActivityConstraintSymbolColor** – Color for the MustEndOn constraint symbol.
PM_MustStartOnColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: value in option pm_defaultActivityConstraintSymbolColor** – Color for the MustStartOn constraint symbol.

Activity Property Name	Type	Description
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_PeriodHighlighterID	string	**Optional, default: undefined** – Reference to a period highlighter object that contains colored time periods. This can be used to show shifts or exceptions to the calendar (see property CalendarID) that defines work and non-work times.
PM_PredictedEndColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color for the predicted end bar.
PM_ProgressBackgroundColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: value in option pm_defaultActivityProgressBackgroundColor** – Color for the background of the progress bar region.
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_ProgressNonworkingTimeColor is undefined. 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_ReleaseDateColor	string (CSS color value, e.g. "#ff0000",	**Optional, default: undefined**

Activity Property Name	Type	Description
	"rgb(255, 0, 0)", or "red")	Color for the release date symbol. If undefined, a default value of the widget will be used.
PM_RightBarSymbolHeight	number	**Optional, default: 12** – Height of the right bar symbol (see property PM_RightBarSymbolID) in pixels at a zoom factor of 100%.
PM_RightBarSymbolID	string	**Optional, default: undefined** – Identifier of the symbol to be shown at the right side of the activity bar. The symbol will be shown vertically centered inside the bar. See also PM_LeftBarSymbolID, PM_RightBarSymbolHeight, and PM_RightBarSymbolWidth.
PM_RightBarSymbolWidth	number	**Optional, default: 12** – Width of the right bar symbol (see property PM_RightBarSymbolID) in pixels at a zoom factor of 100%.
PM_RowCollapsible	boolean	**Optional, default: value of option pm_defaultActivityRowCollapsible** – If set to true, then the row representing this activity will be interactively collapsible when children exist.
PM_RowSelectable	boolean	**Optional, default: value of option pm_defaultActivityRowSelectable** – If set to true, then the row representing this activity will be selectable.
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.

Activity Property Name	Type	Description
		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_RowTooltipTemplateID	string	**Optional, default: undefined** – ID of a tooltip template. The template is used for tooltips that appear on the activity table rows.
PM_SnapTargetsForEnd	number (see enum SnapTargets)	**Optional, default: value of widget option pm_defaultActivitySnap-TargetsForEnd** – When dragging horizontaly, then the visible end date of this allocation will optionally be snapping to date lines and calendar grids. The user can override an active snapping by pressing the ALT key while dragging. See also option PM_MaximumSnapDistance.
PM_SnapTargetsForStart	number (see enum SnapTargets)	**Optional, default: value of widget option pm_defaultActivitySnap-TargetsForStart** – When dragging horizontally, then the visible start date of this activity will optionally be snapping to date lines and calendar grids. The user can override an active snapping by pressing the ALT key while dragging. See also option PM_MaximumSnapDistance.
PM_Status1Color	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color for the status symbol to the right of the bar. If undefined, no symbol appears. Only visible, when property PM_Status1Visible is true.
PM_Status1Visible	boolean	**Optional, default: false** – If set to true and the corresponding status color is set in property PM_Status1Color, then a

Activity Property Name	Type	Description
		predefined symbol is displayed to the right of the bar. 
PM_Status2Color	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color for the status symbol to the right of the bar. If undefined, no symbol appears. Only visible, when property PM_Status2Visible is true.
PM_Status2Visible	boolean	**Optional, default: false** – If set to true and the corresponding status color is set in property PM_Status2Color, then a predefined symbol is displayed to the right of the bar. 
PM_Status3Color	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color for the status symbol to the right of the bar. If undefined, no symbol appears. Only visible, when property PM_Status3Visible is true.
PM_Status3Visible	boolean	**Optional, default: false** – If set to true and the corresponding status color is set in property PM_Status3Color, then a predefined symbol is displayed to the right of the bar. 
PM_Status4Color	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color for the status symbol to the left of the bar. If undefined, no symbol appears. Only visible, when property PM_Status4Visible is true.
PM_Status4Visible	boolean	**Optional, default: false** – If set to true and the corresponding status color is set in property PM_Status4Color, then a predefined symbol is displayed to the left of the bar. **Note: This property may be used with rectangle bar shapes only!** 
PM_TableColor	string	**Optional, default: undefined** – Color for the table row.

Activity Property Name	Type	Description
	(CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	If undefined, a default value of the widget will be used.
PM_TableRowDefinitionID	string	**Optional, default: value of option pm_defaultActivityTableRowDefinitionID** – Identifier of a TableRowDefinition object, that defines the composition of the table row.
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_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_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%.
PredictedEnd	Date|string	**Optional, default: undefined** – A date that indicates the predicted end of the activity. This date is used to display a bar between this date and the end 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

Activity Property Name	Type	Description
		formatted date string in a standardized way, one has to be careful about it.
Progress	number (floating point; in percent; ≥ 0, ≤ 100)	**Optional, default: 0.0** – Used to display a completion layer.
ReleaseDate	Date|string	**Optional, default: undefined** – Release 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. See also option pm_releaseDueDateConnectionsVisible, if you want the widget to draw a connection line between a due date and a release date.
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.
TableText	string	**Optional, default: undefined** – Text to display in the table row (see also property PM_TableRowDefinitionID).

4.2 Allocations

4.2.1 Allocation

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

Allocation Property Name	Type	Description
ActivityID	string	**Optional, default: undefined** – Identifier of an Activity
BarText	string	**Optional, default: undefined** – Text to display in the bar.
EarliestEnd	Date|string	**Optional, default: undefined** – If defined, an additional symbol will be displayed to indicate this date. 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.
EarliestStart	Date|string	**Optional, default: undefined** – If defined, an additional symbol will be displayed to indicate this date. 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.
Entries	AllocationEntry[]	**Optional, default: undefined** – array of allocation entries.
ID	string	**Required** – Identifier of the allocation.
LatestEnd	Date|string	**Optional, default: undefined** – If defined, an additional symbol will be displayed to indicate this date. 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.

Allocation Property Name	Type	Description
LatestStart	Date|string	**Optional, default: undefined** – If defined, an additional symbol will be displayed to indicate this date. 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.
MustEndOn	Date|string	**Optional, default: undefined** – If defined, an additional symbol will be displayed to indicate this date. 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.
MustStartOn	Date|string	**Optional, default: undefined** – If defined, an additional symbol will be displayed to indicate this date. 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.
PM_AllowedBarDragModes	number (see enum AllocationBarDragModes)	**Optional, default: value of option pm_defaultAllocationAllowedBarDragModes** – This option determines the allowed bar drag modes for this allocation in the resources view (these can be overwritten using the callback canDrag).
PM_BarHeight	number (≥ 0, ≤ 1000)	**Optional, default: value in option pm_defaultAllocationBarHeight**

Allocation Property Name	Type	Description
		– Height of the bar in pixels. This is useful, when more than one line of text is shown inside (see attribute BarText). Proposal: For one line take 22, for two lines 38, for three lines 54, and so on. When no progress bar is needed, then you can subtract 4 from the value.
PM_BarSelectable	boolean	**Optional, default: value of option pm_defaultAllocationBarSelectable** – If set to true, then the bar representing this allocation will be selectable.
PM_BarShape	number (see enum AllocationBarShape)	**Optional, default: value in option pm_defaultAllocationBarShape** – This option defines which shape should be used by default for the visualization allocation bars.
PM_BarTextPrefixSymbolHeight	number	**Optional, default: 12** – Height of the bar symbol before the text (see property PM_BarTextSymbolSymbolID) in pixels at a zoom factor of 100%.
PM_BarTextPrefixSymbolID	string	**Optional, default: undefined** – Identifier of the symbol to be shown before the text inside of the allocation bar. The symbol will be shown vertically centered inside the bar.
PM_BarTextPrefixSymbolWidth	number	**Optional, default: 12** – Width of the bar symbol before the text (see property PM_BarTextPrefixSymbolID) in pixels at a zoom factor of 100%.
PM_BarTextWrapMode	number (see enum TextWrapMode)	**Optional, default: TextWrapMode.None** – Specifies whether the text inside the bar is wrapped.
PM_BarTooltipTemplateID	string	**Optional, default: undefined** – ID of a tooltip template. The template is used for tooltips that appear on the allocation bars.
PM_BarTopOffset	number	**Optional, default: 0** – Offset of the bar in pixels relative to its upper side. A negative number

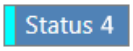
Allocation Property Name	Type	Description
		will shift the bar upwards, a positive number will shift the bar downwards.
PM_BorderColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red" or "calculated")	**Optional, default: undefined** – Color for the border of the bar. If undefined, 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_EarliestDragStart	Date|string	**Optional, default: undefined** – If set, then the time before the given date is grayed, when beginning to drag the allocation bar. If the option pm_dragDatesLimitingInteraction is set to true, then the bar itself cannot be dragged before the date.
PM_EarliestEndColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: value in option pm_defaultAllocationConstraintSymbolColor** – Color for the EarliestEnd constraint symbol.
PM_EarliestStartColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: value in option pm_defaultAllocationConstraintSymbolColor** – Color for the EarliestStart constraint symbol.
PM_EndIsSnapTarget	boolean	**Optional, default: true** – If set to true, then the visible end date of this allocation in the resources view is used as a snap target for a dragged bar (see attributes PM_SnapTargetsForStart and PM_SnapTargetsForEnd and option PM_MaximumSnapDistance)
PM_LatestDragEnd	Date|string	**Optional, default: undefined** – If set, then the time after the given date is grayed, when beginning to drag the allocation bar. If the option pm_dragDatesLimitingInteraction

Allocation Property Name	Type	Description
		is set to true, then the bar itself cannot be dragged after the date.
PM_LatestEndColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: value in option pm_defaultAllocationConstraintSymbolColor** – Color for the LatestEnd constraint symbol.
PM_LatestStartColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: value in option pm_defaultAllocationConstraintSymbolColor** – Color for the LatestStart constraint symbol.
PM_LeftBarSymbolHeight	number	**Optional, default: 12** – Height of the left bar symbol (see property PM_LeftBarSymbolID) in pixels at a zoom factor of 100%.
PM_LeftBarSymbolID	string	**Optional, default: undefined** – Identifier of the symbol to be shown at the left side of the allocation bar. The symbol will be shown vertically centered inside the bar. See also PM_RightBarSymbolID, PM_LeftBarSymbolHeight, and PM_LeftBarSymbolWidth.
PM_LeftBarSymbolWidth	number	**Optional, default: 12** – Width of the left bar symbol (see property PM_LeftBarSymbolID) in pixels at a zoom factor of 100%.
PM_MustEndOnColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: value in option pm_defaultAllocationConstraintSymbolColor** – Color for the MustEndOn constraint symbol.
PM_MustStartOnColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: value in option pm_defaultAllocationConstraintSymbolColor** – Color for the MustStartOn constraint symbol.
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 activity, if available, will be used. If set to "calculated", a color will be calculated using the color defined by the PM_Color property.
PM_PredictedEndColor	string	**Optional, default: undefined** – Color for the predicted end bar.

Allocation Property Name	Type	Description
	(CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	
PM_ProgressBackgroundColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: value in option pm_defaultAllocationProgressBackgroundColor** – Color for the background of the progress bar region.
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_ProgressNonworkingTimeColor is undefined. If undefined, a value of the property with the same in the corresponding activity, if available, 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 value of the property with the same in the corresponding activity, if available, will be used. If set to "calculated", a color will be calculated using the color defined by the PM_ProgressColor property.
PM_RightBarSymbolHeight	number	**Optional, default: 12** – Height of the right bar symbol (see property PM_RightBarSymbolID) in pixels at a zoom factor of 100%.
PM_RightBarSymbolID	string	**Optional, default: undefined** – Identifier of the symbol to be shown at the right side of the allocation bar. The symbol will be shown vertically centered inside the bar. See also PM_LeftBarSymbolID, PM_RightBarSymbolHeight, and PM_RightBarSymbolWidth.
PM_RightBarSymbolWidth	number	**Optional, default: 12** – Width of the right bar symbol (see

Allocation Property Name	Type	Description
		property PM_RightBarSymbolID) in pixels at a zoom factor of 100%.
PM_SnapTargetsForEnd	number (see enum SnapTargets)	**Optional, default: value of widget option pm_defaultAllocationSnap-TargetsForEnd** – When dragging horizontaly, then the visible end date of this allocation will optionally be snapping to date lines, calendar grids, and start or end dates of other allocations in same row, when dragging lets these dates get near the end date (see attribute PM_EndIsSnapTarget). The user can override an active snapping by pressing the ALT key while dragging.
PM_SnapTargetsForStart	number (see enum SnapTargets)	**Optional, default: value of widget option pm_defaultAllocationSnap-TargetsForStart** – When dragging horizontally, then the visible start date of this allocation will optionally be snapping to date lines, calendar grids, and start or end dates of other allocations in same row, when dragging lets these dates get near the start date (see attribute PM_StartIsSnapTarget). The user can override an active snapping by pressing the ALT key while dragging.
PM_StartIsSnapTarget	boolean	**Optional, default: true** – If set to true, then the visible start date of this allocation in the resources view is used as a snap target for a dragged bar (see attributes PM_SnapTargetsForStart and PM_SnapTargetsForEnd and option PM_MaximumSnapDistance)
PM_Status1Color	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color for the status symbol to the right of the bar. If undefined, no symbol appears. Only visible, when property PM_Status1Visible is true.

Allocation Property Name	Type	Description
PM_Status1Visible	boolean	**Optional, default: false** – If set to true and the corresponding status color is set in property PM_Status1Color, then a predefined symbol is displayed to the right of the bar. 
PM_Status2Color	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color for the status symbol to the right of the bar. If undefined, no symbol appears. Only visible, when property PM_Status2Visible is true.
PM_Status2Visible	boolean	**Optional, default: false** – If set to true and the corresponding status color is set in property PM_Status2Color, then a predefined symbol is displayed to the right of the bar. 
PM_Status3Color	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color for the status symbol to the right of the bar. If undefined, no symbol appears. Only visible, when property PM_Status3Visible is true.
PM_Status3Visible	boolean	**Optional, default: false** – If set to true and the corresponding status color is set in property PM_Status3Color, then a predefined symbol is displayed to the right of the bar. 
PM_Status4Color	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color for the status symbol to the left of the bar. If undefined, no symbol appears. Only visible, when property PM_Status4Visible is true.
PM_Status4Visible	boolean	**Optional, default: false** – If set to true and the corresponding status color is set in property PM_Status4Color, then a predefined symbol is displayed to the left of the bar.

Allocation Property Name	Type	Description
		Note: This property may be used with rectangle bar shapes only! 
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_TopRightBarSymbolID	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%.
PredictedEnd	Date|string	**Optional, default: undefined** – A date that indicates the predicted end of the allocation. This date is used to display a bar between this date and the end of the allocation. 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; ≥ 0, ≤ 100)	**Optional, default: 0.0** – Used to display a completion layer.
ResourceID	string	**Optional, default: undefined** – Identifier of a Resource
SuitableResourceIDs	string[]	**Optional, default: undefined** – An array of IDs of those resources to which the allocation could be assigned.

Allocation Property Name	Type	Description
		If the array is defined, then all rows of resources that are not listed in that array will be covered by a half-transparent curtain. If the array is empty, all resource rows will be covered. If the array is not defined, then all rows are displayed in the normal way without any covering. Also see options pm_suitableResourcesOverlayColor and pm_unsuitableResourcesOverlayColor.

4.2.2 AllocationEntry

AllocationEntry Property Name	Type	Description
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: " YYYY-MM-DDThh:mm:ssZ" (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.
PM_Height	number (≥ 0, ≤ 1000)	**Optional, default: value in option pm_defaultAllocationBarHeight** – Height of the entry in pixels.
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).

AllocationEntry Property Name	Type	Description
		If set to "calculated", a color will be calculated using the color defined by the PM_Color property.
PM_RelativeTopOffset	number	**Optional, default: 0** – Offset of the entry in pixels relative to the upper side of the corresponding allocation. A negative number will shift the entry upwards, a positive number will shift the entry downwards.
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: " YYYY-MM-DDThh:mm:ssZ" (this implies that the date is specified in UTC).

4.3 Calendars

4.3.1 Calendar

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

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

4.3.2 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.

CalendarEntry Property Name	Type	Description
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

CalendarEntry Property Name	Type	Description
		formatted date string in a standardized way, one has to be careful about it.
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.
TimeType	number (see enum TimeType)	**Optional, default: 1** 1: WorkingTime, 2: NonworkingTime

4.4 Curves

Curve objects serve to define values over time that can be shown as capacity or load inside resource and activity rows (see properties LoadCurveID and CapacityCurveID in Resource object). Additionally it is possible to stack curves when using curve object of stack type. At the moment there are no curve types that calculate their values automatically.

4.4.1 Curve

Curve Property Name	Type	Description
CurveIDs	string[]	**Optional, default: undefined** – Array of curve IDs (in case of StackedCurve only)
CurvePointEntries	CurvePointEntry[]	**Optional, default: undefined** – Array of point entries (in case of PointCurve only)
ID	string	**Required** – Identifier of the stacked curve
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 Note: If a curve is used as an inventory curve, then the default is "transparent"
PM_InterpolationType	Number (see enum CurveInterpolationType)	**Optional, default: undefined** – Type of interpolation. At the moment there are restrictions concerning putting curves of linear interpolation type into curve stacks. It is recommended to use this interpolation type only inside curve lists.
PM_OverloadColor	string (CSS color value, e.g. "#ff0000",	**Optional, default: undefined** – Used, when the curve is used as the load curve that referenced directly by the property LoadCurveID at the object. Then the area

Curve Property Name	Type	Description
	"rgb(255, 0, 0)", or "red")	above the capacity curve will be colored by this color when the load is higher than the capacity.
PM_StrokeColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** – Color of the curve line itself
Type	number (see enum CurveType)	**Optional, default: 0** – Type of the curve. At the moment it is recommended not to put lists or stacks into other lists/stacks!

4.4.2 CurvePointEntry

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.

4.5 DateLine

A DateLine object is a pure presentation object and defines the properties of a single date line.

DateLine Property Name	Type	Description
Caption	string	**Optional, default: ""** – Text for the caption of the date line.
CaptionColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: "black"** – Color of the caption.
CaptionOrientation	number (see enum DateLineCaptionOrientation)	**Optional, default: 2** – Specifies whether the caption should be oriented vertically or horizontally.
CaptionPosition	number (see enum DateLineCaptionPosition)	**Optional, default: 1** – Specifies where the caption should be positioned relative to the date line.

DateLine Property Name	Type	Description
Color	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: "black"** – Color of the line.
DashArray	string	**Optional, default: "none"** – Pattern of dashes and gaps for drawing the date line. For further information, please see https://www.w3.org/TR/SVG11/painting.html#StrokeDasharrayProperty or https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dasharray. The value "none" indicates that no dashing is used. In this case, the line is drawn solid.
Draggable	boolean	**Optional, default: false** – If set to true, then the date line is draggable and the callback onDrop is triggered, when dropping it at a new date.
ID	string	**Required** – Identifier of this date line.
InFrontOfBars	boolean	**Optional, default: true** – Determines how the date line is displayed. If set to false, the date line will be overlapped by the bars. Otherwise, the line will be displayed in front of the bars.
PointInTime	Date|string	**Optional, default: undefined** – Date, where the date line should become visible. The date line only gets visible, when the date is set and the date lies between the values of the widget options start and end.
Width	number ≥ 0	**Optional, default: 1** – Line width of the date line.

4.6 Entity

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

Entity Property Name	Type	Description
Duration	number (in milliseconds)	**Optional, default: undefined** – Duration of the pure working time of the entity. This property is used, for example, when moving the entity from the entities table to the Gantt diagram to display a bar of correct length during interaction.
ID	string	**Required** – Identifier of this entity
ParentID	string	**Optional, default: undefined** – Description of the entity (freely usable)

Entity Property Name	Type	Description
PM_AllowedRowDragModes	number (see enum RowDragModes)	**Optional, default: value of option pm_defaultEntityAllowedRowDragModes** – This option determines the allowed row drag modes for this entity when the entities table is visible (these can be overwritten using the callback canDrag).
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_HasChildren	boolean	**Optional, default: false** – If set to true, then the row representing this entity will be collapsible/expandable even when there are no children defined. This serves for lazy loading.
PM_MinimumRowHeight	number	**Optional, default: value in option pm_defaultMinimumEntityRowHeight** – Minimum height of the entity row in pixels. This attribute is useful, when more than one line of text is shown inside the table cells. Proposal: For one line take 36*, for two lines 52, for three lines 68, and so on. In order to have the same height also, when no bar is placed in the row, take the maximum bar height adding 20 (f.e. 42) as minimum. For using word wrapping in table cells, it is necessary to use a table row definition by setting the property PM_TableRowDefinitionID and setting the property WrapMode in a contained table cell definition.
PM_RowCollapsible	boolean	**Optional, default: value of option pm_defaultEntityRowCollapsible** – If set to true, then the row representing this entity will be interactively collapsible when children exist.
PM_RowSelectable	boolean	**Optional, default: value of option pm_defaultEntityRowSelectable** – If set to true, then the row representing this entity will be selectable.
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,

Entity Property Name	Type	Description
		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_RowTooltipTemplateID	string	**Optional, default: undefined** – ID of a tooltip template. The template is used for tooltips that appear on the entity table rows.
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_TableRowDefinitionID	string	**Optional, default: value of option pm_defaultEntityTableRowDefinitionID** – Identifier of a TableRowDefinition object that defines the composition of the table row.
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.
TableText	string	**Optional, default: undefined** – Text to display in the table (see also property PM_TableRowDefinitionID).

4.7 Link

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

Link Property Name	Type	Description
ID	string	**Required** – Identifier of this link
PM_Color	string (CSS color value, e.g. "#ff0000", "rgb(255,0, 0)", or "red")	**Optional, default: "black"** – Color for the line.
PM_DashArray	string	**Optional, default: "none"** – Pattern of dashes and gaps for drawing the line. For further information, please see https://www.w3.org/TR/SVG11/painting.html#StrokeDasharrayProperty

Link Property Name	Type	Description
		or https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dasharray. The value "none" indicates that no dashing is used. In this case, the link is drawn solid.
PM_RoutingType	number (see enum LinkRoutingType)	**Optional, default: value of option pm_defaultLinkRoutingType** – type of the link routing. 1: Curved, 2: Orthogonal
PM_Selectable	boolean	**Optional, default: value of option pm_defaultLinkSelectable** – If set to true, then the link will be selectable.
PM_TooltipTemplateID	string	**Optional, default: undefined** – ID of a tooltip template. The template is used for tooltips that appear on the links.
PM_Width	number ≥ 0	**Optional, default: 1** – Line width of the link. The link arrow is also affected by this property.
RelationType	number	**Optional, default: 0** – The relation type is used for drawing: 0: Finish-Start, 1: Finish-Finish, 2: Start-Start
SourceActivityID	string	**Required** – Identifier of the source activity
TargetActivityID	string	**Required** – Identifier of the target activity

4.8 PeriodHighlighters

A PeriodHighlighter object is a pure presentation object and defines the properties of a series of time periods that can be shown on each resource row and activity row (see property PM_PeriodHighlighterID there). Each time period can be colored independently and can have a caption. Period highlighters also support the callbacks onShowTooltip, onDoubleClicked, and onShowContextMenu. In contrast to the grids created by Calendar objects, the time periods do not define work or non-work times, but only highlight time periods visually.

4.8.1 PeriodHighlighter

PeriodHighlighter Property Name	Type	Description
Entries	PeriodHighlighterEntry[]	**Required** – Array of entries that contain single time periods.
ID	string	**Required** – Identifier of this period highlighter.

4.8.2 PeriodHighlighterEntry

PeriodHighlighterEntry Property Name	Type	Description
Caption	string	**Optional, default: ""** – Text to show on the time period.
CaptionColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: "white"** – Color of the caption.
Color	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: "rgba(0,0,0,0.1)"** – Color of this time period.
End	Date|string	**Required** – End of the time period.
Start	Date|string	**Required** – Start of the time period.

4.9 Resource

A Resource object defines the properties of a single resource.

Resource Property Name	Type	Description
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.
ID	string	**Required** – Identifier of the resource
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.
(Deprecated!) Name	string	**Optional, default: undefined** – Name of the resource (freely usable)
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.

Resource Property Name	Type	Description
PM_CollapsedRowDesign	number (see enum RowDesigns)	**Optional, default: value in option pm_defaultResourceCollapsedRowDesign** – Specifies how the time area is filled when the row is collapsed and visible. See enum RowDesigns in the Enumerations chapter for details.
PM_CollapseState	number (see enum CollapseState)	**Optional, default: -1** – Specifies whether the row of the resource should be expanded or collapsed when displayed. See also callback onCollapseStateChanged. -1: no change 0: display resource row in an expanded way 1: display resource row in a collapsed way
PM_CurveCollapseState	number (see enum CollapseState)	**Optional, default: -1** – Specifies whether the curves in a resource row should be expanded or collapsed when displayed. See also callback onCurveCollapseStateChanged. -1: no change 0: display curves 1: hide curves
PM_CurveTooltipTemplateID	string	**Optional, default: undefined** – ID of a tooltip template. The template is used for tooltips that appear on the curve area of resources.
PM_ExpandedRowDesign	number (see enum RowDesigns)	**Optional, default: value in option pm_defaultResourceExpandedRowDesign** – Specifies how the time area is filled when the row is expanded and visible. See enum RowDesigns in the Enumerations chapter for details.
PM_HasChildren	boolean	**Optional, default: false** – If set to true, then the row representing this resource will be collapsible/expandable even when there are no children defined. This serves for lazy loading.
PM_HasCurves	boolean	**Optional, default: false** – If set to true, then the row representing this resource will be collapsible/expandable for curves even where there ae no curves defined. This serves for lazy loading.
PM_MinimumRowHeight	number	**Optional, default: value in option pm_defaultMinimumResourceRowHeight** – Minimum height of the resource row in pixels. This option is useful, when more than one line of text is shown inside the table cells. Proposal: For one line take 36*, for two lines 52, for three lines 68, and so on. In order to have the same height also, when no bar is placed in the row, take the maximum bar height adding 20 (f.e. 42) as minimum.

Resource Property Name	Type	Description
		For using word wrapping in table cells, it is necessary to use a table row definition by setting the property PM_TableRowDefinitionID and setting the property WrapMode in a contained table cell definition.
PM_PeriodHighlighterID	string	**Optional, default: undefined** – Reference to a period highlighter object that contains colored time periods. This can be used to show shifts or exceptions to the calendar (see property CalendarID) that defines work and non-work times.
PM_RowCollapsible	boolean	**Optional, default: value of option pm_defaultResourceRowCollapsible** – If set to true, then the row representing this resource will be interactively collapsible when children exist.
PM_RowSelectable	boolean	**Optional, default: value of option pm_defaultResourceRowSelectable** – If set to true, then the row representing this resource will be selectable.
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_RowTooltipTemplateID	string	**Optional, default: undefined** – ID of a tooltip template. The template is used for tooltips that appear on the resource table rows.
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_TableRowDefinitionID	string	**Optional, default: value of option pm_defaultResourceTableRowDefinitionID** –

Resource Property Name	Type	Description
		Identifier of a TableRowDefinition object, that defines the composition of the table row.
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_ViewArea	number (see enum ViewArea)	**Optional, default: Default** – If set to Top, then the resource and its children are shown in a separate top view area in the resources view. Only settable on resource with no ParentID set. See option pm_topViewAreaVisible.
TableText	string	**Optional, default: undefined** – Text to display in the table row (see also property PM_TableRowDefinitionID).

4.10 Symbol

A Symbol object is a pure presentation object and 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.

Symbol 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. '...'). See https://en.wikipedia.org/wiki/Data_URI_scheme)

4.11 TableRowDefinitions

A TableRowDefinition object defines the composition of a table row containing one or more cells. You can reference these objects with the property PM_TableRowDefinitionID of Activity, Resource, and Entity objects. It is possible to declare one table row definition to provide the table title for the views and the entities table by using the options pm_activity/resource/entityTableRowDefinitionIDForTitle.

4.11.1 TableRowDefinition

TableRowDefinition Property Name	Type	Description
CellDefinitions	TableCellDefinition[]	**Optional, default: [{ Title: "", TextSource: "TableText", Width: 200, HorizontalAlignment: HorizontalAlignment.Left }] –** Array of TableCellDefinition objects.
ID	string	**Required** – Identifier of this table row definition.

4.11.2 TableCellDefinition

TableCellDefinition Property Name	Type	Description
HorizontalAlignment	number (see enum HorizontalAlignment)	**Optional, default: Left** – Horizontal alignment of the shown text. The first column is always shown with left alignment because of the tree symbols on the left side.
HorizontalTitleAlignment	number (see enum HorizontalAlignment)	**Optional, default: Center** – Horizontal alignment of the shown title text. In the entities table the last column is always shown with center alignment.
MaximumWidth	number	**Optional, default: Infinity** – Maximum width of the table cell, when cell width is interactively modified.
MinimumWidth	number	**Optional, default: 3** – Minimum width of the table cell, when cell width is interactively modified.
SymbolIDSource	string	**Optional, default: ""** – Property to take the symbol ID out of the referencing activity, resource, or entity object. The symbol will be displayed in the cell inside a square that has the size of the minimum row height. The symbol will obey the HorizontalAlignment property. It is also possible to use the TextSource property along with this property, but there are the following restrictions: If using left alignment, the text will be indented so that it is to the right of

TableCellDefinition Property Name	Type	Description
		the symbol. If using center or right alignment, the symbol will be overlapped by the text.
TextSource	string	**Optional, default: "TableText"** – Property to take the text out of the referencing activity, resource, or entity object.
Title	string	**Optional, default: ""** – When the table row definition, that contains this table cell definition, is referenced by one of the options pm_activity/resource/entityTableRowDefinitionIDF orTitle, then the title defined here will be shown on the table title.
Width	number	**Optional, default: 200** – Width of the table cell.
WrapMode	number (see enum TextWrapMode)	**Optional, default: None** – If set, then it possible to show more than one line of text using newline characters ('\n').

4.12 TooltipTemplate

A TooltipTemplate object describes the appearance of a tooltip in the form of an HTML string. This string describes a DOM subtree and contains placeholders with references to the object properties to be displayed. At runtime, the placeholders are replaced by the values of the referenced object properties.

There are two ways to apply a template:

1. Either you can specify the template ID inside the out-parameter "tooltipTemplateID" of the onShowTooltip callback.

2. Or you can use the properties PM_TooltipTemplateID, PM_BarTooltipTemplateID, PM_RowTooltipTemplateID, and PM_CurveTooltipTemplateID of the activities, resources, allocations, links, and entities.

TooltipTemplate Property Name	Type	Description
ID	string	**Required** – Identifier of this tooltip template.

InnerHTML	string	**Required** – HTML string that describes the structure of a tooltip.
		
		This string contains the placeholders for object values surrounded by double curly braces {{ }}. For example, based on the following string a tooltip with a table containing three rows of key-value pairs is created, where the values are taken from the object properties "name", "firstName", and "age":
		
		`<table>`
		`  <tr><td>Name: </td><td>{{name}}</td></tr>`
		`  <tr><td>First name: </td><td>{{firstName}}</td></tr>`
		`  <tr><td>Age: </td><td>{{age}}</td></tr>`

TooltipTemplate Property Name	Type	Description
		`</table>` As an escape, the use of three open curly braces {{{ are displayed as {{. Additionally, the property name can be extended to contain the desired property type as in {{Start:date}}. At the moment only the type 'date' is possible besides 'string' (other property types are converted automatically with toString()). The type 'date' converts date values using the same format as other dates in the timescale and at the dragging date line captions. It is possible to get associated objects by using the following keywords: On activities: >Parent, >Calendar On resources: >Parent, >Calendar, >LoadCurve, >CapacityCurve On entities: >Parent On allocations: >Activity, >Resource On links: >SourceActivity, >TargetActivity On associated objects, you can then receive a property by using a prefixed dot: .propertyName. On tooltips for allocations you can get the Entry object by using #Entry as a keyword. On tooltips for curves you can show current values by using #Date, #Capacity, #Load.

5 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").nXYZWidget(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.

5.1 Options

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

Widget Option Name	Type	Description
additionalDateInterpretedAsEmpty	Date|string|null	**Optional, default: null** – If set, then on properties of date type the value can be set to the value given here and will be interpreted as being null/undefined/"". If given as a string, this string is converted to a Date object internally and each date will be checked by comparing the date values.
additionalDateStringInterpretedAsEmpty	string	**Optional, default: ""** – If set, then on properties of date type the value can be set to the value given here and will be interpreted as being null/undefined/"". Each date string will be checked by comparing the strings.
currentDate	Date|string|null	**Optional, default: null** – When set to a valid date, then a darkened area is positioned from the timescale start up to this date. The darkened area can be attributed by using the options pm_pastBackgroundFillColor/ LineColor/ LineWidth/ LineDashArray.
cursorDateLineVisible	boolean	**Optional, default: false** – If this option is set to true, an additional labeled date line will follow the mouse cursor.
curvePanesVisibleInActivitiesView	boolean	**Optional, default: false** – If this option is set to true, a curve pane is displayed in the ActivitiesView for each activity row. In each pane the curves of the resource first found in an allocation related to the corresponding activity are displayed. Please note: This option has to be set when instantiating the widget. If it is set later, it has no effect.
dateLineGridMode	number (see enum DateLineGridModes)	**Optional, default: Weekly** – This option determines the distance of the date lines shown.
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

Widget Option Name	Type	Description
		with constantly non-working time only.
editable	boolean	**Optional, default: true** – If set to false, nothing can be edited.
end	Date|string	**Required** – End date of the considered time area.
entitiesTableViewWidth	number	**Optional, default: null (means current table width)** – This setting defines the width of the entities table view when it becomes visible initially.
entitiesTableVisibleInActivitiesView	boolean	**Optional, default: false –** This option lets appear/disappear the entities table on the right side in the Activities View.
entitiesTableVisibleInResourcesView	boolean	**Optional, default: false –** This option lets appear/disappear the entities table on the right side in the Resources View.
(Deprecated!) entitiesTableWidth	number	**Optional, default: null** – Not recommended when using TableRowDefinition objects! This setting defines the width of the entities table. When not using TableRowDefinition objects, 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. It will appear only in one the following two cases: 1. If using the TableRowDefinition objects for defining the table and the property pm_entityTableRowDefinitionIDForTitle is not set. or 2. If using the deprecated callback onDetermineComumnDefinitions and there additionally the flag hasColumnTitles is **not** set in the callback (see there).

Widget Option Name	Type	Description
fixedTableColumnWidth	number	**Optional, default: 30** – This setting defines the width of the fixed table column that contains the numeric scale for the curves in each row.
interactiveActivationOfLoggingEnabled	boolean	**Optional, default: false** – If set to true, the user can activate the logging by using the keyboard shortcut shift-ctrl-alt-L. The record symbol will appear, the current state of the widget is saved and from then on all calls to the API are recorded. Pressing shift-ctrl-alt-L once again will stop the recording and download a file with the recorded actions. See also option **loggingEnabled**.
licenseKey	string	**Required** – Without a license key, the widget will not work at all. Please contact NETRONIC to get a license. This option must be set at the very beginning of the widget initialization and cannot be changed later at runtime.
locale	string (currently possible values: "da-DK", "de-DE", "en-GB", "en-US", "es-ES", "fi", "fr-FR", "it-IT", "nl-NL", "no", "pl", "pt", "pt-br", "sv")	**Optional, default: "en-US"** – This option will be used for showing the textual parts for date values in the timescale and for formatting date and time values in the timescale and numbers in the numeric scales of curves.
loggingEnabled	boolean	**Optional, default: false** – If set to true, the record symbol will appear, the current state of the widget is saved and from then on all calls to the API are recorded. Resetting this option to false will stop the recording and download a file with the recorded actions. See also option **interactiveActivationOfLoggingEnabled**.
multipleBarDraggingEnabled	boolean	**Optional, default: false** – If set to true, all selected bars are dragged at once. Also see callback options canDrag, onDragStart, onDrop. Currently, the allocation/activity properties PM_EarliestDragStart and PM_LatestDragEnd are not supported when dragging multiple bars. The allocation property

Widget Option Name	Type	Description
		SuitableResourceIDs is supported. When dragging starts, the allowed drag modes are taken from the allocation/activity that is dragged directly as default. This is modifiable by using the callback canDrag or one of the options pm_forcedActivity/AllocationAllowed BarDragModes.
multipleSelectionEnabled	boolean|number (allowed values: false, true, 2)	**Optional, default: true** – If set to true or 1 multiple bars can be selected all at once. When using the selection rectangle, a bar is selected even if it is only partially inside the rectangle. Additionally, if set to 2, the behavior when dragging a selection rectangle from left to right is different from that when dragging from right to left. In the first case, a bar is only selected when it is completely inside the rectangle. In the latter case, it will be selected even if it is only partially inside the rectangle.
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.
pm_activityCalendarsEnabled	boolean	**Option, default: true** – If set to true, calendars assigned to activities by setting the activity property CalendarID are displayed in the Activities View.
pm_activityTableRowDefinitionID ForTitle	string	**Optional, default: value of option pm_defaultActivityTableRowDefinitio nID** – ID of a TableRowDefinition object that will be used to show the table title in the activities view. In parallel, it is currently only possible to interactively change the column widths for the TableRowDefinition object that is referenced here.
pm_bottomRowMarginInTimeAre a	number	**Optional, default: 5** – Height of the margin between the bottom row border and bars above in pixels. The value is also used for the vertical margins of curve panes. See also

Widget Option Name	Type	Description
		pm_topRowMarginInTimeArea and pm_subRowDistanceInTimeArea.

pm_calendarGridColor	string (CSS color value) or Object	**Optional, default: "#f0f0f0"** – Specifies a color used to color the vertical stripes representing the nonworking times inside the diagram. If a string is given, then the widget uses the color for all view types. If an object is given, then one can specify a color string for each view type and one for not mentioned view types. Example:
		{
		0 /*activities view*/: "yellow",
		-1 /*other views*/: null
		}

pm_curvePanesCollapsibleInResourcesView	boolean	**Optional, default: true** – Specifies whether the curve panes can be interactively collapsed or expanded.
pm_defaultActivityAllowedBarDragModes	number (see enum ActivityBarDragModes)	**Optional, default: ActivityBarDragModes.DragHorizontally** – This option holds the default for the attribute PM_AllowedBarDragModes of Activity objects.
pm_defaultActivityBarHeight	number (≥ 0, ≤ 1000)	**Optional, default: 22** – Default height of the activity bars in pixels. See also Activity.PM_BarHeight.
pm_defaultActivityBarSelectable	boolean	**Optional, default: true** – This option holds the default for the attribute PM_BarSelectable of Activity objects.
pm_defaultActivityBarShape	number (see enum ActivityBarShape)	**Optional, default: Regular** – This option defines which shape should be used by default for the visualization of activity bars.
pm_defaultActivityCollapsedRowDesign	number (see enum RowDesigns)	**Optional, default: 11** – This option holds the default for the attribute PM_CollapsedRowDesign of Activity objects.
pm_defaultActivityConstraintSymbolColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: "#646464"** – Specifies the color used by default for the symbols visualizing the constraint dates (EarliestStart/End, LatestStart/End, MustStart/EndOn).
pm_defaultActivityExpandedRowDesign	number (see enum RowDesigns)	**Optional, default: 11** – This option holds the default for the attribute PM_ExpandedRowDesign of Activity objects.
pm_defaultActivityMinimumRowHeight	number	**Optional, default: 42** – Default minimum height of the activity rows in

Widget Option Name	Type	Description
		pixels. See also Activity.PM_MinimumRowHeight.
pm_defaultActivityProgressBackgroundColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: "transparent" –** Color for the background of the progress bar region for activities.
pm_defaultActivityRowCollapsible	boolean	**Optional, default: true** – This option holds the default for the attribute PM_RowCollapsible of Activity objects.
pm_defaultActivityRowSelectable	boolean	**Optional, default: true** – This option holds the default for the attribute PM_RowSelectable of Activity objects
pm_defaultActivitySnapTargetsForEnd	number (see enum SnapTargets)	**Optional, default: 8** – This option holds the default for the attribute PM_SnapTargetsForEnd of Activity objects
pm_defaultActivitySnapTargetsForStart	number (see enum SnapTargets)	**Optional, default: 8** – This option holds the default for the attribute PM_SnapTargetsForStart of Activity objects.
pm_defaultActivityTableRowDefinitionID	string	**Optional, default: null** – ID of a TableRowDefinition object that will be used when an activity object has set the property PM_TableRowDefinitionID to "".
pm_defaultAllocationAllowedBarDragModes	number (see enum AllocationBarDragModes)	**Optional, default: AllocationBarDragModes.DragAutoHorOrVer** – This option holds the default for the attribute PM_AllowedBarDragModes of Allocation objects.
pm_defaultAllocationBarHeight	number (≥ 0, ≤ 1000)	**Optional, default: 22** – Default height of the allocation bars in pixels. See also Allocation.PM_BarHeight.
pm_defaultAllocationBarSelectable	boolean	**Optional, default: true** – This option holds the default for the attribute PM_BarSelectable of Allocation objects.
pm_defaultAllocationBarShape	number (see enum AllocationBarShape)	**Optional, default: Regular** – This option defines which shape should be used by default for the visualization of allocation bars.
pm_defaultAllocationConstraintSymbolColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: "#646464"** – Specifies the color used by default for the symbols visualizing the constraint dates (EarliestStart/End, LatestStart/End, MustStart/EndOn).

Widget Option Name	Type	Description
pm_defaultAllocationProgressBackgroundColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: "transparent"** – Color for the background of the progress bar region for allocations.
pm_defaultAllocationSnapTargetsForEnd	number (see enum SnapTargets)	**Optional, default: 11** – This option holds the default for the attribute PM_SnapTargetsForEnd of Allocation objects.
pm_defaultAllocationSnapTargetsForStart	number (see enum SnapTargets)	**Optional, default: 11** – This option holds the default for the attribute PM_SnapTargetsForStart of Allocation objects.
(Deprecated! Please see pm_defaultActivityAllowedBarDragModes) **pm_defaultAllowedActivityBarDragModes**	number (see enum ActivityBarDragModes)	**Optional, default: ActivityBarDragModes.DragHorizontally** – This option holds the default for the attribute PM_AllowedBarDragModes of Activity objects.
(Deprecated! Please see pm_defaultAllocationAllowedBarDragModes) **pm_defaultAllowedAllocationBarDragModes**	number (see enum AllocationBarDragModes)	**Optional, default: AllocationBarDragModes.DragAutoHorOrVer** – This option holds the default for the attribute PM_AllowedBarDragModes of Allocation objects.
(Deprecated! Please see pm_defaultEntityAllowedRowDragModes) **pm_defaultAllowedEntityRowDragModes**	number (see enum RowDragModes)	**Optional, default: RowDragModes.DragOutside** – This option holds the default for the attribute PM_AllowedRowDragModes of Entity objects.
pm_defaultEntityAllowedRowDragModes	number (see enum RowDragModes)	**Optional, default: RowDragModes.DragOutside** – This option holds the default for the attribute PM_AllowedRowDragModes of Entity objects.
pm_defaultEntityMinimumRowHeight	number	**Optional, default: 42** – Default minimum height of the entity rows in pixels. See also Entity.PM_MinimumRowHeight.
pm_defaultEntityRowCollapsible	boolean	**Optional, default: true** – This option holds the default for the attribute PM_RowCollapsible of Entity objects.
pm_defaultEntityRowSelectable	boolean	**Optional, default: true** – This option holds the default for the attribute PM_RowSelectable of Entity objects.
pm_defaultEntityTableRowDefinitionID	string	**Optional, default: null** – ID of a TableRowDefinition object that will be used when an entity object has set the property PM_TableRowDefinitionID to "".

Widget Option Name	Type	Description
pm_defaultLinkRoutingType	number (see enum LinkRoutingType)	**Option, default LinkRoutingType.Curved** – This option holds the default for the attribute PM_RoutingType of Links objects.
pm_defaultLinkSelectable	boolean	**Optional, default: false** – This option holds the default for the attribute PM_Selectable of Link objects.
pm_defaultLoadCurvePaneColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: "rgba(43,86,158,0.2)"** – Color for the background of the load curve pane.
pm_defaultResourceCollapsedRowDesign	number (see enum RowDesigns)	**Optional, default: 11** – This option holds the default for the attribute PM_CollapsedRowDesign of Resource objects.
pm_defaultResourceExpandedRowDesign	number (see enum RowDesigns)	**Optional, default: 11** – This option holds the default for the attribute PM_ExpandedRowDesign of Resource objects.
pm_defaultResourceMinimumRowHeight	number	**Optional, default: 42** – Default minimum height of the resource rows in pixels. See also Resource.PM_MinimumRowHeight.
pm_defaultResourceRowCollapsible	boolean	**Optional, default: true** – This option holds the default for the attribute PM_RowCollapsible of Resource objects.
pm_defaultResourceRowSelectable	boolean	**Optional, default: true** – This option holds the default for the attribute PM_RowSelectable of Resource objects.
pm_defaultResourceTableRowDefinitionID	string	**Optional, default: undefined** – ID of a TableRowDefinition object that will be used when a resource object has set the property PM_TableRowDefinitionID to "".
pm_detailedActivityConstraintSymbolsEnabled	boolean	**Optional, default: true** – If set to true, there will be shown different symbols for the constraint dates depending on their constraint types: • EarliestStart:  • LatestStart:  • MustStartOn:  • EarliestEnd:  • LatestEnd:  • MustEndOn: 

Widget Option Name	Type	Description
		Otherwise, a simple down arrow will be shown:  . Please consider to set the option pm_topRowMarginInTimeArea when using detailed symbols.
pm_detailedAllocationConstraintSymbolsEnabled	boolean	**Optional, default: true** – If set to true, there will be shown different symbols for the constraint dates depending on their constraint types: • EarliestStart: • LatestStart:  • MustStartOn:  • EarliestEnd:  • LatestEnd:  • MustEndOn:  Otherwise, a simple down arrow will be shown:  . Please consider to set the option pm_topRowMarginInTimeArea when using detailed symbols.
pm_dragDatesLimitingInteraction	boolean	**Option, default: false** – If set to true, then bars cannot be dragged before the value in the property PM_EarliestDragStart and later than PM_LatestDragEnd, respectively.
pm_entitiesTableHeaderBackgroundColor	string (CSS color value) or Object	**Optional, default: "#646464"** – Specifies a color used to color the background of the entities table header. If a string is given, then the widget uses the color for all view types. If an object is given, then one can specify a color string for each view type and one for not mentioned view types. Example see at option pm_calendarGridColor.
pm_entitiesTableHeaderColumnSeparatorColor	string (CSS color value) or Object	**Optional, default: "white"** – Specifies a color used to color the column separators in the entities table header. If a string is given, then the widget uses the color for all view types. If an object is given, then one can specify a color string for each view type and one for not mentioned view types. Example see at option pm_calendarGridColor.
pm_entitiesTableHeaderHighlightingColor	string (CSS color value)	**Optional, default: "#f7c365"** – Specifies the color to be used during

Widget Option Name	Type	Description
	or Object	the interaction, e.g. to highlight the separation line between two adjacent columns when altering the column widths.
pm_entitiesTableHeaderTextColor	string (CSS color value) or Object	**Optional, default: "white"** – Specifies a color used to color the text in the entities table header. If a string is given, then the widget uses the color for all view types. If an object is given, then one can specify a color string for each view type and one for not mentioned view types. Example see at option pm_calendarGridColor.
pm_entityTableRowDefinitionIDForTitle	string	**Optional, default: value of option pm_defaultEntityTableRowDefinitionID** – ID of a TableRowDefinition object that will be used to show the table title in the entities table. In parallel, it is currently only possible to interactively change the column widths for the TableRowDefinition object that is referenced here.
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.
pm_forcedActivityAllowedBarDragModes	number|null	**Optional, default: null** – If set to a number, then this value overrides any setting in option pm_defaultActivityAllowedBarDragModes, Activity property PM_AllowedBarDragModes, property allowedDragModes in canDrag callback. This option is only important when your application cannot use the other mentioned settings.
pm_forcedAllocationAllowedBarDragModes	number|null	**Optional, default: null** – If set to a number, then this value overrides any setting in option pm_defaultAllocationAllowedBarDragModes, Allocation property PM_AllowedBarDragModes, property allowedDragModes in canDrag callback. This option is only important

Widget Option Name	Type	Description
		when your application cannot use the other mentioned settings.
pm_ignoreCalendarOnAllocationBarInteractions	boolean	**Optional, default: false** – If set to true, then the resource calendar is not taken into account when dragging an allocation bar.
pm_linksVisibleInActivitiesView	boolean	**Option, default: true** – If set to false, the activities view does not show links between activities.
pm_linksVisibleInResourcesView	boolean	**Option, default: false** – If set to true, the resources view shows links between all allocations whose referenced activities are linked.
pm_mainViewAreaVisible	boolean	**Optional, default: true** – When set to false, then in resources view the main view area is invisible. The main view area contains the rows for resources with PM_ViewArea set to Default. If pm_topViewAreaVisible is also false, then the main view area will be visible nevertheless.
pm_maximumSnapDistance	number	**Optional, default: 8** – Maximum distance in pixels of a currently dragged bar to a snap target, within which a dragged bar will get snapped to the snap target.
pm_maximumTopViewAreaHeightRatio	number ($0 \leq n \leq 0.8$)	**Optional, default: 0.5** – This value determines the maximum height of the top view area. If the resources shown in total are higher than the view then a vertical scroll bar is shown. See also option pm_topViewAreaVisible.
pm_pastBackgroundFillColor	string	**Optional, default: "rgba(0,0,0,0.2)"** – This option defines the color of the darkened area between timescale start and value of the option currentDate.
pm_pastBackgroundLineColor	string	**Optional, default: "darkgrey"** – This option defines the color of the date line at the value of the option currentDate.
pm_pastBackgroundLineDashArray	string	**Optional, default: "1,1"** – This option defines the pattern of dashes and gaps for the date line at the value of the option currentDate. For further information, please see https://www.w3.org/TR/SVG11/painting.html#StrokeDasharrayProperty

Widget Option Name	Type	Description
		or https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dasharray. The value "none" indicates that no dashing is used. In this case, the line is drawn solid.
pm_pastBackgroundLineWidth	number	**Optional, default: 1** – This option defines the width of the date line at the value of the option currentDate.
pm_preventDefaultOnContextMenuEvents	boolean	**Option, default: true** – This option determines whether "contextmenu" triggered by the browser's DOM should get a call to preventDefault(). If set to false, then the system default behavior is not prevented (useful for Microsoft Dynamics 365 Finance and Operations).
pm_progressBarWidthCalculationMode	number (see ProgressBarWidthCalculationMode)	**Option, default: ProgressBarWidthCalculationMode.ConsiderWorkingTimesOnly** – This option determines how the widths of the progress bars are calculated. Possible values: • ConsiderWorkingTimesOnly – If this value is used, it is assumed that there is no progress during non-working times. • ConsiderWorkingAndNonworkingTimes – If this value is used, it is assumed that there is progress during both working and non-working times.
pm_releaseDueDateConnectionsVisible	boolean	**Optional, default: false** – If set to true and an activity has set both a ReleaseDate and a DueDate, a line will be displayed to visually connect both dates: 
pm_resourceTableRowDefinitionIDForTitle	string	**Optional, default: value of option pm_defaultResourceTableRowDefinitionID** – ID of a TableRowDefinition object that will be used to show the table title in the resources view. In parallel, it is currently only possible to interactively change the column

Widget Option Name	Type	Description
		widths for the TableRowDefinition object that is referenced here.
pm_scrollToObjectAnimationEnabled	boolean	**Optional, default: false** – If set to true, then scrolling to the target position is animated when using the method scrollToObject.
pm_scrollToObjectHighlightFlashingEnabled	boolean	**Optional, default: true** – Specifies whether or not the frame displayed around an object that has been scrolled to by using the scrollToObject method should flash.
pm_scrollToObjectHighlightingColor	string (CSS color value) or Object	**Optional, default: "#7f0000" –** Color of the frame displayed around an object that has been scrolled to by using the method scrollToObject. If a string is given, then the widget uses the color for all view types. If an object is given, then one can specify a color string for each view type and one for not mentioned view types. Example see at option pm_calendarGridColor.
pm_selectionColor	string (CSS color value) or Object	**Optional, default: "#ffa000"** – Specifies a color used to highlight selected bars, links or table rows. If a string is given, then the widget uses the color for all view types. If an object is given, then one can specify a color string for each view type and one for not mentioned view types. Example see at option pm_calendarGridColor.
pm_splitterHighlightingColor	string (CSS color value) or Object	**Optional, default: "#ffa000"** – Specifies a color used to highlight the splitters when a splitter is dragged. This refers to the splitters between the table or entities table and the Gantt area. If a string is given, then the widget uses the color for all view types. If an object is given, then one can specify a color string for each view type and one for not mentioned view types. Example see at option pm_calendarGridColor.
pm_subRowDistanceInTimeArea	number	**Optional, default: 5** – Vertical distance between two bars in pixels. See also pm_topRowMarginInTimeArea and pm_bottomRowMarginInTimeArea.

Widget Option Name	Type	Description
		Please have in mind that symbols are drawn inside this distance.
pm_suitableResourcesOverlayColor	string (CSS color value)	**Optional, default: "transparent"** – This option determines the color that is added to resource rows that are mentioned in the allocation property SuitableResourceIDs when dragging. See option pm_unsuitableResourcesOverlayColor.
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_tableHeaderBackgroundColor	string (CSS color value) or Object	**Optional, default: "#646464"** – Specifies a color used to color the background of the table header of the Gantt diagram. If a string is given, then the widget uses the color for all view types. If an object is given, then one can specify a color string for each view type and one for not mentioned view types. Example see at option pm_calendarGridColor.
pm_tableHeaderColumnSeparatorColor	string (CSS color value) or Object	**Optional, default: "white"** – Specifies a color used to color the column separators in the table header of the Gantt diagram. If a string is given, then the widget uses the color for all view types. If an object is given, then one can specify a color string for each view type and one for not mentioned view types. Example see at option pm_calendarGridColor.
pm_tableHeaderHighlightingColor	string (CSS color value) or Object	**Optional, default: "#f7c365"** – Specifies the color to be used during the interaction, e.g. to highlight the separation line between two adjacent columns when altering the column widths.
pm_tableHeaderTextColor	string (CSS color value) or Object	**Optional, default: "white"** – Specifies a color used to color the text in the table header of the Gantt diagram. If a

Widget Option Name	Type	Description
		string is given, then the widget uses the color for all view types. If an object is given, then one can specify a color string for each view type and one for not mentioned view types. Example see at option pm_calendarGridColor.
pm_timeAreaBackgroundColor	string (CSS color value) or Object	**Optional, default: "white"** – Specifies a color used to color the background of the time area. If a string is given, then the widget uses the color for all view types. If an object is given, then one can specify a color string for each view type and one for not mentioned view types. Example see at option pm_calendarGridColor.
pm_timeAreaPanningMode	number (see enum PanningMode)	**Optional, default: 3** – Specifies, how the widget reacts to user interactions inside the empty space of the time area. **Note:** When panning with the mouse, this option is only considered if the multipleSelectionEnabled option is set to false.
pm_timescaleBackgroundColor	string (CSS color value) or Object	**Optional, default: "#646464"** – Specifies a color used to color the background of the timescale. If a string is given, then the widget uses the color for all view types. If an object is given, then one can specify a color string for each view type and one for not mentioned view types. Example see at option pm_calendarGridColor.
pm_timescaleHighlightingColor	string (CSS color value) or Object	**Optional, default: "#f7c365"** – Specifies the color to be used during the interaction on the timescale, e.g. to highlight the time period under the mouse cursor.
pm_timescaleInteractionsEnabled	boolean	**Optional, default: true** – If set to false, the user cannot interact with the timescale. This means that the smart navigation mechanism and the mouse wheel functionality for spreading or compressing the time area are disabled. Nevertheless, the interactive horizontal panning of the time area still works.

Widget Option Name	Type	Description
pm_timescaleTextColor	string (CSS color value) or Object	**Optional, default: "white"** – Specifies a color used to color the text in the timescale. If a string is given, then the widget uses the color for all view types. If an object is given, then one can specify a color string for each view type and one for not mentioned view types. Example see at option pm_calendarGridColor.
pm_timescaleTickColor	string (CSS color value) or Object	**Optional, default: "white"** – Specifies a color used to color the ticks in the timescale. If a string is given, then the widget uses the color for all view types. If an object is given, then one can specify a color string for each view type and one for not mentioned view types. Example see at option pm_calendarGridColor.
pm_timescaleWeekendBackgroundColor	String (CSS color value) or Object	**Optional, default: "#888888"** – Specifies a color used to color the background of the weekend cells of the timescale. If a string is given, then the widget uses the color for all view types. If an object is given, then one can specify a color string for each view type and one for not mentioned view types. Example see at option pm_calendarGridColor.
pm_topBarSymbolsVisible	boolean	**Optional, default: true** – If set to false, then no symbols are shown at the top left and top right of allocation bars and activity bars.
pm_topRowMarginInTimeArea	number	**Optional, default: 10** – Height of the margin between the top row border and bars in pixels. See also pm_bottomRowMarginInTimeArea and pm_subRowDistanceInTimeArea. Please have in mind that symbols are drawn inside this margin. When one of the options pm_detailedActivity/ AllocationConstraintSymbolsEnabled is set to true, then the value here should be set to a value of 15 or above in order to avoid an vertical overlap.
pm_topViewAreaVisible	boolean	**Optional, default: false** – If set to true, then resources in the resources view are shown in a separate top view area, that have the attribute PM_ViewArea set to Top. See also

Widget Option Name	Type	Description
		options pm_mainViewAreaVisible and pm_maximumTopViewArea-HeightRatio.
pm_unsuitableResourcesOverlayColor	string (CSS color value)	**Optional, default: "rgba(0,0,0,0.2)"** – This option determines the color that is added to resource rows that are NOT mentioned in the allocation property SuitableResourceIDs when dragging. See option pm_suitableResourcesOverlayColor.
start	Date|string	**Required** – Start of the considered time area.
tableViewWidth	number	**Optional, default: null (means table width)** – This setting defines the width of the table view when it becomes visible initially.
(Deprecated!) tableWidth	number	**Optional, default: undefined** – Not recommended when using TableRowDefinition objects! This setting defines the width of the table. When TableRowDefinition objects are not used, 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 also onDetermineColumnDefinitions).
timescaleNavigationMode	number (see TimescaleNavigationMode)	**Optional, default: 0** – Mode of navigation in the timescale.
timeStepUnit	string (one of "second", "minute", "hour", "day")	**Optional, default: "second"** – Unit for time steps on horizontal drag interactions of bars. See timeStepUnitFactor. **Attention!** Currently, the dates of the bars as well as the dates in the calendar must not be defined more finely than this unit together with the timeStepUnitFactor indicate. Otherwise, unexpected jumps will occur when moving bars.
timeStepUnitFactor	number (>= 1)	**Optional, default: 1** – Number of units for a single time step on horizontal drag interactions of bars. See timeStepUnit. Integer values are recommended.

Widget Option Name	Type	Description
		Attention! Currently, the dates of the bars as well as the dates in the calendar must not be defined more finely than this factor together with the timeStepUnit indicate. Otherwise, unexpected jumps will occur when moving bars.
timeZone	string	**Optional, default: undefined** – This option determines the time zone for which dates are shown in the timescale. If set to undefined, then local time zone of the browser is used. When using this option, it is necessary to load the JavaScript libraries Moment.js and Moment Timezone at application startup. The possible values are all the ones that Moment Timezone knows (based on IANA TimeZone database, e.g. "Europe/Berlin"", see also https://en.wikipedia.org/wiki/List_of_tz_database_time_zones for a detailed list of allowed zone names).
titleText	string	**Optional, default: undefined** – This text will be shown in the table header. It will appear only in one the following two cases: 1. If using the TableRowDefinition objects for defining the table and the property pm_activityTableRowDefinitionIDForTitle or pm_resourceTableRowDefinitionIDForTitle appropriate to the corresponding view type is **not** set. or 2. If using the deprecated callback onDetermineComumnDefinitions and there additionally the flag hasColumnTitles is set to **false** in the callback (see there).
version	string	**Read only** – This option holds the version number of the widget set by NETRONIC. Usually it is formatted using the semantic versioning format

Widget Option Name	Type	Description
		"MAJOR.MINOR.PATCH" (see also https://semver.org/).
viewType	number (see enum ViewType)	**Optional, default: ViewType.Activities** – This option determines the type of view that is shown.
visualZoomFactor	number	**Optional, default: 1.0** – Factor used to zoom in (>1) and out (<1) the whole widget. Values <= 0 will be ignored.
weekNumbering	string|null (currently possible values: "ISO8601", "USA")	**Optional, default: undefined** – This option determines the first day of the week (ISO8601: Monday, USA: Sunday) and the week numbering scheme. If set to undefined, then the implicit setting of the option "locale" is used.
workDate	Date|string|null	**Optional, default: null** – Date on which the work date line will be displayed. Please note: The work date line is a simple line only. There are no further properties like color, line width, or line pattern to be set. If such properties are needed, then a date line should be used.
workDateLineCaption	string	**Optional, default: ""** – Text to be displayed at the work date line.
worldViewExtent	number	**Optional, default: 150** – Defines the extent of the world view in pixels.
worldViewPosition	number (see enum WorldViewPosition)	**Optional, default: Bottom** – Defines the position of the world view within the widget.
worldViewVisible	boolean	**Optional, default: false –** If set to true, then a world view is visible at the bottom of the Gantt chart. Only the table row background colors and bar colors are shown. Also date lines and separation lines between left table, timescale, top view area are shown. Additionally, selections are shown and frames for the visible parts shown in the widget (separately for table and time area). These frames can also be dragged to modify the visible parts.

5.2 Callbacks

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

Callback Name	Type	Description

canDrag	Function	**Optional, default: undefined** – This function is called when the user is moving the mouse cursor over an activity/allocation or touches an activity/allocation with a finger.
		
		Profile:
		`function (args)`
		`args = {`
		`  "objectType" : ObjectType,`
		`  "object" : Object,`
		`  "visualType" : VisualType,`
		`  "entry"[1] : AllocationEntry,`
		`  "entryIndex"[1] : number,`
		`  "allowedDragModes" : ActivityBarDragModes`
		`  |AllocationBarDragModes, //[in/out]`
		`  "selectedObjects" : Object[]|undefined`
		`}`
		
		If the application sets args.allowedDragModes to None, then no dragging will be possible. The same is possible by setting option pm_forcedActivity/AllocationAllowedBarDragModes to None. On input, args.allowedDragModes contains the value of the property PM_Allowed(Row/Bar)DragModes of the object to drag.
		
		If the option multipleBarDraggingEnabled is set to true and more than one bar is selected, the property selectedObjects will contains all selected objects, so that the application can determine the value for allowedDragModes.
		
		This callback is called only once every time when the mouse enters the visual representation of the object (bar).

canSelect	Function	**Optional, default: undefined** – This function is called when the user moves the mouse cursor onto the graphical representation of an object.
		
		Profile:
		`function (args)`
		`args = {`
		`  "objectType" : ObjectType,`
		`  "object" : Object,`
		`  "visualType" : VisualType,`
		`  "otherSelectedObjects" : Object[],`
		`  "event" : DOMEvent,`
		`  "cancel" : boolean [out]`
		`}`

[1] Available only if objectType == ObjectType.Allocation or if visualType == VisualType.PeriodHighlighter.

Callback 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.

onClicked	Function	**Optional, default: undefined** – This function is called when an object is clicked by the user.
		
		Profile:
		`function (args)`
		`args = {`
		`"objectType" : ObjectType,`
		`"object" : Object,`
		`"visualType" : VisualType,`
		`"date" : Date,`
		`"entry"[1] : AllocationEntry|PeriodHighlighterEntry,`
		`"entryIndex"[1] : number,`
		`"curve" : Object // Only available when clicked on a curve; the "object" parameter will then hold the corresponding resource`
		`}`
		
		On time area and timescale, the object is null.

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: `function ()`
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

Callback Name	Type	Description

		• 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" : CollapseState,`
		`"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 or activity row.
		
		Profile:
		`function (args)`
		`args = {`
		`"objectType" : ObjectType,`
		`"object" : Object,`
		`"newCollapseState" : CollapseState,`
		`"resource" : Resource|undefined,`
		`"promise" : Promise [out]`
		`}`
		
		The property "resource" is only set, when the object is not the resource itself.
		
		The application can update the property PM_CurveCollapseState of the object if needed.
		
		If the application sets the promise attribute, then the update of the DOM is delayed until the promise is resolved.

(Deprecated!) onDetermineColumnDefinitions	Function	**Optional, default: undefined** – Please use object TableRowDefinition instead for same purpose.
		
		This function is called to determine the definitions of the table columns.
		
		Profile:
		`function (args)`
		`args = {`
		`"tableType" : TableType`
		`"level" : number,`
		`"objectType" : ObjectType,`
		`"object" : Object,`
		`"columns" : Object[], //[in/out]`
		`"hasColumnTitles" : boolean //[in/out]`
		`}`

Callback Name	Type	Description

		The content of args.columns can be changed or replaced. For each column there is an object as follows:
		
		`{`
		`  "initialWidth" : number, /*in pixels*/`
		`  "horizontalTextAlignment" :`
		`  HorizontalAlignment,`
		`  "textSource" : string /*property name*/,`
		`  "title" : string,`
		`  "wrapMode" : TextWrapMode`
		`}`
		
		If args.hasColumnTitles is set to true, the values of the "title" property of the column objects are displayed as table column headers with the option of interactively resizing the column widths. Only the table cells of rows that apply the definition will alter their sizes.
		
		This means that a maximum of one definition per table type can have the property "hasColumnTitles" set to true. Otherwise, the interactive resizing may have undesirable effects.
		
		Attention: For all rows that should get the same column definition, one and the same array object should be returned by the property "columns"! Otherwise it will not be possible to properly resize column widths interactively!
		
		If the maximum sum of all column widths per row is less than the table width specified by the "tableWidth" option, then the last cells of each row will be enlarged if necessary.

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"[1] : AllocationEntry|`
		`PeriodHighlighterEntry,`
		`  "entryIndex"[1] : number`
		`}`
		
		On time area and timescale, the object is null.

onDrag	Function	**Optional, default: undefined** – This function is called when the user drags an activity, allocation or allocation entry (called anew on every new

Callback Name	Type	Description

		move of the mouse/finger). If args.dropAllowed is set to false on return of the callback, then a forbidden cursor is shown within the widget and a drop will be ignored.
		
		If args.cancel is set to true, then the drag action will be canceled.
		
		If an allocation is dragged, then the additional property newRowObjectIsSuitableResource gives the information whether the dragged object is over a suitable resource. Then the application can transfer the value to the property dropAllowed if wishful.
		
		Profile:
		`function (args)`
		`args = {`
		`"objectType" : ObjectType,`
		`"object" : Object,`
		`"visualType" : VisualType,`
		`"entry"[1] : AllocationEntry,`
		`"entryIndex"[1] : number,`
		`"dragMode" : ActivityBarDragModes`
		`|AllocationBarDragModes,`
		`"newRowObjectType" : ObjectType,`
		`"newRowObject" : Object,`
		`"newRowObjectIsSuitableResource" : boolean,`
		`"newStart" : Date, // not for date lines`
		`"newEnd" : Date, // not for date lines`
		`"newDate" : Date, // only for date lines`
		`"dropAllowed" : boolean [out],`
		`"cancel" : boolean [out]`
		`}`

onDragEnd		**Optional, default: undefined** – This function is called when the user ends dragging an activity, allocation, allocation entry, or entity (please check args.objectType!) even when dropping is not allowed on the new row.
		
		Profile:
		`function (args)`
		`args = {`
		`"objectType" : ObjectType,`
		`"object" : Object,`
		`"visualType" : VisualType,`
		`"entry"[1] : AllocationEntry,`
		`"entryIndex"[1] : number,`
		`"dragMode" : ActivityBarDragModes`
		`|AllocationBarDragModes`
		`}`

onDragStart	Function	**Optional, default: undefined** – This function is called when the user starts to drag an activity, allocation, allocation entry, or entity (please check args.objectType!). If args.cancel is set to true, then the drag action will be canceled. Profile: `function (args)`

Callback Name	Type	Description

		args = {
		"objectType" : ObjectType,
		"object" : Object,
		"visualType" : VisualType,
		"entry"[1] : AllocationEntry,
		"entryIndex"[1] : number,
		"dragMode" : ActivityBarDragModes
		|AllocationBarDragModes,
		"cancel" : boolean [out]
		}

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"[1] : AllocationEntry,
		"entryIndex"[1] : number,
		"dragMode" : ActivityBarDragModes
		|AllocationBarDragModes,
		"newRowObjectType" : ObjectType,
		"newRowObject" : Object,
		"newStart" : Date,
		"newEnd" : Date,
		"cancel" : boolean, // [out]
		"promise" : Promise, // [out]
		"workingTimeDistance" : number, /*in milliseconds*/
		"coupledObjects": Allocation[]|Activity[],
		"startsAndEndsOfCoupledObjects" : Object[]
		}
		
		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.
		
		When the option multipleBarDraggingEnabled is set to true and more than one object was dragged, then the properties coupledObjects

Callback Name	Type	Description
		and startsAndEndsOfCoupledObjects are set. The latter one contains objects of the form: { object : Allocation|Activity, newStart : Date, newEnd : Date }. Remark: If one of the properties newStart or newEnd hat a value of null, then the user dragged this object outside of the visible time area and there is no working time in the calendar to calculate the appropriate date.

onSelectionChanged	Function	**Optional, default: undefined** – This function is called when the user selects/deselects an object solely or in addition. The property "selectedObjects" holds the new selection completely and can be changed by the application, while the previously selected objects (if any) are contained in the property "previouslySelectedObjects".
		
		Profile:
		`function (args)`
		`args = {`
		`"objectType" : ObjectType|0,`
		`"object" : Object|null,`
		`"selectedObjects" : Object[], //[in/out]`
		`"visualType" : VisualType,`
		`"previouslySelectedObjects" : Object[]|null,`
		`"previouslySelectedObjectsType" : ObjectType|null,`
		`"event": DOMEvent,`
		`"cancel": Boolean /* [in/out], Default: false */`
		`}`

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 (only when shown as separate bars instead of allocation bars), links, timescale, empty time area, and period highlighters.
		
		Profile:
		`function (args)`
		`args = {`
		`"objectType" : ObjectType,`
		`"object" : Object,`
		`"visualType" : VisualType,`
		`"date" : Date,`
		`"event" : DOMEvent or jQuery.Event,`
		`"entry"[1] : AllocationEntry,`
		`"entryIndex"[1] : number,`
		`"promise" : Promise //[out]`
		`}`

Callback Name	Type	Description

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. Possible objects are resources, activities, allocations, links, and period highlighters.
		
		Profile:
		`function (args)`
		`args = {`
		`"objectType" : ObjectType,`
		`"object" : Object,`
		`"visualType" : VisualType,`
		`"event" : DOMEvent,`
		`"date"[2] : Date // date at mouse cursor,`
		`"capacity"[2] : number,`
		`"load"[2] : number,`
		`"singleLoads"[3] : Object,`
		`"entry"[1] : AllocationEntry| PeriodHighlighterEntry,`
		`"entryIndex"[1] : number,`
		`"innerHTML"[4] : string //[in/out]`
		`"tooltipTemplateID" : string // [in/out]`
		`}`

onTableCellDefinitionWidthChanged	Function	**Optional, default: undefined** – If set, then this function is called when the user has changed the width of a table column. This callback will only work, when the table columns were defined by TableRowDefinition objects. You then are able to update the cell definition inside of the appropriate TableRowDefinition object e.g. for gaining persistency inside the application.
		
		Profile:
		`function (args)`
		`args = {`
		`"tableType" : TableType,`
		`"tableRowDefinition" : Object,`
		`"cellIndex": number,`
		`"newWidth": number,`
		`"oldWidth" : number`
		`}`

onTimeAreaViewParametersChanged	Function	**Optional, default: undefined** – This function is called when the visible time area changes either by changing the visible start or by changing the

[2] Available only if objectType == ObjectType.Resource and the mouse cursor hovers over a curve area.
[3] 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.
[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 elements. **Line breaks** can be inserted by adding a
 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 ">").

Callback Name	Type	Description

		resolution. There is an internal delay of 500 milliseconds.
		
		Profile:
		`function (args)`
		`args = {`
		`  "scrollOffset" : number, /* in pixels */`
		`  "width" : number, /* in pixels */`
		`  "start" : Date,`
		`  "end" : Date,`
		`  "timeResolutionUnit" : string (possible`
		`  values: "seconds"/"minutes"/"hours"/`
		`  "days"),`
		`  "timeResolutionUnitCount" : number`
		`}`
		
		The values of the properties “start” and “end” can be used in the method fitTimeAreaIntoView to restore the current view at a later time. Alternatively the he values of the properties "timeResolutionUnit" and "timeResolutionUnitCount" can be used for the method setTimeResolutionForView.

onVerticalScrollOffsetChanged	Function	**Optional, default: undefined** – This function is called when the visible area is scrolled vertically or when the row object visible at top has changed. There is an internal delay of 500 milliseconds.
		
		Profile:
		`function (args)`
		`args = {`
		`  "tableType" : TableType,`
		`  "scrollOffset" : number, /* in pixels */`
		`  "rowObjectTypeAtTop" : ObjectType,`
		`  "rowObjectAtTop" : Object`
		`}`

visibilityFilter	Function	**Optional, default: undefined** – This function is called in order to hide objects. At the moment the callback is triggered only for resources, allocations, and activities, but it is planned to extend the number of object types. The result has to be set in the property named "result": true means visible and false means invisible.
		
		Profile:
		`function (args)`
		`args = {`
		`  "objectType" : ObjectType,`
		`  "object" : Object,`
		`  "result" : boolean /* [out], Default: true */`
		`}`

5.3 Methods

The following methods are callable in two ways:

- $("#ganttDiv").nXYZWidget("methodName", param1, param2, …)
- $("#ganttDiv").nXYZWidget("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
about	-	-	Opens a popup dialog that shows the licenses of all libraries used. The dialog can be made visible also directly by the user by pressing Ctrl+Alt+Shift+F12.
addActivities	-	activities : Activity[]	Adds activities. [5]
addAllocations	-	allocations : Allocation[]	Adds allocations. [5]
addCalendars	-	calendars : Calendar[]	Adds calendars. [5]
addCurves	-	curves : Curve[]	Adds curves. [5]
addDateLines	-	dateLines : DateLine[]	Adds date lines. [5]
addEntities	-	entities : Entity[]	Adds entities. [5]
addLinks	-	links : Link[]	Adds links. [5]
addPeriodHighlighters	-	periodHighlighters : PeriodHighlighter[]	Adds period highlighters. [5]
addResources	-	resources : Resource[]	Adds resources. [5]
addSymbols	-	symbols : Symbol[]	Adds symbols. [5]
addTableRowDefinitions	-	tableRowDefinitions : TableRowDefinition[]	Adds table row definitions. [5]
addTooltipTemplates	-	tooltipTemplates : TooltipTemplate[]	Adds tooltip templates. [5]
addWorkingTime	Date	calendarID : number, start : Date|string, workingTime : number	Add a working time given in milliseconds to a date and returns a new date object with the calculated date.
calculateWorkingTime	number	calendarID : number, start : Date|string, end : Date|string	Calculates the working time of a time period given by a start and an end date. The working time returned is given in milliseconds.
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

[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 timer which calls the method "render" automatically, but this eventually leads to flickering within the Widget's visualization.

Method Name	Result Type	Parameters	Description
			replaced by start and end date of the complete time area.

getSelectedObjects	Object	-	Gets all currently selected objects. The result is an object with the following properties:
			
			{
			objects : Object[],
			objectType : ObjectType|undefined,
			visualType : VisualType|undefined
			}
			
			When no objects are currently selected, then the array is empty and the type properties are set to undefined.
			
			See also selectObjects method.

removeActivities	-	activitiesOrIDs : string[]|Activity[]	Removes activities. [5]
removeAll	-	-	Removes all objects.
removeAllocations	-	allocationsOrIDs : string[]|Allocation[]	Removes allocations. [5]
removeCalendars	-	calendarsOrIDs : string[]|Calendar[]	Removes calendars. [5]
removeCurves	-	curvesOrIDs : string[]|Curve[]	Removes curves. Resources have to be unused to be removable. [5]
removeDateLines	-	dateLinesOrIDs : string[]|DateLine[]	Removes date lines. [5]
removeEntities	-	entitiesOrIDs : string[]|Entity[]	Removes entities. [5]
removeLinks	-	linksOrIDs : string[]|Link[]	Removes links. [5]
removePeriodHighlighters	-	periodHighlightersOrIDs : PeriodHighlighter[]|string[]	Removes period highlighters. [5]
removeResources	-	resourcesOrIDs : string[]|Resource[]	Removes resources. [5]
removeSymbols	-	symbolsOrIDs : string[]|Symbol[]	Removes symbols. [5]
removeTableRowDefinitions	-	tableRowDefinitionsOrIDs : string[]|TableRowDefinition[]	Removes table row definitions. [5]
removeTooltipTemplates	-	tooltipTemplatesOrIDs : string[]|TooltipTemplate[]	Removes tooltip templates. [5]

Method Name	Result Type	Parameters	Description
render	-	-	Refreshes the view after changes to data objects. When the application forgets to call this method, then it is called automatically when the application goes idle.
scrollToDate	-	Date	Scrolls to the given date.
scrollToObject	-	objectType: ObjectType, object: object, targetPositionInView : TargetPositions, highlightingEnabled : boolean	Scrolls to the object (activity/allocation/entity/resource). If the object is not visible, the corresponding rows are expanded automatically. The third and the fourth parameter are optional. targetPositionInView (default is Necessary) determines the position of the object in the view after scrolling to it. If highlightingEnabled is set to true (default), then a (eventually blinking) frame is shown until another method is used or a user interaction takes place. See also options pm_scrollToObject-HighlightingColor, pm_scrollTo-ObjectHighlightFlashingEnabled, pm_scrollToObjectAnimation-Enabled.
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. See also getSelectedObjects method.

Method Name	Result Type	Parameters	Description
setTimeResolutionForView	-	unit : string ("seconds", "minutes", "hours", "days"), unitCount : number|undefined, start : Date|undefined	Sets the resolution in the time area view. If unitCount is undefined, then 1 is used. If start is undefined, then the current visible start is used.
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.
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.
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.
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.
updateDateLines	-	dateLines : DateLine[], updateMode : UpdateModes	Updates date lines. Allowed changes are modification of all attributes besides ID. [5] updateMode is optional. See enum UpdateModes in the Enumerations chapter for details.
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.

Method Name	Result Type	Parameters	Description
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.
updatePeriodHighlighters	-	periodHighlighters : PeriodHighlighter[], updateMode : UpdateModes	Updates period highlighters. [5] updateMode is optional. See enum UpdateModes in the Enumerations chapter for details.
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.
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.
updateTableRowDefinitions	-	tableRowDefinitions : TableRowDefinition[], updateMode : UpdateModes	Updates table row definitions. Allowed changes are modification of all attributes besides ID. [5] updateMode is optional. See enum UpdateModes in the Enumerations chapter for details.
updateTooltipTemplates	-	tooltipTemplates: TooltipTemplate[], updateMode : UpdateModes	Updates tooltip templates. Allowed changes are modification of all attributes besides ID. [5] updateMode is optional. See enum UpdateModes in the Enumerations chapter for details.

6 Enumerations

The following enumerations are provided:

6.1 ActivityBarDragModes

```
netronic.nVSW.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,
};
```

6.2 ActivityBarShape

```
netronic.nVSW.ActivityBarShape = {
   Regular: 0,
   Summary: 1,
   Diamond: 2,
   Rectangle: 3,
};
```

6.3 AllocationBarDragModes

```
netronic.nVSW.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,
};
```

6.4 AllocationBarShape

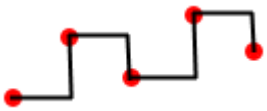
```
netronic.nVSW.AllocationBarShape = {
   Regular: 0,
   Summary: 1,       // Only to be used if the allocation has only one
                     // entry
   Rectangle: 3,
};
```

6.5 CollapseState

```
netronic.nVSW.CollapseState = {
   Unchanged: -1,
   Expanded: 0,
   Collapsed: 1
};
```

6.6 CurveInterpolationType

```
netronic.nVSW.CurveInterpolationType = {
   StepAfter: 0,
```



```
   Linear: 1,
```

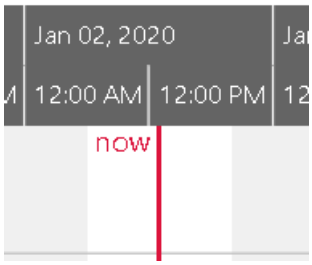
```
};
```

6.7 CurveType

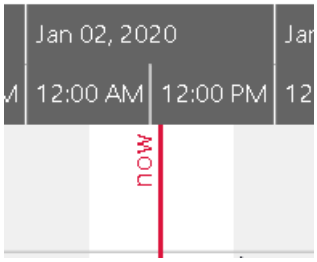
```
netronic.nVSW.CurveType = {
   PointCurve: 0,
   CurveStack: 3,
   CurveList: 4
};
```

6.8 DateLineCaptionOrientation

```
netronic.nVSW.DateLineCaptionOrientation = {
   Horizontal: 1,
```




```
   Vertical: 2
```

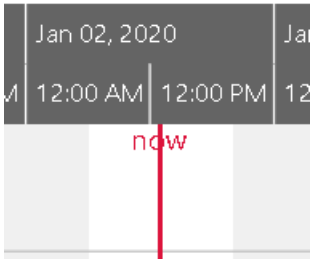



```
};
```

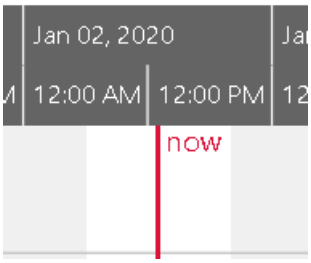
6.9 DateLineCaptionPosition

```
netronic.nVSW.DateLineCaptionPosition = {
   Left: 1,
```

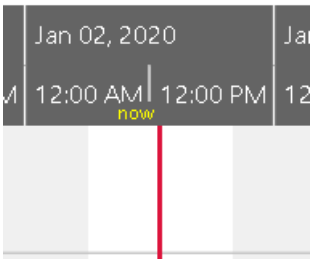
```
   Center: 2,
```



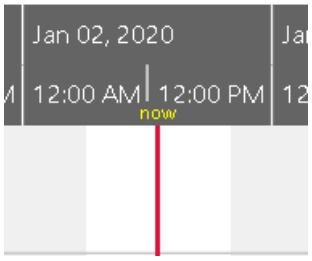

```
   Right: 4,
```



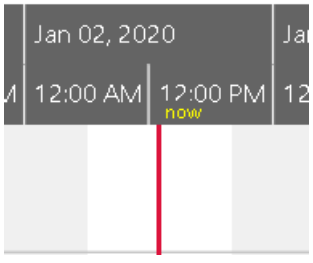

```
   TopLeft: 9,

   // inside
   // timescale
   // area
```




```
   TopCenter: 10,

   // inside
   // timescale
   // area
```




```
   TopRight: 12

   // inside
   // timescale
   // area
```




```
};
```

6.10 DateLineGridModes

```
netronic.nVSW.DateLineGridModes = {
   None: 0,
   Auto: 1,
   Weekly: 2,
```

```
   Daily: 4
};
```

6.11 HorizontalAlignment

```
netronic.nVSW.HorizontalAlignment = {
   Left: 0,
   Center: 1,
   Right: 2
};
```

6.12 LinkRoutingType

```
netronic.nVSW.LinkRoutingType = {
   Curved: 1,
   Orthogonal: 2
};
```

6.13 ObjectType

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

6.14 PanningMode

```
netronic.nVSW.PanningMode = {
   None: 0,
   HorizontallyOnly: 1,
   VerticallyOnly: 2,
   HorAndVer: 3,
   AutoHorOrVer: 4,
};
```

6.15 ProgressBarWidthCalculationMode

```
netronic.nVSW.ProgressBarWidthCalculationMode = {
   ConsiderWorkingTimesOnly: 0,
   ConsiderWorkingAndNonworkingTimes: 1,
};
```

6.16 RowDesigns

```
netronic.nVSW.RowDesigns = {

   // Note: flags!
   // These values can be combined by using bitwise OR operators.
```

```
   Empty: 0,
   Bars: 1,                              // Shows bars assigned to row object directly
   Optimized: 2,                         // Shows all bars without vertical overlapping
   BarsInHiddenDescendantRows: 4,        // Shows bars of other hidden descendant rows
   CalendarGrid: 8                       // Shows calendar grid of row object
};
```

6.17 RowDragModes

```
netronic.nVSW.RowDragModes = {

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

   None: 0,
   DragOutside: 32,
};
```

6.18 SnapTargets

```
netronic.nVSW.SnapTargets = {

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

   None: 0,
   Start: 1, // only valid for bars representing allocations
   End: 2, // only valid for bars representing allocations
   DateLines: 4,
   CalendarGrids: 8,
   DateLineGrids: 16
};
```

6.19 TableType

```
netronic.nVSW.TableType = {
   Gantt: 0,
   Entities: 1
};
```

6.20 TargetPositions

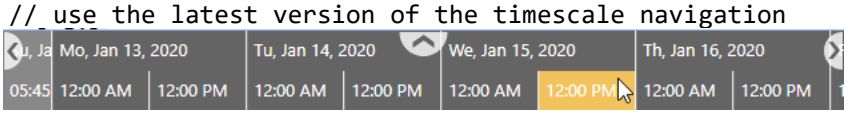
```
netronic.nVSW.TargetPositions = {

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

   Necessary: 0,
   Left: 1,
   HCenter: 2,
   Right: 4,
   Top: 8,
   VCenter: 16,
   Bottom: 32
}
```

6.21 TextWrapMode

```
netronic.nVSW.TextWrapMode = {
   None: 0, // no wrapping at all
   Line: 1, // text is wrapped at \n
};
```

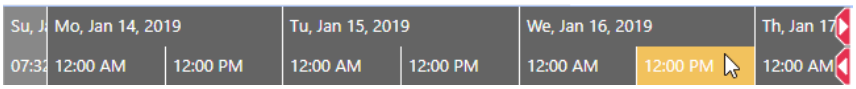
6.22 TimescaleNavigationMode

```
netronic.nVSW.TimescaleNavigationMode = {
   Latest: 0,          // use the latest version of the timescale navigation
```



- A click onto the left and right button scrolls the chart sideward by the width of the view.
- A click onto the up button reduces the timescale resolution.
- A click onto a highlighted period (see orange area) fits this period completely into the view.
- Use the mouse wheel for increasing and reducing the timescale resolution.

```
   LegacyVersion1: 1,
```



- A click onto the left and right button scrolls the chart sideward by the widths of one unit in the upper timescale ribbon
- A click onto a highlighted period (see orange area) fits this period completely into the view.
- Use the mouse wheel for increasing and reducing the timescale resolution.

```
};
```

6.23 TimeType

```
netronic.nVSW.TimeType = {
   WorkingTime: 1,
   NonWorkingTime: 2
};
```

6.24 UpdateModes

```
netronic.nVSW.UpdateModes = {
   UpdateOnly: 0,
   ImplicitAddObjects: 1 // If an object to be updated does not exist,
                         // it will be added automatically.
};
```

6.25 ViewArea

```
netronic.nVSW.ViewArea = {
   Top: -1,
   Default: 0
};
```

6.26 ViewType

```
netronic.nVSW.ViewType = {
   Activities: 0,
   Resources: 1,
   Loads: 2
};
```

6.27 VisualType

```
netronic.nVSW.VisualType = {
   Background: -1,
   Bar: 0,
   Row: 1,
   Curve: 2,
   Link: 3,
   PeriodHighlighter: 4,
   DateLine: 5
};
```

6.28 WorldViewPosition

```
netronic.nVSW.WorldViewPosition = {
   Left: 1,
   Right: 2,
   Top: 3,
   Bottom: 4
}
```