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

Valid for the VSW SE as of version 6.3.5/6.4.0
2022.12.21-15:05

Contents

1   Changelog

Version	Description of changes
6.3.5	PATCH: Method getSelectedObjects did not return selected links.
	PATCH: Row dragging did not work for entities when using option pm_defaultEntityAllowedRowDragModes set to DragVertically only.
	PATCH: When using one of the options for filtering visibility of row objects it could happen, that bars for invisible rows were drawn. Since version 6.3.3.
6.3.4	PATCH: The callback canDrag was triggered too often.
	PATCH: The callback onShowTooltip was not called always, when the mouse cursor left a bar.
	PATCH: When using a bar shape for an allocation different from the default one, this was not always visible on the first allocation entry after updating an allocation object.
	PATCH: Fixed rare exception when positioning links between invisible bars.
6.3.3	PATCH: Performance improvements for several remove… method calls and for artificial links from activity links in resources view.
	PATCH: Fixed not working press gesture directly following a double-click.
	PATCH: Fixed missing cursor icon when hovering an application-defined release or due date symbol of an activity bar.
	PATCH: The library html2canvas was required at start-up of VSW but is only necessary when saving a PDF file and using the PDF options topHTML or bottomHTML.
	Additionally: This document now lists polyfill-library as included resource (see chapter 2.3).
6.3.2	Internal release.
6.3.1	PATCH: Performance improvements for several update… method calls.
	PATCH: The curve values displayed in the tooltips did not exactly match the definition in the curve entries due to JavaScript side effects when adding and subtracting values.
	PATCH: When dragging a row with allowed drag mode DragOnSameLevelOnly target rows are not expanded automatically when the level is below the one of the dragged row.
	PATCH: Period highlighters were not visible on allocation rows in activities view.
	PATCH: Rows were too high, when more than one zero-width bar with same start date existed there.
	PATCH: Fixed exception when calling one of the methods selectObjects and highlightObjects with an allocation object in activities view.
	Additionally: Added missing return values (Promise objects) to the description of

Version	Description of changes
	methods fitTimeAreaIntoView, scrollToDate, scrollToObject in this document.
6.3.0 (See also [release notes](#)!)	MINOR: Activity, entity, and resource rows optionally can be dragged and dropped vertically inside the appropriate table now: • New options pm_defaultActivity/ResourceAllowedRowDragModes. • New options pm_forcedActivity/Entity/ResourceAllowedRowDragModes (suitable for cases where the callback handler of the application for canDrag cannot provide property changes on callback arguments on return). • New property PM_AllowedRowDragModes in Activity and Resource objects. • New enum values DragVertically and DragOnSameLevelOnly for RowDragModes. • New properties in callbackArgs when callbacks onDrag and onDrop are triggered. Furthermore, for this now exists the new enum RowInsertionMode. • Precalculation of sort code value when sort mode is set to be ascending (see new options activity/entity/resourceRowSortMode). MINOR: Sorting now also is possible by using a definable property in objects instead of implementing an appropriate compare callback: • New options activity/allocation/entity/resourceRowSortCodePropertyName and activity/allocation/entity/resourceRowSortMode. • New enum RowSortMode. • New property PM_SortCode for Activity/Allocation/Entity/Resource objects serving as default property for new options above. MINOR: New property PM_BorderDashArray of Activity and Allocation objects. MINOR: New callback options visibilityFilterForActivities/Allocations/Resources/Entities. These ones replace the now deprecated option visibilityFilter for performance reasons. MINOR: New callback options compareActivities/Allocations/Resources/Entities. These ones replace the now deprecated option compareObjects for performance reasons. MINOR: New options defaultValuesFor*Object*Properties with *Object* standing for Activity, ActivityEntry, Allocation, AllocationEntry, Entity, Link, Resource. MINOR: New property entitiesTableViewWidth for callback onTimeAreaViewParametersChanged. MINOR: New option loggingVerboseLevel. MINOR: New property commandCounter for callbacks onLogWarning and onLogError. MINOR: Symbols on date lines are now also placed optimized. See option pm_dateLineCaptionOptimizedPositioningEnabled. MINOR: Automatic recognition of duplicate IDs and cycles in hierarchy when using ParentIDs on adding or updating objects, see callback options onLogWarning and onLogError. PATCH: Fixed issues with cursor icon when hovering draggable splitters, column separators, and date lines.

Version	Description of changes
	PATCH: Fixed internal exception after calling method saveAsPDF.
6.2.8	PATCH: Fixed missing tooltips on curves when using Firefox (since 7.5.3).
	PATCH: Sometimes it was not possible to drag the bar onto the original position when interactively dragging bars with options timeStepUnit and timeStepUnitFactor set to coarser values (since 7.5.2).
	PATCH: It was impossible to gain keyboard focus when using Firefox and clicking or tapping into widget element (since 7.5.3).
	PATCH: When starting a web application from file system, the PDF export failed in whole when a URL was used inside top or bottom HTML strings. Now only a warning is triggered.
	PATCH: Dragging of bars was not always possible when using a hatch pattern.
	PATCH: The callback onShowContextMenu was not triggered on period highlighter entries.
	PATCH: It was not possible to access neither row object properties using #RowObject nor period highlighter properties in tooltip templates assigned to period highlighter entries.
	PATCH: When showing allocation rows in activities view, selecting bars by drawing a rectangle optically selected activity and allocation bars.
	PATCH: When option editable was set to false, you could nevertheless drag entities into the time area. Additionally, changing the option editable at run-time had no effect.
	PATCH: When dragging entities into the time area with options timeStepUnit and timeStepUnitFactor set to coarser values, then the calendar was not continuously considered.
	PATCH: Dragging bars with option timeStepUnit set to "year" did not work at all.
	PATCH: Option pm_barSortModeForOptimizedRowDesign did not work.
	PATCH: When changing option pm_bottomRowMarginInTimeArea, row heights were not updated concerning a currently visible curve pane.
	PATCH: When clicking/tapping onto a curve, sometimes an exception occurred, and sometimes selected elements were not deselected.
	PATCH: Neither the callback onSelectionChanged was triggered nor selected entities were deselected, when the user clicked/tap on the time area background or on curves.
	Attention: When using option pm_dateLineCaptionOptimizedPositioningEnabled, the optimization of caption positions does not work when symbols are used on the date lines! This will be fixed with upcoming version 7.6.0.
6.2.7	PATCH: More performance when using add methods for Activity, Entity, Resource objects.
	PATCH: Fixed additional horizontal scrollbar for top view area in Firefox (since version 6.2.3).

Version	Description of changes
6.2.6	PATCH: Fixed functionality of property PM_BorderColor of Activity objects when entries are existing.
	PATCH: Fixed missing values when evaluating a TooltipTemplate for a tooltip (since version 6.2.3).
	In this document now the number of possible hierarchy levels are explicitly described as being limited to a maximum of approx. 100 (see property ParentID of Activity, Entity, and Resource objects) because of possible performance issues.
6.2.5	PATCH: Fixed false empty triggering of callback onShowTooltip when moving the mouse pointer from one bar to another one (since version 6.2.1).
	PATCH: Fixed internal exception when adding resources, entities, or activities with IDs that were removed before under some circumstances (since version 6.2.1).
	PATCH: Fixed not visible links in resources view when using activity links and adding allocation objects again after removing all existing allocation objects before (since version 6.2.1).
	PATCH: Fixed a false triggering of a UI event "contextmenu" when using Firefox additional to triggering the callback onShowContextMenu (since version 6.2.3).
	PATCH: Fixed issue of rows remaining invisible after updating the property ParentID of Activity, Entity, or Resource objects (since version 6.2.1).
	PATCH: Fixed false interpretation of property permissionToPrint in options object of method saveAsPDF.
6.2.4	PATCH: When dragging bars vertically, the drag date lines showed the dates temporarily (since version 6.2.3).
	PATCH: Fixed exception when dragging entity object into the time area at the lower end of the chart.
	PATCH: Fixed an update issue when using the callback visibilityFilter to hide activity bars on collapsed activity rows and showing the bars in ancestor rows.
	PATCH: Fixed a performance issue when using the callback visibilityFilter with a big number of allocations.
	PATCH: The about dialog now shows the open-source library "core-js" as an included component (see also chapter 2.3).
6.2.3	PATCH: The widget now supports showing scroll bars when using Firefox beginning with version 100 on Windows 11.
	PATCH: When using Firefox the inner DIV elements of the widget were focusable by using the Tab key. Now this is prevented.
	PATCH: Fixed an exception when the user clicked into the background of a row within the time area.
	PATCH: The callback onShowTooltip was triggered too often after leaving the curve area of a row.
6.2.2	PATCH: The property cellIndex was not working as expected in callback onShowTooltip. Additionally, it was not documented by accident.

Version	Description of changes
	PATCH: When showing the context menu on a currently selected object all selected objects were deselected (since version 6.2.1).
	PATCH: When using a newer version of D3, the dragged bar was not surrounded by a flashing rectangle anymore.
	PATCH: The shown symbol for the ReleaseDate property of Activity objects was overdrawn by the activity bar when the dates in properties Start and End were in same range.
	PATCH: Interactively resizing bars with options timeStepUnit and timeStepUnitFactor set to coarser values was not showing a smooth phantom.
	Attention: The widget does not show scroll bars when using Firefox beginning with version 100 on Windows 11. This will be fixed in a later patch release. As a workaround you can change the setting "Always show scrollbars" in System Preferences > Accessibility > Visual Effects.
6.2.1	PATCH: Much more performance when changing the property ParentID of Activity, Entity, or Resource objects.
	PATCH: Activity links in resources view under circumstances remained invisible when added in startup phase of widget.
	PATCH: When using activity links in resources view, these were not selectable interactively.
	PATCH: When using multiple bar dragging with one of the options or properties concerning …AllowedBarDragModes set to DragHor+DragVer, horizontal dragging of bars in more than one resource row at the same time did not work as expected.
	PATCH: In some cases, the mouse cursor was not cleared when leaving a bar on screen.
	PATCH: The context menu of the browser is suppressed now on the timescale and in the fixed symbol column of the table when using the secondary mouse button.
	Additionally: The predefined text for topHTML in the PDF options dialog was corrected.
	Additionally: This reference guide now shows small class diagrams for each data object type.
6.2.0 (See also [release notes](#)!)	MINOR: New options topText, bottomText, topHTML, bottomHTML, ownerPassword, userPassword, and permissionTo…, author, title, subject, keywords for method saveAsPDF. (For using topHTML and bottomHTML the additional open-source library "html2canvas" is needed.)
	MINOR: New option pm_dateLineCaptionOptimizedPositioningEnabled.
	MINOR: New property PredefinedGroups for HierachyLevelSupplementary-Definition objects.
	MINOR: Renaming of the following properties in GroupingLevelDefinition objects for alignment with a fallback to the older property names:
	• CodeToTextMap to GroupingCodeToTextMap, • CodeSource to GroupingCodeSource, • TableBackgroundColor to TableColor.

Version	Description of changes
	MINOR: New properties ScaleMinimumValue and ScaleMaximumValue for Curve objects.
	MINOR: New option pm_linesShownInLoadCurvePanes.
	MINOR: New property SuitableResourceIDs for Entity objects and new property SuitableActivityIDs for Allocation and Entity objects.
	MINOR: New callbacks onLogError and onLogWarning.
	PATCH: Fixed positioning when dragging a date line or the vertical splitter and the option visualZoomFactor was set to value unequal to 1.
	PATCH: When saving a PDF document, symbols shown at the top of date lines were not exported.
	PATCH: More performance when updating activity objects.
	Additionally: The Sample App now contains an example for a PDF options dialog.
	Additionally: The Sample App now references current versions of 3rd party libraries.
6.1.11	PATCH: The world view sometimes was scaled to high, so that the view rectangles were not visible fully.
	PATCH: When an active HierarchySupplementaryDefinition object was updated the changes did not get visible in the entities.
	PATCH: The mouse cursor changed too often on allocation bars with allocation entries since version 6.1.10.
6.1.10	PATCH: When dragging allocation bars horizontally erroneously the dragMode property in callback onDrop had flag DragVertically switched on.
	PATCH: A defined symbol on a date line was invisible when either width or height left undefined.
	PATCH: All resources were grayed on dragging multiple allocation bars when the property SuitableResourceIDs was empty on at least one of the Allocation objects. Now an empty value is interpreted consistently on single and multiple bar dragging.
	PATCH: The mouse cursor did not change to "resize" in certain cases on allocation bars.
	PATCH: When dragging an allocation bar vertically the non-working time was not updated, when it should be visible inside the bar.
	PATCH: Setting the option pm_symbolColumnTitleSymbolIDs with the same value as before took too much time.
6.1.9	PATCH: After calling scrollToObject with an allocation the bar representing this allocation was not visible fully when it was out of sight before and is positioned below the first sub row.
	PATCH: When the option loggingEnabled was set to true in the initiation options of the widget, the red recording button was not shown on screen.
	PATCH: When using saveAsPDF without setting zoomFactorInPercent, horPageCountLimit, and verPageCountLimit, the resulting PDF document eventually contained more than page.

Version	Description of changes
6.1.8	PATCH: After calling method selectObjects the time area did not work normally, e.g. rescaling by using the timescale did not update the time area anymore.
	PATCH: When a row object shows a selection frame the sensible area around the collapse/expand button was smaller than without the selection frame.
	PATCH: In a special case the call to fitTimeAreaIntoView started an animation for horizontal scrolling and hindered a following call to scrollToObject.
6.1.7	PATCH: Fixed false scroll position when method scrollToObject was called with option pm_scrollToObjectAnimationEnabled set to true.
	PATCH: Now horizontal scrolling to begin of the time area is avoided when scrollToObject is called for an activity or an allocation object that has no defined start date. In this case now only vertical scrolling is done.
	PATCH: Setting a non-existent ID into the property ParentID of an activity, resource, or entity object using an update method was not working anymore.
	PATCH: Internal exceptions now are visible in browser again.
	PATCH: After calling the method scrollToObject with an entity object, highlighting was not working anymore.
	PATCH: In TooltipTemplate objects used for curve tooltips the reserved words #Load, #Capacity, #Date, #SingleLoads did not show the appropriate values.
6.1.6	PATCH: Fixed missing triggering of callbacks onShowTooltip, onClicked, and onDoubleClicked when visualType is PeriodHighlighter and mouse cursor is on allocation rows.
6.1.5	PATCH: Fixed missing attributes periodHighlighter and entryIndex on callbacks onShowTooltip, onClicked, and onDoubleClicked when visualType is PeriodHighlighter.
6.1.4	PATCH: When using Allocation.SuitableResourceIDs on a bigger data model, it took too long to start dragging on bars.
	PATCH: When adding allocation rows right on expanding the containing row these remained invisible.
6.1.3	PATCH: Fixed exception after updating activity objects while allocation rows are shown in activities view.
6.1.2	PATCH: Fixed hanging mouse cursor when leaving links.
	PATCH: Fixed issue with actual time resolution when options maximumTimeResolutionUnit/-Factor were modified again after widget instantiation.
	PATCH: Fixed exception when dragging an activity bar where the property PM_HasAllocationRows of the activity is set to true without being assigned to existing allocations.
6.1.1	PATCH: The property cellIndex was missing in callback onShowTooltip at least for table rows representing entities.

Version	Description of changes
	PATCH: Fixed missing update of allocation rows in activities view after updating PeriodHighlighter objects.
	PATCH: Time axis could not be shrinked enough anymore when using a big time resolution unit step.
	PATCH: Fixed internal exception handling.
	PATCH: Fixed crash, when start and end were set to null.
6.1.0 (See also release notes!)	MINOR: New property Entries for Activity objects and new object type ActivityEntry.
	MINOR: New options defaultUpdateMode and resetValueForDifferentialUpdate. New value for enumeration UpdateModes.
	MINOR: New option pm_timescaleInteractionMode.
	MINOR: New property DefaultCode for GroupingLevelDefinition objects.
	MINOR: New option pm_resourceHierarchySupplementaryDefinitionIDInLoadsView.
	MINOR: New options maximumTimeResolutionUnit and maximumTimeResolutionUnitFactor. New values for options maximumTimeResolutionUnit, timeStepUnit, and for unit parameter of setTimeResolutionInView method. New enumeration TimeUnit as an alternative for string values.
	MINOR: In TooltipTemplates single curve values are now accessible by using #SingleLoads.*curveID*.
	MINOR: New option onCollapseStateChangedTriggeredByUpdateCalls.
	PATCH: Switching the grouping on or modifying the grouping now is much faster.
	PATCH: Fixed an issue with missing animation on grouping modifications.
	PATCH: Fixed an issue when modifying property TableRowDefinitionID in HierarchyLevelSupplementaryDefinition objects.
	PATCH: Fixed the visibility of allocation object values in tooltips generated by tooltip templates on allocation bars in resources view.
	PATCH: When an allocation referenced a non-existing resource, it could not be filtered in activities view.
	PATCH: Fixed world view issues concerning scaling and scroll bar visibility.
	PATCH: About box now only contains the libraries incorporated inside of VSW library files prefixed with "nwaf-". See revised chapter "System Requirements" for more details.
6.0.5	PATCH: When using the callback visibilityFilter for filtering allocation rows, then the containing row showed a collapse or expand button even when all allocation rows were invisible. Now a special symbol appears instead.
	PATCH: Fixed an issue in PDF export when the time range is huge.
	PATCH: Fixed an issue when modifying the property PM_ViewArea on row objects that have visible allocation rows.
	PATCH: Fixed an issue when the visibilityFilter was modified and allocation rows

Version	Description of changes
	therefore became visible again.
6.0.4	PATCH: When changing the "start" or "end" option, the view start date is now preserved, if possible. PATCH: Fixed exception and malfunction concerning animation when changing objects in fast sequence.
6.0.3	PATCH: Period highlighter grids assigned to resources are now shown also in allocation rows of these resources, in analogy to calendar grids. PATCH: Fixed an issue with PDF export newly appeared with version 6.0.2.
6.0.2	PATCH: Parameters "entry" and "entryIndex" in callbacks onShowTooltip and onShowContextMenu working again. PATCH: The colorization for the grouping and hierarchy levels in the table was not working fully caused by vertical virtualization that is internally used to get more performance. PATCH: Sorting of rows now is triggered correctly when setting the callback option compareObjects using the same Function object. PATCH: Callback onVerticalScrollOffsetChanged now also works correctly when grouping is used.
6.0.1	PATCH: Missing parameter "date" in onClicked callback when time area background was clicked. PATCH: After changing the group criteria in the data of a row object with active grouping, the grouping was not updated. PATCH: When changing the data of an activity object while allocation rows are visible in the activities view, then the allocation rows for this activity object disappeared. PATCH: The new tree view feature did not work correctly when the top view area is visible and the property PM_ViewArea was changed. Additionally: The Sample App now demonstrates the new feature "grouping by criteria" (see also context menu of resource table rows).
6.0.0 (See also [release notes](#)!)	MINOR: New additional grouping by criteria within the current parent-child hierarchy: • New objects HierarchySupplementaryDefinition, HierarchyLevelSupplementary-Definition, GroupingLevelDefinition to specify grouping. • New options pm_activity/resource/entityHierarchySupplementaryDefinitionID to specify the active hierarchy supplementary definition object for additional grouping of activity/resource/entity objects appearing as rows in the appropriate table. • New parameters for callbacks onClicked, onDoubleClicked, onShowContextMenu, onShowTooltip, compareObjects when grouping rows are affected. • New callback determineGroupingCode to specify grouping information. MINOR: New method highlightObjects and therefore renamed options

Version	Description of changes
	pm_scrollToObjectHighlightingColor to pm_objectHighlightingColor and pm_scrollToObjectHighlightFlashingEnabled to pm_objectHighlightFlashingEnabled.
	MINOR: New bar shape named Symbol for allocation and activity bars:
	• New enum value named Symbol in enumerations AllocationBarShape and ActivityBarShape used in property PM_BarShape of allocation and activity objects. • New properties PM_BarShapeSymbolID and PM_BarShapeSymbolWidth for allocation and activity objects.
	MINOR: The bar shape named Diamond now is usable additionally for allocation bars.
	MINOR: To improve a more compact layout optionally there are new options pm_allocation/activityBarTopOffsetAndHeightScaleFactor, pm_entitiesTableCellContentTopOffset, pm_tableCellContentTopOffset, pm_tableTitleAndTimescaleHeight, pm_entitiesTableTitleHeight, pm_progressBarHeight.
	MINOR: New options treeVisualizationMode, pm_treeViewLineColor/DashArray, entitiesTableTreeVisualizationMode, pm_entitiesTableTreeViewLineColor/DashArray and new enumeration TreeVisualizationMode.
	MINOR: New property PM_StrokeDashArray for Curve objects.
	MINOR: New option pm_barSortModeForOptimizedRowDesign, new enumeration BarSortMode.
	MINOR: New option pm_clickCallbackTriggeringOnRowInTimeArea.
	MINOR: New property promise for callbackArgs object in callback onShowTooltip.
	MINOR: New options pm_symbolColumnTitleBackgroundColor and pm_entitiesTableSymbolColumnTitleBackgroundColor.
	MINOR: New property PM_RowSymbolColumnBackgroundColor for Activity, Allocation, Entity, and Resource objects and new property SymbolColumnBackgroundColor for TableRowDefinition objects.
	MINOR: Options pm_top/mainViewAreaVisible for resources view extended to pm_top/mainViewAreaVisibleInActivities/Loads/ResourcesView to cover activities view and loads view additionally.
5.3.7	Internal release.
5.3.6	PATCH: More performance for callback compareObjects by reducing the number of calls to the minimum. Additionally the callback arguments now contains the property viewType.
5.3.5	PATCH: PDF export fixed (issue since 5.3.4). PATCH: The callback compareObjects was not called for allocation rows. PATCH: The callback onClicked was not called on curves anymore. PATCH: In case where allocation rows are visible in resources view, collapsing a resource row did not lead allocation links disappear where needed.

Version	Description of changes
5.3.4	PATCH: Fixed graphical issue when canceling dragging of a date line.
	PATCH: Bars without start and end dates are not shown anymore.
	PATCH: Snapping while dragging a bar now also works for date line grids in mode Automatic.
	PATCH: World view now cannot get higher/wider than widget extent anymore.
	PATCH: Now update calls should be possible when in callback handler function for onSelectionChanged.
	PATCH: Fixed issue when dragging more than one allocation bar and the property SuitableResourceIDs is used at least on some of the allocations.
	PATCH: Setting one of the properties PM_CollapseState and PM_CollapseStateInLoadsView for Resource objects is now also working when resources view or loads view, resp., is not visible.
5.3.3	PATCH: Fixes snapping to start and end dates of other allocations when dragging an allocation.
	PATCH: Fixes an issue for the options pm_ignoreCalendarOnActivity/AllocationBar-Interactions when updating objects while dragging.
	PATCH: Fixes an eventual exception when option pm_activityBaselinesVisible was set.
5.3.2	PATCH: Fixes issue for not showing about dialog anymore when pressing ctrl-alt-shift-F12 since 5.3.1.
5.3.1	PATCH: Property visualSubtype added to the argument of onShowTooltip.
	PATCH: The option pm_activityBaselineBarsVisible now works correctly when it is modified with resources view open and then switched to activities view.
	PATCH: Fixed issue when dragging a release date symbol or due date symbol in Firefox, when the symbol is user-defined.
	PATCH: Fixed issue when using method scrollToObject with an allocation object.
5.3.0 (See also [release notes](#)!)	MINOR: New property PM_CollapseStateInLoadsView for Resource objects.
	MINOR: New options pm_symbolColumnBackgroundColor and pm_entitiesTableSymbolColumnBackgroundColor.
	MINOR: New option pm_ignoreCalendarOnActivityBarInteractions.
	MINOR: New callback onSaveAsPDFProgress and new method cancelSaveAsPDF.
	MINOR: New enumeration PatternType, new properties PM_BarPatternType and PM_BarPatternColor on Activity objects, and new properties PM_PatternType and PM_PatternColor on AllocationEntry objects.
	MINOR: New options pm_dateLineGridColor, pm_dateLineGridDashArray, and pm_dateLineGridWidth.
	MINOR: New option pm_activityBaselineBarsVisible.
	MINOR: New link property PM_TargetMarker and new enumeration LinkMarker.

Version	Description of changes
	PATCH: Fixed exception when modifying the property ParentID of an Activity, Entity, or Resource object with an ID of a non-existing object.
	PATCH: Fixed issues with colored background rectangles in the table when saving a PDF document.
	PATCH: Fixed issue when canceling dragging of a date line by pressing Escape key.
	PATCH: Fixed positioning issues when using bar diamond shapes.
5.2.12	PATCH: Fixed performance issue for removeAll(ObjectType.Resource) again and additionally removeAll(ObjectType.Allocation).
5.2.11	PATCH: Fixed performance issue for removeAll(ObjectType.Resource).
5.2.10	PATCH: Issue fixed for invisible curve pane when resource has set property PM_CurveCollapseState to 0 and was added before its parent resource.
	PATCH: The scrollToDate method was missing the offset parameter.
	PATCH: Fixed issues when using curves of type List.
	PATCH: Fixed issues concerning symbols in table title.
	PATCH: Watermarks were not scaled on screen anymore since version 5.2.9.
	PATCH: When using the secondary mouse button while dragging the action now is canceled.
	PATCH: Fixed issue when sizing a table column interactively that has a background color.
	PATCH: Sometimes the saved PDF file showed collapsed allocation rows and vice versa.
	PATCH: Fixed exception when user clicked into timescale with visible world view.
	PATCH: Fixed issue of resolving object references in an applied tooltip template.
5.2.9	PATCH: Exception fixed concerning adding allocations after first refresh in resources view.
	PATCH: Issue fixed in method saveAsPDF concerning referenced SVG images not visible in PDF.
	PATCH: Infinite loop fixed in method saveAsPDF when many images are not loadable.
5.2.8	PATCH: Performance issue fixed concerning allocation rows in resources view.
5.2.7	PATCH: Issue fixed for callback onShowContextMenu.
	PATCH: Issue fixed for callback visibilityFilter.
5.2.6	PATCH: Issue fixed concerning the callback visibilityFilter used with allocation. Additionally setting the filter did not re-render the widget content.
	PATCH: The callback arguments for the callbacks onClicked, onDoubleClicked, onShowContextMenu did not contain the property cellIndex when called for a table row.

Version	Description of changes
5.2.5	PATCH: Fixed issue of not recognized setting option pm_allocationRowsVisibleInActivities/ResourcesView before first call to render method. Workaround was to change the view type twice. PATCH: Fixed issue with option visibilityFilter not being called immediately when set (issue was existent since 4.0.0).
5.2.4	PATCH: Supplemented missing property tableViewWidth in onTimeAreaViewParametersChanged callback. PATCH: In some situations, the allocation bar was not vanishing on the source row in resources view after dragging vertically to another row. PATCH: Delivered JavaScript files now are prefixed with a UTF8-BOM since in one case concerning Firefox they were misinterpreted as being encoded in ANSI. PATCH: Fixed hanging issue when updating resource objects or period highlighters within drag & drop interaction of allocation bars.
5.2.3	PATCH: Property PM_HasAllocationRows for Resource objects was missing in code and documentation. PATCH: Option pm_defaultResourceAllocationRowsCollapsible was missing in code and documentation. PATCH: Bars within a row object disappeared when the grandparent row object was collapsed, and the bars should remain visible (see PM_CollapsedRowDesign) and the parent row object was not collapsed. PATCH: Property PM_AllocationRowsCollapseState of Resource objects was not working. This document lacked documentation for the properties PM_AllocationRowsCollapsible, PM_AllocationRowsCollapseState for Resource objects introduced with version 5.1.0.
5.2.2	PATCH: Crash fixed when calling saveAsPDF without any links in the chart.
5.2.1	PATCH: New default for property TextSource in TableCellDefinition objects is "", when property SymbolIDSource is set, else the default is "TableText" as before. This is for convenience. PATCH: Animation on expanding/collapsing rows in loads view is enabled again. PATCH: The vertical splitter can now be dragged to the left until the table has a width of 0 even if fixed columns exist. PATCH: When dragging a bar to the border of time than the user cannot drag it out of sight anymore. PATCH: Fixed false property value false of property newRowObjectIsSuitableResource in onDrag callbacks. PATCH: The method selectObjects did not work for allocation bars anymore. PATCH: Symbols and status fields on bars now are stabilized in z-order also concerning the texts in the bars.

Version	Description of changes
	PATCH: Enumeration ObjectType now is correctly documented.
5.2.0 (See also [release notes](#)!)	MINOR: Now additional dates on allocation and activities can be defined either as link source or link target: - New properties LinkSourceDate/LinkTargetDate on activity and allocation objects. - New values for property RelationType on link objects: SourceDateStart, SourceDateEnd, EndTargetDate, StartTargetDate, SourceDateTargetDate. MINOR: Now it is possible to click and double click on symbols in the left fixed symbol column in a table. Therefore a new property symbolIndex was added to the callback arguments of the callbacks onClicked and onDoubleClicked. MINOR: New property PM_TooltipTemplateID on PeriodHighlighterEntry objects. MINOR: New option firstDayOfWeek. MINOR: New options to specify default tooltip templates: pm_defaultActivityBar/RowTooltipTemplateID, pm_defaultAllocationBar/RowTooltipTemplateID, pm_defaultEntityRowTooltipTemplateID, pm_defaultLinkTooltipTemplateID, pm_defaultPeriodHighlighterEntryTooltipTemplateID, pm_defaultResourceRow/CurveTooltipTemplateID. MINOR: New enumeration RelationType for links. MINOR (is a MAJOR change when updating from 5.1.0): New properties start/endPropertyName in callbackArgs of callbacks canDrag, onDragStart, onDrag, onDragEnd, onDrop. PATCH: Fixed performance issue that was existent since 5.1.0 because of implementation of allocation rows when using links. PATCH: Fixed issues with tooltip template markup using keywords beginning with #, and concerning date formatting. PATCH: Fixed issue concerning eventually false week numbering in timescale. PATCH: Fixed issue in world view that occurred under certain circumstances when changing time resolution in main view. PATCH: Fixed issues concerning graphical links and missing animations after updating the data model.
5.1.0 **(WAS NOT OFFICIALLY PUBLISHED)**	MINOR: Now it is possible to show allocations in own rows also in the resources view using the new option pm_allocationRowsVisibleInResourcesView. MINOR: Symbols shown for properties ReleaseDate and DueDate on Activity objects are now specifiable and draggable: - New properties PM_ReleaseDateAllowedDragModes, PM_ReleaseDateSymbolHeight, PM_ReleaseDateSymbolID, and PM_ReleaseDateSymbolWidth for the release date. - New properties PM_DueDateAllowedDragModes, PM_DueDateSymbolHeight, PM_DueDateSymbolID, and PM_DueDateSymbolWidth for the due date. - New property propNames in callbackArgs of callback onDrop. See MAJOR change in 5.2.0 to properties startPropertyName and endPropertyName.

Version	Description of changes
	MINOR: New properties Background/TextColor on TableRowDefinition objects.
	MINOR: New argument cellIndex at callbacks onClicked and onDoubleClicked.
	MINOR: New methods scrollViewAreaHorizontally and scrollViewAreaVertically.
	MINOR: New locales added for Japanese, Russian, Thai, and Chinese.
	MINOR: Clarification of options and properties concerning title, a.o. renaming Header to Title (compatibility is given):
	• New option names are pm_tableTitleBackgroundColor, pm_tableTitleTextColor, pm_tableTitleColumnSeparatorColor, pm_tableTitleHighlightingColor, pm_entitiesTableTitleBackgroundColor, pm_entitiesTableTitleTextColor, pm_entitiesTableTitleColumnSeparatorColor, pm_entitiesTableTitleHighlightingColor. Old option names remain in the interface, but are marked as deprecated.
	MINOR: Property Title of TableCellDefinition objects renamed to TitleText (compatibility is given).
	PATCH: Method selectObjects did not work for allocation objects in activities view.
5.0.2	PATCH: Wrong coloring of allocation bars.
	PATCH: After updating links, they have not been rendered correctly.
5.0.1	PATCH: Fix for issue when dragging an entity without a non-zero duration.
5.0.0 (see also [release notes](#)!)	MINOR: New method saveAsPDF.
	MINOR: Now allocation rows can be made visible in activities view by using the new option pm_allocationRowsVisibleInActivitiesView. Additionally other additions were made in this environment:
	• New options pm_defaultAllocationTableRowDefinitionID, pm_defaultAllocationMinimumRowHeight, pm_defaultAllocationRowSelectable, pm_defaultActivityAllocationRowsCollapsible, pm_defaultAllocationRowDesign, pm_defaultAllocationAllowedBarDragModesInActivitiesView, pm_forcedAllocationAllowedBarDragModesInActivitiesView. • New properties PM_AllocationRowsCollapsible, PM_AllocationRowsCollapseState, PM_HasAllocationRows on Activity objects. • New properties PM_MinimumRowHeight, PM_RowSelectable, PM_RowDesign, PM_AllowedBarDragModesInActivitiesView, TableText on Allocation objects. • New callbackArgs property isForAllocationRows in callback onCollapseStateChanged.
	MINOR: Now links can be defined between allocations:
	• New properties Source/TargetAllocationID for links. • New options pm_definedAllocationLinksVisibleInActivitiesView/ResourcesView.

Version	Description of changes
	MINOR: New optional parameter for method removeAll.
	MINOR: New options pm_watermarkSymbolID and pm_watermarkOpacity.
	MINOR: New properties PM_StatusFrameColor/Visible for activities and allocations. New options pm_defaultActivity/AllocationStatusFrameColor.
	MINOR: New property PM_BarOpacity for Activity and Allocation objects.
	MINOR: New properties Background/TextColor, Background/TextColorSource on TableCellDefinition objects.
	MINOR: New options tableViewWidthInActivitiesView/ResourcesView/LoadsView and tableViewWidthsSynchronized.
	MINOR: New options pm_symbolColumnTitleVisible/SymbolIDs as well as pm_entitiesTableSymbolColumnTitleVisible/SymbolIDs.
	MINOR: New option pm_scrollOffsetsChangedCallbackTimeDelay.
	MINOR: Additional values for callback onVerticalScrollOffsetChanged.
	MINOR: Constraint dates are now also considered in the summary and in the diamond bar shape of allocations and activities. For the diamond shape, the PredictedEnd property is also taken into account.
	MINOR: Options pm_activity/resourceTableRowDefinitionIDForTitle renamed to pm_tableRowDefinitionIDForTitleInActivitiesView/ResourcesView. Option pm_entityTableRowDefinitionIDForTitle renamed to pm_tableRowDefinitionIDForTitleInEntitiesTable. Old option names are deprecated but accepted for compatibility reasons.
	MINOR: New option pm_tableRowDefinitionIDForTitleInLoadsView.
	MINOR: New option tooltipDelay.
	MAJOR: After dragging and dropping a draggable date line, the application now needs to update the date line object within the onDrop callback handler to apply the changed date. In older versions the date line incorrectly remained on the new date.
	MAJOR: Activity rows and resource rows do not show the calendar of an ancestor anymore when the represented objects do not have an own calendar.
	PATCH: Fix for issue with hidden rows when using method scrollToObject.
	PATCH: Several fixes concerning tooltips and captions of PeriodHighlighterEntry objects.
4.0.5	PATCH: Fixed and improved appearance of bars while dragging and of dragged entities in time area also especially when using the options timeStepUnit and timeStepUnitFactor for a more intuitive user experience.
	PATCH: Fixed cursor issue on entities table, not being updated correctly when moving the mouse.
	PATCH: Fixed issue concerning snapping when dragging bars in collapsed rows and bars of collapsed sub rows are shown.

Version	Description of changes
	PATCH: Fixed issue concerning snapping when dragging bars and start dates or end dates of other bars contain millisecond values. Now these start dates and end dates are rounded down or up, resp., to full seconds.
	PATCH: Property date of callbackArgs in callback onShowTooltip was not existent since 4.0.3.
4.0.4	PATCH: Fixed crash when using world view together with bar symbols.
	PATCH: Canceling of deselection of all objects in a callback handler for callback onSelectionChanged now possible. Additionally, new attributes for callbackArgs named reason, reasonObject, reasonObjectType.
	PATCH: Zooming out of the currently visible time range by using the "up" button in the timescale resulted in an incorrect horizontal scroll offset.
	PATCH: Fixed issue when using the method updateDateLines (not all properties have been updated).
	PATCH: Fixed issue where the curves were not shown when adding a resource with property PM_CurveCollapseState set to 0.
	PATCH: Fixed issue with setting options timeStepUnit and timeStepUnitFactor.
4.0.3	PATCH: Fixed naming issues for external dependencies "hammerjs" and "tinycolor2".
4.0.2	PATCH: Improved behavior for bar dragging.
	PATCH: Fixed issue concerning AMD for external dependency "jquery-ui/ui/widget" and "jquery".
4.0.1	PATCH: Fixed a crash that occurred when the pm_defaultActivityBarHeight option was set when initializing the widget.
	PATCH: Default for option viewType now is ViewType.Activities again (since 4.0.0 it was set to ViewType.Resources).
4.0.0 (See also [release notes](#)!)	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: When dragging a bar vertically the visible time span of it now is adapted according to the calendar of the current target row.
	MINOR: World view implemented. See options worldViewVisible, worldViewPosition, worldViewExtent.
	MINOR: Improved loading performance.
	MINOR: New options loggingEnabled and interactiveActivationOfLoggingEnabled.

Version	Description of changes
	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: New callback "visibilityFilter" triggered for filtering 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 onClicked again.
3.2.0 (See also release notes!)	MINOR: New options cursorDateLineVisible, pm_timeAreaPanningMode, pm_timescaleInteractionsEnabled, and pm_curvePanesCollapsibleInResourcesView.
	MINOR: New options currentDate, pm_pastBackgroundFillColor/LineColor/LineWidth/LineDashArray.
	MINOR: New option timeZone.
	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.

Version	Description of changes
	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.
3.1.0 (See also [release notes](#)!)	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
3.0.0	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. 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.
2.1.0	MINOR: New method about.

Version	Description of changes
	MINOR: New message boxes for invalid, expiring, expired, not existing license.
2.0.0	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.
1.0.0	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 at delivery date of library)

2.2   Needed 3rd Party Libraries and Versions

The libraries with the prefix nwaf contain the VSW SE and are copyrighted by NETRONIC. They depend on the following 3rd party, open-source libraries.

In your application you can include these libraries directly or you can download these from a Content Delivery Network (CDN). For an example see the VSW SE Sample App.

Your application must obey the license conditions of each 3rd party library!

Library Name NPM Module Name	Supported Versions	Comment
jQuery **"jquery"**	2.x.x/3.x.x (last one tested: 3.6.0)	Required. Needed for HTML handling. Versions 2.x.x support older Internet Explorer versions (but these are not supported by VSW SE!). License: MIT. URL: https://jquery.com/
jQuery UI **"jquery-ui/ui/…"**	1.11.x/1.12.x/1.13.x (last one tested 1.13.1)	Required. Needed parts are the Widget Factory (widget) and Dialogs (widgets/dialog). License: MIT. URL: https://jqueryui.com/

D3.js "d3"	4.0.0-7.x.x (last one tested: 7.4.4)	Required. Needed for SVG handling. Beginning with version 5.0.0 Internet Explorer is not supported officially anymore. License: ISC. URL: https://d3js.org/ Used modules: d3-axis, d3-ease, d3-format, d3-scale, d3-selection, d3-shape, d3-time-format, d3-timer, d3-transition.
Hammer.js "hammerjs"	2.0.8	Required. Needed for touch and mouse gesture handling. License: MIT. URL: https://hammerjs.github.io/
TinyColor "tinycolor2"	1.4.1	Required. Needed for calculating derived colors e.g. for coloring non-working times. License: MIT. URL: https://bgrins.github.io/TinyColor/
Moment.js "moment"	2.x.x (last one tested: 2.29.1)	Optional. Needed only, when using option "timeZone". The developer can decide, which data to serve with Moment Timezone. License: MIT. URL: https://momentjs.com/
Moment.Timezone "moment-timezone"	0.5.x (last one tested: 0.5.34)	Optional. Needed only, when using option "timeZone". License: MIT. URL: https://momentjs.com/
SVG-to-PDFKit "svg-to-pdfkit"	0.1.8 *	Optional. Needed only, when using method saveAsPDF. License: MIT. URL: https://github.com/alafr/SVG-to-PDFKit
PDFKit "pdfkit"	0.11.0-0.13.0	Optional. Needed only, when using method saveAsPDF (needed by SVG-to-PDFKit). Provided standalone version contains libraries "stream" and "emitter-component". Licenses: MIT. URL: https://pdfkit.org/
blob-stream "blob-stream"	0.1.3	Optional. Needed only, when using method saveAsPDF (needed for receiving the PDF content generated by PDFKit). Provided standalone version contains libraries "stream", "emitter-component", and "blob". Licenses: MIT. URL: https://github.com/devongovett/blob-stream

html2canvas "html2canvas"	1.4.x (last one tested: 1.4.1)	Optional. Needed only, when using method saveAsPDF together with the options topHTML and/or bottomHTML. License: MIT. URL: https://html2canvas.hertzen.com/

Note: The jQuery plugin jquery.mousewheel, which was required up to VSW SE 3.1, is no longer required as of VSW SE 3.2.

* There is a bug in all versions of SVG-to-PDFKit that lowers the performance from about 1 second per exported page to about 1 minute per page. We were able to find and fix this bug (see pull request here: https://github.com/alafr/SVG-to-PDFKit/pull/143). The patch is included in master version of the source code on GitHub, but there is no release yet. Therefore, we deliver the downloaded master version for your convenience.

2.3   Imported 3ʳᵈ Party Libraries and Versions

These libraries are imported into our library file nwaf-apptools.min.js.

Library Name NPM Module Name	Version	Comment
CSS Element Queries css-element-queries"	1.2.3	The included resize sensor triggers a callback when the size of a DIV element changes in the DOM. License: MIT. URL: https://github.com/marcj/css-element-queries
FileSaver "file-saver"	2.0.5	Imported to support logging (see option loggingEnabled) and PDF export (see method saveAsPDF). License: MIT. URL: https://github.com/eligrey/FileSaver.js
core-js "core-js"	3.23.1	Parts of this polyfill library are compiled into NWAF libraries by Babel.js to allow developing with current JavaScript standards while remaining compatible to the supported browsers and versions. License: MIT. URL: https://github.com/zloirock/core-js
Polyfill-library "polyfill-library"	4.5.0	Parts of this polyfill library are used to allow using web standard methods while remaining compatible to the supported browsers and versions. License: MIT. URL: https://github.com/Financial-Times/polyfill-library

3   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.

In this document you will find some UML diagrams that illustrate at a glance the relationships between the object type currently under consideration and the associated ones**. Only those object properties are listed that are essential for understanding the concept of this data model**. The **dark blue shaped types** in these diagrams are the ones that can be processed with the methods add..., update... and remove... of the widget. To do this, they explicitly provide identifiers in the form of the ID property. In contrast, the **light blue types** include dependent objects without their own identifiers.

A compact description of the model can be found in the document "A Model for Resource Planning HTML5 Gantt Charts" delivered with this product (please see the file ResourcePlanningModel.pdf).

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

3.1   Activities
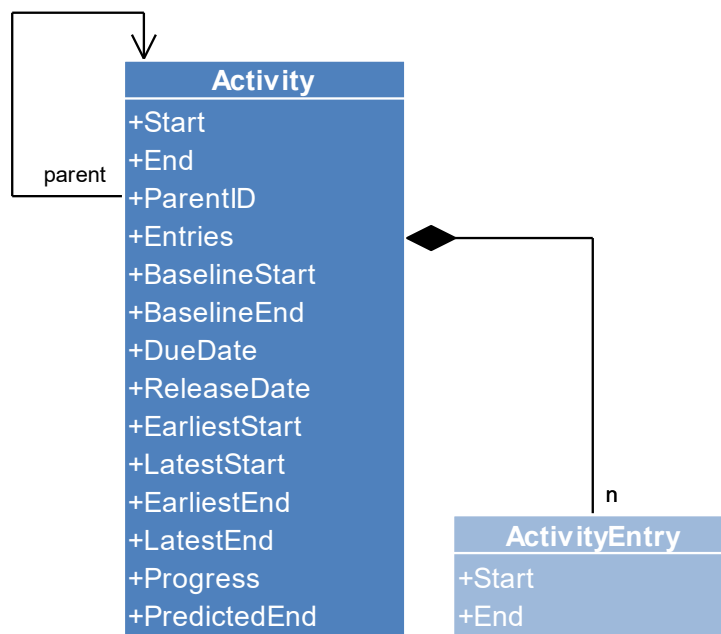
3.1.1   Activity
An Activity object defines the properties of a single activity.

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. See also setting PM_ActivityBaselineBarsVisible.
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. See also setting PM_ActivityBaselineBarsVisible.
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_releaseDueDateConnections Visible, 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.
(Deprecated! Use PM_AllowedBarDragModes instead.) **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
Entries	ActivityEntry[]	**Optional, default: undefined** – array of activity entries. If used, then the entries will be shown as colored rectangles within the bar representation of the activity. Additionally the property PM_BarShape must be set to Regular or Rectangle.
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.
LinkSourceDate	Date|string	**Optional, default: undefined** – Additional date serving as an additional "start point" to connect a link. See also property "RelationType" of link object.
LinkTargetDate	Date|string	**Optional, default: undefined** – Additional date serving as an additional "end point" to connect a link. See also property "RelationType" of link object.
MustEndOn	Date|string	**Optional, default: undefined** – If defined, an additional symbol will be displayed to indicate this date.

Activity Property Name	Type	Description
		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.
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. We recommend using only a low number of hierarchy levels and we do not guarantee correct function beyond approx. 100 levels including hierarchy levels created by using Hierarchy-SupplementaryDefinitions.
PM_AllocationRowsCollapseState	number (see enum CollapseState)	**Optional, default: -1** – Specifies whether the allocation rows of the activity should be expanded or collapsed when displayed. See also callback onCollapseStateChanged and options pm_allocationRowsVisibleInActivitiesView, pm_allocationRowsVisibleInResourcesView.

Activity Property Name	Type	Description
		-1: no change 0: display activity row in an expanded way for allocation rows. 1: display activity row in a collapsed way for allocation rows.
PM_AllocationRowsCollapsible	boolean	**Optional, default: true** – If set to true, then the row representing this activity row will be interactively collapsible when allocation rows exist.
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_AllowedRowDragModes	number (see enum RowDragModes)	**Optional, default: value of option pm_defaultActivityAllowedRowDragModes** – This option determines the allowed row drag modes for this activity (these can be overwritten using the callback canDrag).
PM_BarHeight	number ($\geq 0, \leq 1000$)	**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_BarOpacity	number ($\geq 0.0, \leq 1.0$)	**Optional, default: undefined** – Specifies the opacity of the entire activity bar (including the visualization of the progress bar, symbols, constraint dates, and baseline bar).
PM_BarPatternColor	string	**Optional, default: "white"** – Color for the pattern when this is visible by using property PM_BarPatternColor.

Activity Property Name	Type	Description
PM_BarPatternType	number (see enum PatternType)	**Optional, default: undefined** – If set, then a pattern is shown on top of the fill color and behind the text. ATTENTION: Value ignored when using IE11!
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_BarShapeSymbolID	string	**Optional, default: undefined** – If the bar shape named Symbol is used (see property PM_BarShape), then the symbol defined here will be shown. The symbol will be resized to the height defined in property PM_BarHeight and to the width defined in property PM_BarShapeSymbolWidth.
PM_BarShapeSymbolWidth	number > 0	**Optional, default: value of property PM_BarHeight** – Width of the symbol defined in property PM_BarShapeSymbolID when the bar shape named Symbol is used.
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 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%.

Activity Property Name	Type	Description
PM_BarTextWrapMode	number (see enum <u>TextWrapMode</u>)	**Optional, default: TextWrapMode.None** – Specifies whether the text inside the bar is wrapped.
PM_BarTooltipTemplateID	string	**Optional, default: value of option pm_defaultActivityBarTooltip-TemplateID** – 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_BaselineNonworkingTimeColor 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 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.

Activity Property Name	Type	Description
PM_BorderDashArray	string	**Optional, default: ""** – Dash array of the border of the bar.
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 explanation for possible values in enumeration chapter.
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: "#646464"** – 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_DueDateAllowedDragModes	number (see enum [ActivityBarDragModes](#))	**Optional, default: None** – This option determines the allowed drag mode for the due date of this activity in the activities view (these can be overwritten using the callback canDrag). In this context only None and DragHorizontally are used.

Activity Property Name	Type	Description
PM_DueDateColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: internal diamond symbol** – Color for the due date symbol.
PM_DueDateSymbolHeight	number	**Optional, default: 12** – Height of the due date symbol (see property PM_DueDateSymbolID) in pixels at a zoom factor of 100%. The default symbol is not sizable.
PM_DueDateSymbolID	string	**Optional, default: internal diamond symbol** – Identifier of the symbol to be shown at the due date of the activity. See also PM_DueDateSymbol-Height, and PM_DueDate-SymbolWidth.
PM_DueDateSymbolWidth	number	**Optional, default: 12** – Width of the due date symbol (see property PM_DueDateSymbolID) in pixels at a zoom factor of 100%. The default symbol is not sizable.
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.
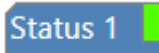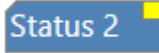
Activity Property Name	Type	Description
		See explanation for possible values in enumerations chapter.
PM_HasAllocationRows	boolean	**Optional, default: false** – If set to true, then the row representing this activity will be collapsible/expandable for allocation rows even when no allocations exist referencing this activity. This serves for lazy loading.
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.
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
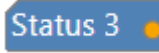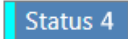
Activity Property Name	Type	Description
		PM_LeftBarSymbolID) in pixels at a zoom factor of 100%.
PM_MinimumRowHeight	number	**Optional, default: value in option pm_defaultMinimumActivityRowHeight** – Minimum height of the activity row in pixels. This attribute is useful, when more than one line of text 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.
PM_NonworkingTimeColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red" or "calculated")	**Optional, default: value of property PM_Color** – Color for the nonworking time periods of the bar. 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.

Activity Property Name	Type	Description
PM_PredictedEndColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: "#646464"** – 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: "#646464"** – 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: value of property PM_ProgressColor** – Color for the nonworking time periods of the progress bar. If set to "calculated", a color will be calculated using the color defined by the PM_ProgressColor property.
PM_ReleaseDateAllowedDragModes	number (see enum ActivityBarDragModes)	**Optional, default: None** – This option determines the allowed drag mode for the release date of this activity in the activities view (these can be overwritten using the callback canDrag). In this context only None and DragHorizontally are usable.
PM_ReleaseDateColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: undefined** Color for the release date symbol. If undefined, a default value of the widget will be used.
PM_ReleaseDateSymbolHeight	number	**Optional, default: 12** – Height of the release date symbol (see property PM_ReleaseDateSymbolID) in pixels at a zoom factor of 100%. The default symbol is not sizable.
PM_ReleaseDateSymbolID	string	**Optional, default: internal diamond symbol** – Identifier of

Activity Property Name	Type	Description
		the symbol to be shown at the due date of the activity. See also PM_ReleaseDateSymbolHeight and PM_ReleaseDateSymbolWidth.
PM_ReleaseDateSymbolWidth	number	**Optional, default: 12** – Width of the release date symbol (see property PM_ReleaseDateSymbolID) in pixels at a zoom factor of 100%. The default symbol is not sizable.
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_RowSymbolColumn-BackgroundColor	string (CSS color value)	**Optional, default: value of property SymbolColumnBackgroundColor of assigned table row definition or option pm_symbolColumnBackgroundColor** – Determines the color of

Activity Property Name	Type	Description
		the symbol column within this table row.
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: value of option pm_defaultActivityRow-TooltipTemplateID** – 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

Activity Property Name	Type	Description
		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_SortCode	number|string|Date	**Optional, default: undefined** – If set, then the value will be used when sorting activity rows. The value type can be anyone that can be compared using JavaScript. See also options activityRowSortMode and activityRowSortCodePropertyName.
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 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",	**Optional, default: undefined** – Color for the status symbol to the right of the bar. If undefined, no

Activity Property Name	Type	Description
	"rgb(255, 0, 0)", or "red")	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_StatusFrameColor	string	**Optional, default: value of option pm_defaultActivityStatusFrameColor** – Color for the status frame that will be shown when property PM_StatusFrameVisible is set.
PM_StatusFrameVisible	boolean	**Optional, default: false** – If set to true, then a frame is shown around the bar. See also property PM_StatusFrameColor.
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_defaultActivityTableRowDefinitionID** – Identifier of a TableRowDefinition object, that defines the composition of the table row.

Activity Property Name	Type	Description
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%.
PM_ViewArea	number (see enum [ViewArea](#))	**Optional, default: Main** – 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 also options pm_mainViewAreaVisibleInActivitiesView and pm_topViewAreaVisibleInActivitiesView.
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 formatted date string in a

Activity Property Name	Type	Description
		standardized way, one has to be careful about it.
Progress	number (floating point; in percent; $\geq 0, \leq 100$)	**Optional, default: 0.0** – Used to display a completion layer.
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_releaseDueDateConnections Visible, 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).

3.1.2 **ActivityEntry**

Objects of this type are only used within the array of the Entries property of Activity objects.

ActivityEntry Property Name	Type	Description
End	Date|string	**Optional, default: undefined** – End date of the activity entry. This date itself is not(!) part of the interval described by this entry.

ActivityEntry Property Name	Type	Description
		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 activity, if available, will be used. The nonworking time periods of the bar will be colored with the same color as long as the property PM_NonworkingTimeColor of the appropriate activity is undefined or set to "calculated".
PM_Height	number (≥ 0, ≤ 1000)	**Optional, default: value in option pm_defaultActivityBarHeight** – 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 activity, if available, will be used. If that one is also undefined, then the nonworking time periods of the bar will be colored with the same color as the working times (see PM_Color property). If set to "calculated", a color will be calculated using the color defined by the PM_Color property.
PM_PatternColor	string	**Optional, default: "white"** – Color for the pattern when this is visible by using property PM_PatternColor.
PM_PatternType	number (see enum PatternType)	**Optional, default: undefined** – If set, then a pattern is shown on top of the fill color and behind the text. ATTENTION: Value ignored when using IE11!
PM_RelativeTopOffset	number	**Optional, default: 0** – Offset of the entry in pixels relative to the upper side of the corresponding activity bar. 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 activity entry.

ActivityEntry Property Name	Type	Description
		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).

3.2 Allocations
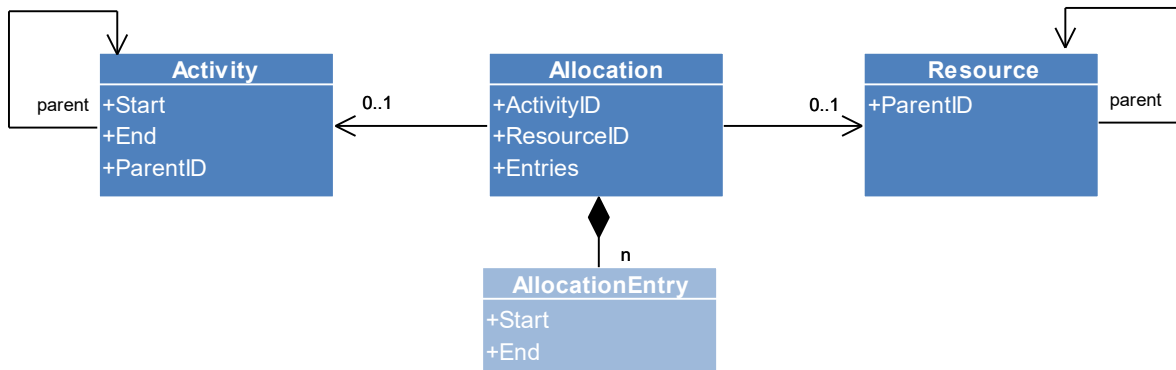
3.2.1 Allocation

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

Allocation Property Name	Type	Description
		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.
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.
LinkSourceDate	Date|string	**Optional, default: undefined** – Additional date serving as an additional "start point" to connect a link. See also property "RelationType" of link object.
LinkTargetDate	Date|string	**Optional, default: undefined** – Additional date serving as an additional "end point" to connect a link. See also property "RelationType" of link object.
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.

Allocation Property Name	Type	Description
		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 AllocationBarDrag-Modes)	**Optional, default: value of option pm_defaultAllocation-AllowedBarDragModes** – This option determines the allowed bar drag modes for this allocation in the resources view (these can be overwritten using the callback canDrag).
PM_AllowedBarDragModes-InActivitiesView	number (see enum AllocationBarDrag-Modes)	**Optional, default: value of option pm_defaultAllocation-AllowedBarDragModesInActivitiesView** – This option determines the allowed bar drag modes for this allocation in the activities view (these can be overwritten using the callback canDrag).
PM_BarHeight	number ($\geq 0, \leq 1000$)	**Optional, default: value in option pm_defaultAllocationBarHeight** – 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_BarOpacity	number ($\geq 0.0, \leq 1.0$)	**Optional, default: undefined** – Specifies the opacity of the entire allocation bar (including the visualization of the progress bar, symbols, and constraint dates).
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_BarShapeSymbolID	string	**Optional, default: undefined** – If the bar shape named Symbol is used (see property PM_BarShape), then the symbol defined here will be shown. The symbol will be resized to the height defined in property PM_BarHeight and to the width defined in property PM_BarShapeSymbolWidth.
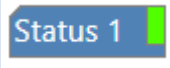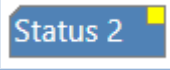
Allocation Property Name	Type	Description
PM_BarShapeSymbolWidth	number > 0	**Optional, default: value of property PM_BarHeight** – Width of the symbol defined in property PM_BarShapeSymbolID when the bar shape named Symbol is used.
PM_BarTextPrefixSymbolHeight	number	**Optional, default: 12** – Height 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: value of option pm_defaultAllocationBarTooltip-TemplateID** – 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 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_BorderDashArray	string	**Optional, default: ""** – Dash array of the border of the bar.
PM_Color	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red" or "calculated")	**Optional, default: "#646464" or color of the referenced activity** –Color for the working time periods of the bar. If undefined, either the value of the PM_Color property of the referenced activity - if such a reference exists or else the value "#646464" will be used.

Allocation Property Name	Type	Description
		See also property PM_NonworkingTimeColor for the coloring of the nonworking times.
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 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.

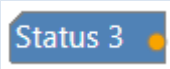| Allocation Property Name	Type	Description
		See also PM_RightBarSymbolID, PM_LeftBarSymbolHeight, and PM_LeftBarSymbolWidth.
PM_LeftBarSymbolWidth	number	**Optional, default: 12** – Width of the left bar symbol (see property PM_LeftBarSymbolID) in pixels at a zoom factor of 100%.
PM_MinimumRowHeight	number	**Optional, default: value in option pm_defaultAllocationMinimumRowHeight** – Minimum height of the allocation 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. 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_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: "#646464" or color of the referenced activity** –Color for the nonworking time periods of the bar. If set to "calculated", a color will be calculated using the color defined by the PM_Color property. If undefined, either the nonworking time color of the referenced activity - if such a reference exists - or else the value "#646464" will be used. See also property PM_Color for the coloring of the working times.
PM_PredictedEndColor	string (CSS color value, e.g. "#ff0000",	**Optional, default: undefined** – Color for the predicted end bar.

Allocation Property Name	Type	Description
	"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: value of property PM_ProgressColor of activity or if undefined, then "#646464"** – 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.
PM_ProgressNonworking-TimeColor	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 property PM_RightBarSymbolID) in pixels at a zoom factor of 100%.
PM_RowDesign	number (see enum RowDesigns)	**Optional, default: value in option pm_defaultAllocationRowDesign** – Specifies how the time area is filled when the row is visible. See explanation for possible values in enumeration chapter.
PM_RowSelectable	boolean	**Optional, default: value of option pm_defaultAllocationRowSelectable** – If set

Allocation Property Name	Type	Description
		to true, then the row representing this allocation will be selectable.
PM_RowSymbolColumn-BackgroundColor	string (CSS color value)	**Optional, default: value of property SymbolColumnBackgroundColor of assigned table row definition or option pm_symbolColumnBackgroundColor** – Determines the color of the symbol column within this table row.
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: value of option pm_defaultAllocationRowTooltip-TemplateID** – ID of a tooltip template. The template is used for tooltips that appear on the entity table rows.
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

Allocation Property Name	Type	Description
		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_SortCode	number|string|Date	**Optional, default: undefined** – If set, then the value will be used when sorting allocation rows. The value type can be anyone that can be compared using JavaScript. See also options allocationRowSortMode and allocationRowSortCodePropertyName.
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.
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.

Allocation Property Name	Type	Description
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_StatusFrameColor	string	**Optional, default: value of option pm_defaultAllocationStatusFrameColor** – Color for the status frame that will be shown when property PM_StatusFrameVisible is set.
PM_StatusFrameVisible	boolean	**Optional, default: false** – If set to true, then a frame is shown around the bar. See also property PM_StatusFrameColor.
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

Allocation Property Name	Type	Description
		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, $\leq$ 100)	**Optional, default: 0.0** – Used to display a completion layer.
ResourceID	string	**Optional, default: undefined** – Identifier of a [Resource](#).
SuitableActivityIDs	string[]	**Optional, default: undefined** – An array of IDs of those activities to which the allocation could be assigned. If the array is defined, then all rows of activities that are not listed in that array will be covered by a half-transparent curtain. If the array is empty, all activity 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_suitableActivityOverlayColor and pm_unsuitableActivityOverlayColor.
SuitableResourceIDs	string[]	**Optional, default: undefined** – An array of IDs of those resources to which the allocation could be assigned. 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.

Allocation Property Name	Type	Description
TableText	string	**Optional, default: undefined** – Text to display in the table row (see also property PM_TableRowDefinitionID).

3.2.2   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: default: value of property PM_Color of allocation** – 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: value of property PM_NonworkingTimeColor of allocation or if undefined then value of property PM_Color** – Color for the nonworking time periods of the bar. If set to "calculated", a color will be calculated using the color defined by the PM_Color property.
PM_PatternColor	string	**Optional, default: "white"** – Color for the pattern when this is visible by using property PM_PatternColor.
PM_PatternType	number (see enum [PatternType](#))	**Optional, default: undefined** – If set, then a pattern is shown on top of the fill color and behind the text. ATTENTION: Value ignored when using IE11!
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

AllocationEntry Property Name	Type	Description
		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).

3.3   Calendars
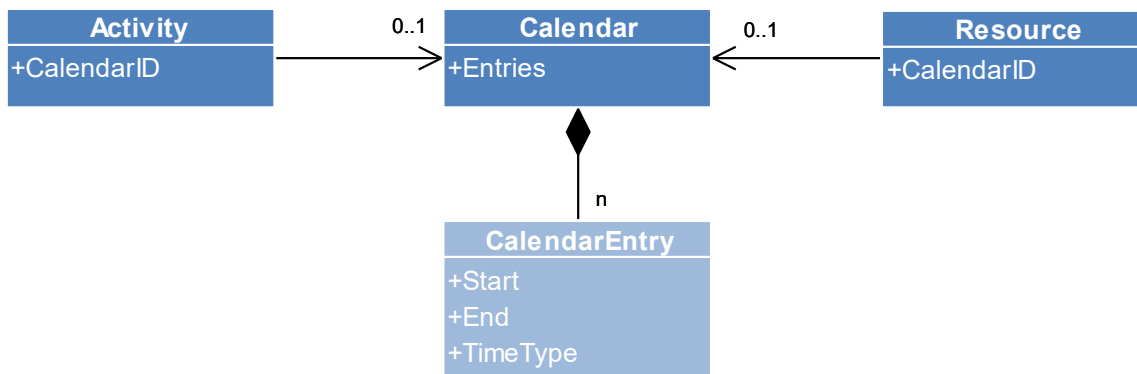
3.3.1   Calendar

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

3.3.2   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 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

3.4 Curves
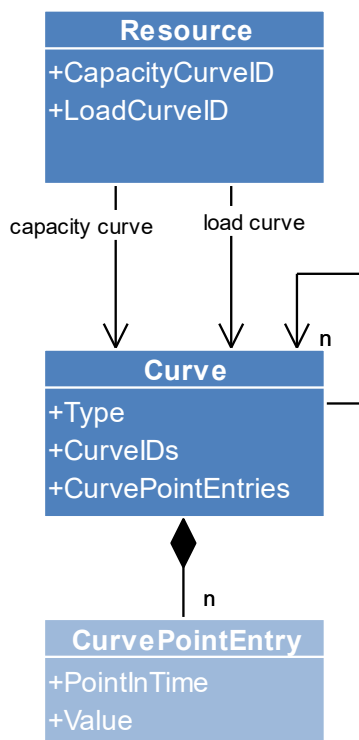
Curve objects serve to define values over time that can be shown as capacity or load inside resource and activity rows (see properties LoadCurveID and CapacityCurveID in Resource object). Additionally,

it is possible to stack curves when using curve object of stack type. Currently, there are no curve types that calculate their values automatically.

Curves are displayed in curves panes. Each pane can hold several curves (e.g. a capacity and a load curve). On the right side of a pane a numerical scale is displayed. By default, the minimum and maximum values for the tick markers are calculated automatically. However, this can be affected by setting the ScaleMinimumValue and the ScaleMaximumValue properties of the curves.

3.4.1   Curve

Curve Property Name	Type	Description
CurveIDs	string[]	**Optional, default: undefined** – Array of curve IDs (in case of CurveStack and CurveList only).
CurvePointEntries	CurvePointEntry[]	**Optional, default: undefined** – Array of point entries (in case of PointCurve only).
ID	string	**Required** – Identifier of the curve.
PM_FillColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: "transparent"** – 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: StepAfter** – 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", "rgb(255, 0, 0)", or "red")	**Optional, default: "#E01818"** – Used, when the curve is used as the load curve that referenced directly by the property LoadCurveID at the object. Then the area 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: "transparent"** – Color of the curve line itself.
PM_StrokeDashArray	string (SVG stroke dash array value)	**Optional, default: ""** – Dash array definition for the curve line.
ScaleMaximumValue	number	**Optional, default: Number.NEGATIVE_INFINITY** – If this value here is greater than all values of this curve, then it defines the explicit maximum. Otherwise, the maximum of all curve values defines the overall maximum of this curve.

Curve Property Name	Type	Description
		Hereby together with the property ScaleMinimumValue you can define the shown value range for the curve pane.
ScaleMinimumValue	number	**Optional, default: Number.POSITIVE_INFINITY** – If this value here is lower than all values of this curve, then it defines the explicit minimum. Otherwise, the minimum of all curve values defines the overall minimum of this curve. Hereby together with the property ScaleMaximumValue you can define the shown value range for the curve pane.
Type	number (see enum CurveType)	**Optional, default: 0** – Type of the curve. A point curve contains a number of entries in the property CurvePointEntries. A curve stack and a curve list contains a number of IDs of other curves in the property CurveIDs. A curve stacks stacks the contained curves in the order inside the array optically. A curve list shows the contained curves one by one at the same space, so it is recommended to use translucent colors for filling the curves. Currently it is recommended not to put lists or stacks into other lists/stacks!

3.4.2 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.

3.5   DateLine

DateLine
+PointInTime
+Caption
+SymbolID

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. **Note:** If the caption position is TopCenter, TopLeft or TopRight (see the CaptionPosition property) and if the orientation is vertical, the caption orientation will still be positioned at the date line and not within the timescale.
CaptionPosition	number (see enum DateLineCaptionPosition)	**Optional, default: 1** – Specifies where the caption should be positioned relative to the date line. See also property CaptionOrientation and option pm_dateLineCaptionOptimizedPositioningEnabled.
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.

DateLine Property Name	Type	Description
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.
SymbolID	string	**Optional, default: undefined** – When set, then the referenced symbol will be visible at the top of the date line. See also properties SymbolHeight/Width.
SymbolHeight	number > 0	**Optional, default: 12** – Height of the symbol referenced by property SymbolID.
SymbolWidth	number > 0	**Optional, default: 12** – Width of the symbol referenced by property SymbolID.
Width	number ≥ 0	**Optional, default: 1** – Line width of the date line.

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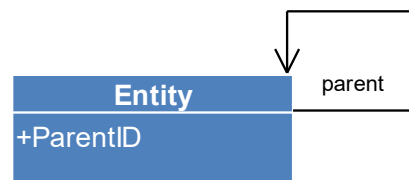
An Entity object defines the properties of a single entity. Entities are shown in a separate table on the right side.

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** – Identifier of the parent of the entity. This serves for setting up a hierarchy of entities. If this property is undefined the current entity will be considered as a root node of the entity hierarchy.

Entity Property Name	Type	Description
		We recommend using only a low number of hierarchy levels and we do not guarantee correct function beyond approx. 100 levels including hierarchy levels created by using Hierarchy-SupplementaryDefinitions.
PM_AllowedRowDrag-Modes	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_RowSymbolColumn-BackgroundColor	string (CSS color value)	**Optional, default: value of property SymbolColumnBackgroundColor of assigned table row definition or option pm_entitiesTableSymbolColumnBackgroundColor** – Determines the color of the symbol column within this table row.
PM_RowSymbolIDs	string[]	**Optional, default: undefined** – Array of identifiers of the symbols to be shown in the

Entity Property Name	Type	Description
		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: value of option pm_defaultEntityRowTooltipTemplateID** – 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.
SuitableActivityIDs	string[]	**Optional, default: undefined** – An array of IDs of those activities to which the entity could be dropped. If the array is defined, then all rows of activities that are not listed in that array will be covered by a half-transparent curtain. If the array is empty, all activity rows will be covered.

Entity Property Name	Type	Description
		If the array is not defined, then all rows are displayed in the normal way without any covering. Also see options pm_suitableActivityOverlayColor and pm_unsuitableActivityOverlayColor and property SuitableActivityIDs of Allocation objects.
SuitableResourceIDs	string[]	**Optional, default: undefined** – An array of IDs of those resources to which the entity could be dropped. 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 and property SuitableResourceIDs of Allocation objects.
TableText	string	**Optional, default: undefined** – Text to display in the table (see also property PM_TableRowDefinitionID).

3.7 HierarchySupplementaryDefinitions

```
        ┌─────────────────────────────────────────┐
        │              Hierarchy                  │
        │            Supplementary                │
        │             Definition                  │
        ├─────────────────────────────────────────┤
        │ +HierarchyLevelSupplementaryDefinitions │
        └─────────────────────────────────────────┘
                          ◆ n

        ┌─────────────────────────────────────────┐
        │            HierarchyLevel               │
        │            Supplementary                │
        │             Definition                  │
        ├─────────────────────────────────────────┤
        │ +GroupingLevelDefinitions               │
        │ +PredefinedGroups                       │
        └─────────────────────────────────────────┘
                          ◆ n

        ┌─────────────────────────────────────────┐
        │            GroupingLevel                │
        │             Definition                  │
        ├─────────────────────────────────────────┤
        │ +GroupingCodeSource                     │
        └─────────────────────────────────────────┘

        ┌─────────────────────────────────────────┐
        │           TableRowDefinition            │
        ├─────────────────────────────────────────┤
        │ +CellDefinitions                        │
        └─────────────────────────────────────────┘
                          ◆ n

        ┌─────────────────────────────────────────┐
        │          TableCellDefinition            │
        │ +TextSource                             │
        │ +Title                                  │
        └─────────────────────────────────────────┘
```
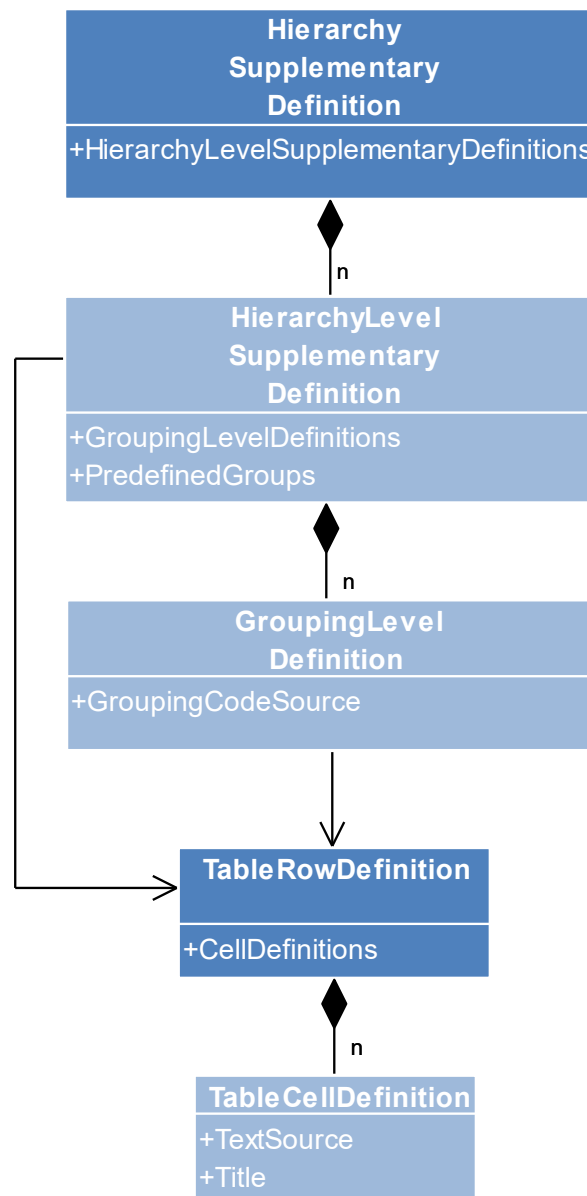
3.7.1   HierarchySupplementaryDefinition

A HierarchySupplementaryDefinition object defines the additional grouping of a complete hierarchy of row objects by using criteria that are taken from property values within the row objects. Each HierarchyLevelSupplementaryDefinition object defines all groupings for a hierarchy level of row objects. The hierarchy levels are already built by using the property ParentID of the row objects of type Activity, Entity, Resource. The ID of a HierarchySupplementaryDefinition object is set into one of the options pm_activity/entity/resourceHierarchySupplementaryDefinitionID.

A first simple definition for the first hierarchy level with only one grouping level only consists of a few properties set:

```
{
  "ID": "HSD1",
  "HierarchyLevelSupplementaryDefinitions": [
    // hierarchy level 0
    {
      "GroupingLevelDefinitions": [
        // first grouping level
        {
          "GroupingCodeSource": "_Grp1", // e.g. referencing an application-
                                          // defined property
          "TableBackgroundColor": "gold"
        }
      ]
    }
  ]
}
```

HierarchySupplementaryDefinition Property Name	Type	Description
ID	string	**Required** – Identifier of this hierarchy definition.
HierarchyLevel-SupplementaryDefinitions	HierarchyLevel-SupplementaryDefinition[]	**Optional, default: undefined** – Array of hierarchy level supplementary definitions. The first object contains definitions for hierarchy level 0, the second defines level 1 and so on. If one hierarchy level should not be grouped additionally, then you can leave the array entry empty either by using an empty object or alternatively undefined/null.

3.7.2   **HierarchyLevelSupplementaryDefinition**

Each HierarchyLevelSupplementaryDefinition object defines additional grouping for a level of row objects. Used in HierarchySupplementaryDefinition objects.

| HierarchyLevel-SupplementaryDefinition Property Name	Type	Description
GroupingLevelDefinitions	GroupingLevel-Definition[]	**Optional, default: undefined** – Array of grouping level definitions. By using more than one GroupingLevelDefinition object in the array you can define multiple grouping criteria for one hierarchy level at once.
InitiallyCollapsed	boolean	**Optionally, default: false** – If this property is set to true, then the table rows in this hierarchy level will initially show collapsed. This property is only effective when the property PM_CollapseState is not set to 0 or 1

HierarchyLevel-SupplementaryDefinition Property Name	Type	Description
		on the affected table row object (Activity/Entity/Resource).

PredefinedGroups	Object[]	**Optionally, default: undefined** – If this property is set to an array, then each object in the array defines one or more predefined groups.
		
		In each object you can define which predefined group(s) to create by including some or all grouping codes up to the current hierarchy level. Additionally, you can set the following graphical properties for the predefined group: PM_MinimumRowHeight, PM_TableColor, PM_TableTextColor.
		
		Example for second hierarchy level:
		
		{
		　"*GroupingCodeSourceOfLevel0*": "a0",
		　"*GroupingCodeSourceOfLevel1*": "b1",
		　"PM_TableColor": "lime"
		}
		
		The property names in italics have to be replaced by the property names in your hierarchy level definitions on levels 0 and 1! This example defines a predefined group with code "b1" within another group with the code "a0". The group "a0" may be predefined or not. If it does not exist, it will be created additionally before creating "b1" using default coloring.
		
		In general, predefined groups are created before any other groups that are determined by grouping codes within the grouped row objects.

TableRowDefinitionID	string	**Optional, default: undefined** – If set then the value overwrites the setting on an affected table row object (Activity/Entity/Resource).

3.7.3   GroupingLevelDefinition

The GroupingLevelDefinition object defines the grouping criteria for all grouping levels of one hierarchy level as seen in a table on screen. The grouping level definition also defines the display of the resulting group lines. Used in a HierarchyLevelSupplementaryDefinition object.

GroupingLevelDefinition Property Name	Type	Description
CodeToTextMap Deprecated!	Object	**Deprecated, please use GroupingCodeToTextMap instead**.
GroupingCodeSource	string	**Optional, default: ""** – Names a property of the objects under consideration, the content of which is used as a grouping criterion.
GroupingCodeToTextMap	Object	**Optional, default: undefined** – If set then the object is used to map the group codes (key) to a long text (value) that will be shown on the table row representing the generated group.
DefaultCode Deprecated!	string	**Deprecated, please use DefaultGroupingCode instead**.
DefaultGroupingCode	string	**Optional, default: ""** – If this property is set, the value serves as a default grouping criterion, if not otherwise defined.
InitiallyCollapsed	boolean	**Optional, default: false** – If this property is set to true, then the generated group rows initially are shown collapsed.
MinimumRowHeight	number > 0	**Optional, default: value of appropriate default option for the table** – Minimum height of the group 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. 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.
TableBackgroundColor Deprecated!	string (CSS color value)	**Deprecated, please use TableColor instead**.
TableColor	string (CSS color value)	**Optional, default: value of the higher group row** – Color for the table row.
TableTextColor	string (CSS color value)	**Optional, default: value of the higher group row** – Color for the table row texts.
TableTextFormat	string	**Optionally, default: undefined** – If this property is set, then text in table on generated group rows is formatted using the value. The string can contain the keyword {{>MapText}} to include the value of the map that is defined by the property CodeToTextMap else will show the group code itself.

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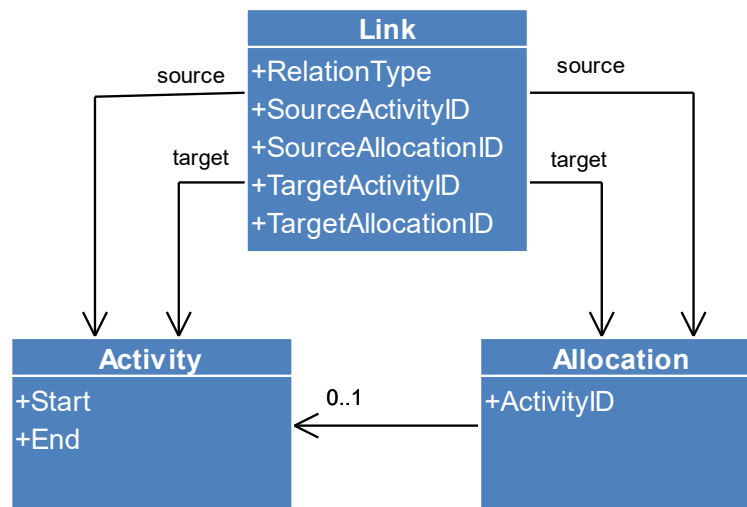
A Link object defines the properties of a single link between activities or or allocations.

Links between activities are shown in activities view if the option pm_linksVisibleInActivitiesView is true. Additionally, links between allocations are shown in activities view if the two options pm_definedAllocationLinksVisibleInActivitiesView and pm_allocationRowsVisibleInActivitiesView are also set to true.

Links between allocations in resources view are shown if the option pm_linksVisibleInResourcesView is true. By default, the links between activities are shown as allocation links, but when the option pm_definedAllocationLinksVisibleInResourcesView is true, then the defined allocation links are shown in resources view instead.

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 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	**Optional, default: value of option pm_defaultLinkRoutingType** – type of the link routing.

Link Property Name	Type	Description
	LinkRoutingType)	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_TargetMarker	number (see enum LinkMarker)	**Optional, default: 1 (FilledArrow)** – allows to change the marker at the end (target) of a link.
PM_TooltipTemplateID	string	**Optional, default: value of option pm_defaultLinkTooltip-TemplateID** – 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 (see enum RelationType)	**Optional, default: 0** – The relation type is used for drawing: 0: Finish-Start, 1: Finish-Finish, 2: Start-Start, 3: Start-Finish, 4: SourceDate-Start, 5: SourceDate-Finish, 8: Finish-TargetDate, 10: Start-TargetDate, 12: SourceDate-TargetDate
SourceActivityID	string	**Optional, default: undefined** – Identifier of the source activity. This property or SourceAllocationID has to be set.
SourceAllocationID	string	**Optional, default: undefined** – Identifier of the source allocation. This property or SourceActivityID has to be set. Please also note the explanations at the beginning of this "Link" chapter.
TargetActivityID	string	**Optional, default: undefined** – Identifier of the source activity. This property or TargetAllocationID has to be set.
TargetAllocationID	string	**Optional, default: undefined** – Identifier of the target allocation. This property or TargetActivityID has to be set. Please also note the explanations at the beginning of this "Link" chapter.

3.9 PeriodHighlighters

Activity	PeriodHighlighter	Resource
+PM_PeriodHighlighterID	+Entries	+PM_PeriodHighlighterID

PeriodHighlighterEntry
+Start
+End
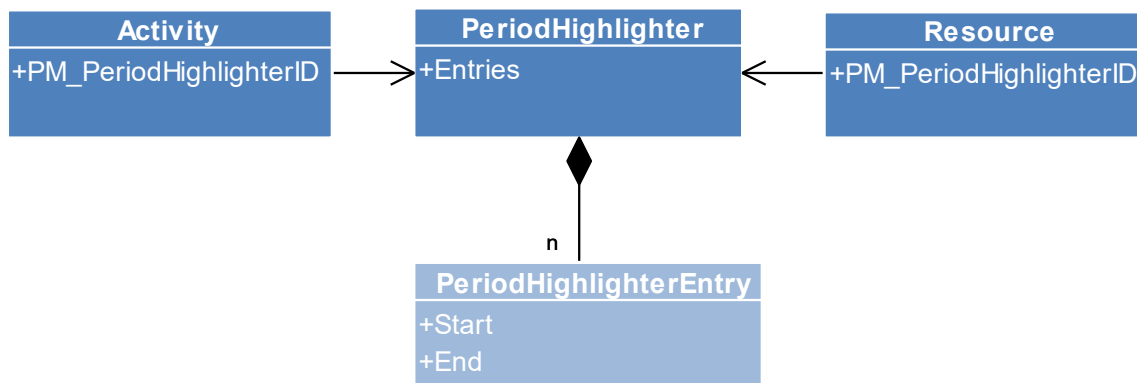
A PeriodHighlighter object is a pure presentation object and defines the properties of a series of time periods that can be shown on each resource row and activity row (see property PM_PeriodHighlighterID there). Each time period can be colored independently and can have a caption. Period highlighters 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.

3.9.1 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.

3.9.2 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.

PeriodHighlighterEntry Property Name	Type	Description
PM_TooltipTemplateID	string	**Optional, default: value of option pm_defaultPeriodHighlighterEntry-TooltipTemplateID** – ID of a tooltip template. The template is used for tooltips that appear on the period highlighter entry.
Start	Date|string	**Required** – Start of the time period.

3.10 Resource
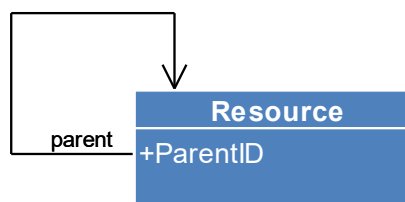
A Resource object defines the properties of a single resource.

| 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.

Resource Property Name	Type	Description
		If this property is undefined the current resource will be considered as a root node of the resource hierarchy. We recommend to use only a low number of hierarchy levels and we do not guarantee correct function beyond approx. 100 levels including hierarchy levels created by using Hierarchy-SupplementaryDefinitions.
PM_AllocationRowsCollapseState	number (see enum CollapseState)	**Optional, default: -1** – Specifies whether the allocation rows of the resource should be expanded or collapsed when displayed. See also callback onCollapseStateChanged and option pm_allocationRowsVisibleInResourcesView. -1: no change 0: display resource row in an expanded way for allocation rows. 1: display resource row in a collapsed way for allocation rows.
PM_AllocationRowsCollapsible	boolean	**Optional, default: value of option pm_defaultResourceAllocationRowsCollapsible** – If set to true, then the row representing this resource row will be interactively collapsible when allocation rows exist.
PM_AllowedRowDragModes	number (see enum RowDragModes)	**Optional, default: value of option pm_defaultResourceAllowedRowDragModes** – This option determines the allowed row drag modes for this resource (these can be overwritten using the callback canDrag).
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 explanation for possible values in enumerations chapter.
PM_CollapseState	number (see enum CollapseState)	**Optional, default: -1** – Specifies whether the row of the resource should be expanded or collapsed when displayed in resources view and eventually in loads view (see property PM_CollapseStateInLoadsView). See also callback onCollapseStateChanged. -1: no change 0: display resource row in an expanded way 1: display resource row in a collapsed way
PM_CollapseStateInLoadsView	number (see enum	**Optional, default: undefined** – Specifies whether the row of the resource should be expanded or

Resource Property Name	Type	Description
	CollapseState)	collapsed when displayed in the loads view. If undefined, then the property PM_CollapseState is used for compatibility reasons. 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: value of option pm_defaultCurveTooltipTemplateID** – 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 explanation for possible values in enumerations chapter.
PM_HasAllocationRows	boolean	**Optional, default: false** – If set to true, then the row representing this resource will be collapsible/expandable for allocation rows even when no allocations exist referencing this resource. This serves for lazy loading.
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

Resource Property Name	Type	Description
		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_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_RowSymbolColumn-BackgroundColor	string (CSS color value)	**Optional, default: value of property SymbolColumnBackgroundColor of assigned table row definition or option pm_symbolColumnBackgroundColor** – Determines the color of the symbol column within this table row.
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: value of option pm_defaultResourceRowTooltipTemplateID** – ID of a tooltip template.

Resource Property Name	Type	Description
		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** – 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: Main** – 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 also options pm_mainViewAreaVisibleInResourcesView/InLoads View and pm_topViewAreaVisibleInResourcesView/InLoadsView.
TableText	string	**Optional, default: undefined** – Text to display in the table row (see also property PM_TableRowDefinitionID).

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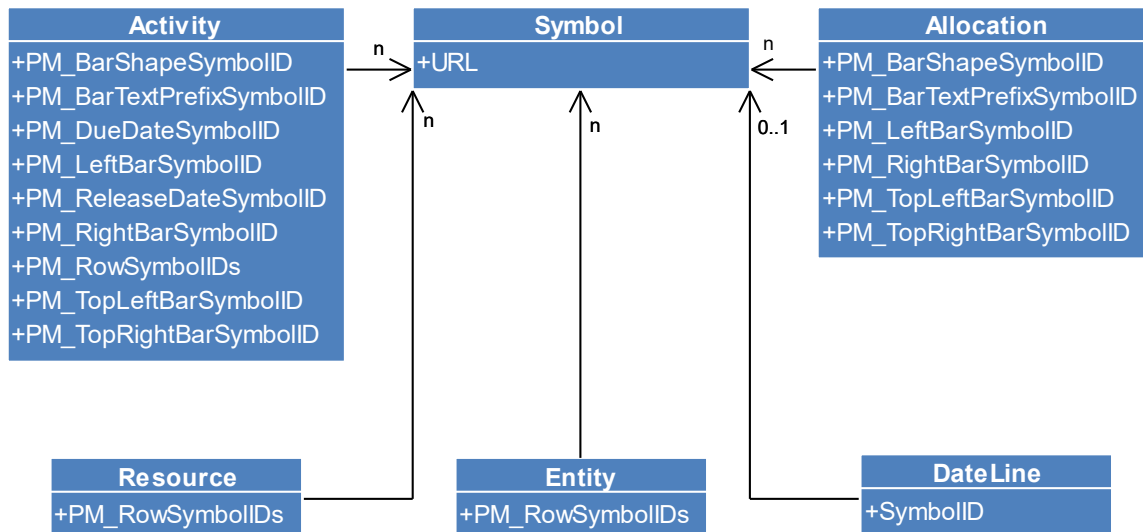
A Symbol object is a pure presentation object and defines the properties of a single symbol. Symbols are used by resources, activities, and allocations. They can be displayed at different locations inside the table and the diagram area.

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.

One hint: If your application needs to be compatible to IE11, then check that SVG symbols are shown correctly scaled in IE11. Otherwise maybe the SVG tag inside the symbol file does not contain the property 'viewBox' that has to be added in that case.

Another hint: If you want to use our method saveAsPDF, then you will have to ensure that your SVG image files do not contain <style> tags, since the contained selectors may change the appearance of the exported SVG content. In a case of existing <style> try to replace them by using style attributes on other tags. We can help if there are problems arising.

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")

Symbol Property Name	Type	Description
		• relative URL (e.g. "images/symbol.svg") – In this case, the anchor path for the symbol directory is the application directory. • Data URI (e.g. 'data:image/svg+xml;base64,…'). See https://en.wikipedia.org/wiki/Data_URI_scheme)

3.12 TableRowDefinitions
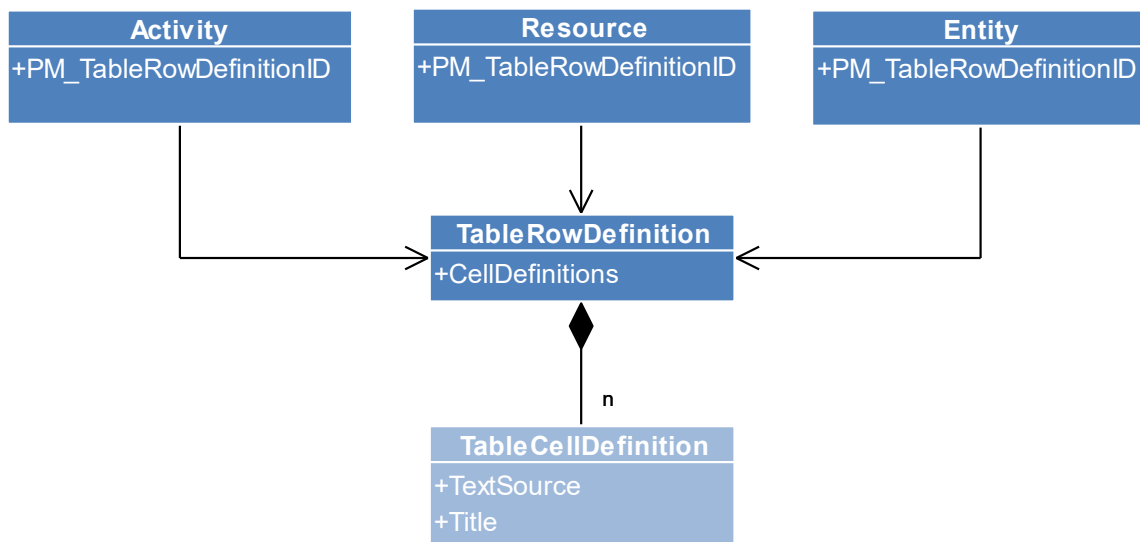
A TableRowDefinition object defines the composition of a table row containing one or more cells. You can reference these objects with the property PM_TableRowDefinitionID of Activity, Allocation, Entity, and Resource objects. There are options for each property defining a default value for the corresponding property (see pm_defaultActivity/Allocation/Entity/ResourceTableRowDefinitionID).

Additionally, 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_tableRowDefinitionIDForTitleInActivities/Resources/LoadsView or pm_tableRowDefinitionIDForTitleInEntitiesTable.

3.12.1 TableRowDefinition

| TableRowDefinition Property Name	Type	Description
BackgroundColor	string (CSS color)	**Optional, default: undefined** – BackgroundColor of the table row definition. The value is only used when not undefined and is overlayed by the background color of the table row defined in the property PM_TableColor of the row object.

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.
SymbolColumnBackgroundColor	string (CSS color value)	**Optional, default: value of option pm_symbolColumnBackgroundColor or pm_entitiesTableSymbolColumnBackgroundColor, resp.** – Determines the color of the symbol column within this table row when the property PM_RowSymbolColumnBackgroundColor is not set on the object where this table row definition is applied to.
TextColor	string (CSS color)	**Optional, default: undefined** – Text color of the table row definition. The value is only used when not undefined and is overlayed by the text color of the table row defined in the property PM_TableTextColor of the row object.

3.12.2 TableCellDefinition

TableCellDefinition Property Name	Type	Description
BackgroundColor	string (CSS color)	**Optional, default: undefined** – If set and property BackgroundColorSource is empty or the referenced property on a row object is empty, then this color overlays the background color of the table row defined in the property PM_TableColor of the row object and the property BackgroundColor of the TableRowDefinition object.
BackgroundColorSource	string	**Optional, default: undefined** – If set to an object's property name and the value of the referenced property on a row object is not empty, then the value there overlays the background color defined by property BackgroundColor and property PM_TableColor of the row object.
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.

| TableCellDefinition Property Name	Type	Description
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 the symbol. If using center or right alignment, the symbol will be overlapped by the text.
TextColor	string (CSS color)	**Optional, default: undefined** – If set and property TextColorSource is empty or the referenced property on a row object is empty, then this color overlays the text color of the table row defined in the property PM_TableTextColor of the row object and the property TextColor of the TableRowDefinition object.
TextColorSource	string	**Optional, default: undefined** – If set to an object's property name and the value of the referenced property on a row object is not empty, then the value there overlays the text color defined by property TextColor and property PM_TableTextColor of the row object.
TextSource	string	**Optional, default: "TableText"** – Property to take the text out of the referencing activity, resource, or entity object.
(Deprecated!) Title	string	**Deprecated, see property TitleText.**
TitleText	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').

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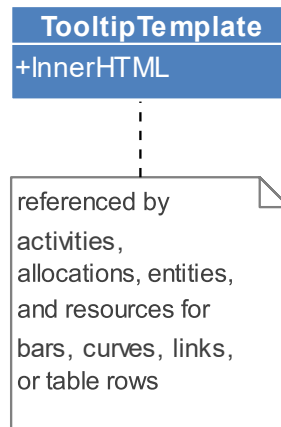
A TooltipTemplate object describes the appearance of a tooltip in the form of an HTML string. This string describes a DOM subtree and contains placeholders with references to the object properties to be displayed. At runtime, the placeholders are replaced by the values of the referenced object properties.

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. Additionally there exists the property PM_TooltipTemplateID on period highlighter entries. All these properties have fallback options named:
   - pm_defaultActivityBar/AllocationBarTooltipTemplateID
   - pm_defaultActivityRow/AllocationRow/ResourceRowTooltipTemplateID
   - pm_defaultCurve/Entity/PeriodHighlighterEntryTooltipTemplateID

Here is an additional hint for designing the HTML markup:

If you fill the markup with a normal table, you will get tooltips that are eventually very wide:
```
<table><tr><td>Long text with several words that should be wrapped into
more than one line</td></tr></table>
```
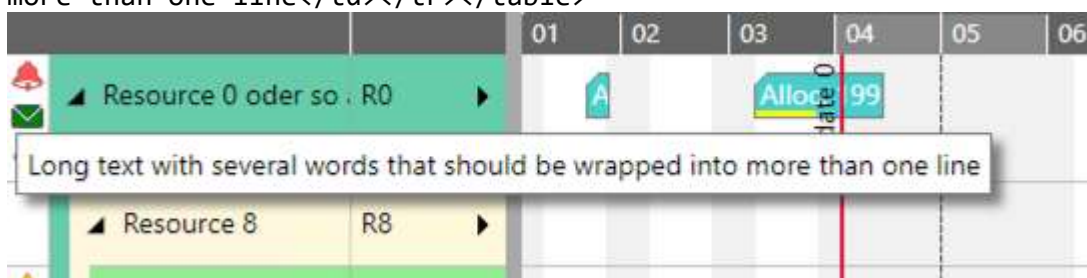

In order to limit the width of the tooltip, you can set some attributes on the tags:

```
<table style="word-wrap: break-word;"><tr><td style="max-width:
100px;">Long text with several words that should be wrapped into more than
one line</td></tr></table>
```
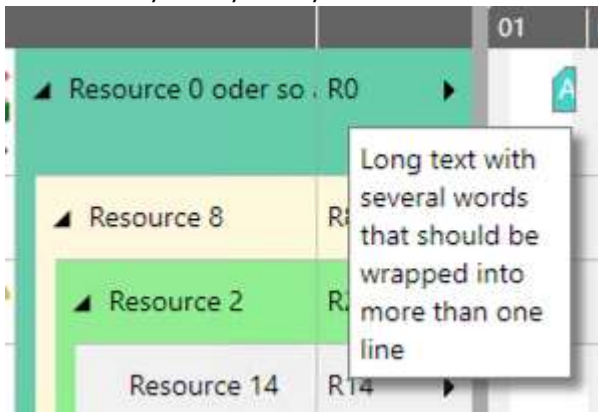

Of course, you are free to use other HTML tags within the markup, also including images by using data URIs.

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>`
		`</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.
		
		The referenced object is the object on which the tooltip will be shown. For period highlighter entries and allocation entries the referenced object is the main object and not the entry object.
		
		It is possible to access related objects by using the following keywords within the property accessor string:
		• On activities: >Parent, >Calendar
		• On resources: >Parent, >Calendar, >LoadCurve, >CapacityCurve

TooltipTemplate Property Name	Type	Description
		• On entities: >Parent • On allocations: >Activity, >Resource • On links: >SourceActivity, >TargetActivity It is also possible to access other objects that are otherwise reachable by the callback arguments of the callback onShowTooltip by using the following keywords at the beginning of the property accessor string: • On allocations: #Entry • On period highlighters: #Entry, #RowObject. • On curves: #Date, #Capacity, #Load, #SingleLoads. For #SingleLoads you have to add *.curveID* to get the curve value for a single curve. If the reached value is an object, you can then access a property value by using a prefixed dot: .propertyName.

4 Widget

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

At first the widget has to be instantiated using a call like $("#ganttDiv").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.

4.1 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
activityRowSortCodePropertyName	string	**Optional, default: "PM_SortCode"** – Name of a data property to be used as sort criteria while sorting activity rows. The values of the addressed property in the activities can contain strings, numbers, or date values. If using interactive vertical row dragging, the specified data property must contain values of number type.

Widget Option Name	Type	Description
		See also option activityRowSortMode.
activityRowSortMode	number (see enum RowSortMode)	**Optional, default: None –** If set to a mode unequal to None activity rows are sorted ascending or descending. It is a prerequisite to use the ascending mode for dragging activity rows vertically. See also option activityRowSortCodePropertyName
additionalDateInterpretedAs-Empty	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.
additionalDateStringInterpreted-AsEmpty	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.
allocationRowSortCodeProperty-Name	string	**Optional, default: "PM_SortCode" –** Name of a data property to be used as sort criteria while sorting allocation rows. The values of the addressed property in the allocations can contain strings, numbers, or date values. See also option allocationRowSortMode.
allocationRowSortMode	number (see enum RowSortMode)	**Optional, default: None –** If set to a mode unequal to None allocation rows are sorted ascending or descending. See also option allocationRowSortCodePropertyName.
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.

Widget Option Name	Type	Description
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. See also options pm_dateLineGridColor, pm_dateLineGridDashArray, pm_dateLineGridWidth.
defaultCalendarID	string	**Optional, default: undefined** – Specifies a default calendar to be used in the widget. If calendars are defined on activities or resource they will override this calendar. If there is no calendar defined on an activity or a resource and if this default calendar ID is undefined, then the calendar is assumed to be one with constantly non-working time only.
defaultUpdateMode	number (see enum [UpdateModes](#))	**Optional, default: Default** – Specifies the default for the parameter updateMode in all update methods.
defaultValuesForActivity-Properties	Object	**Optional, default: null** – Specifies an object with default values used for all properties of Activity objects except ID.
defaultValuesForActivityEntry-Properties	Object	**Optional, default: null** – Specifies an object with default values used for all properties of ActivityEntry objects.
defaultValuesForAllocation-Properties	Object	**Optional, default: null** – Specifies an object with default values used for all properties of Allocation objects except ID.
defaultValuesForAllocationEntry-Properties	Object	**Optional, default: null** – Specifies an object with default values used for all properties of AllocationEntry objects.

Widget Option Name	Type	Description
defaultValuesForEntityProperties	Object	**Optional, default: null** – Specifies an object with default values used for all properties of Entity objects except ID.
defaultValuesForLinkProperties	Object	**Optional, default: null** – Specifies an object with default values used for all properties of Link objects except ID.
defaultValuesForResource-Properties	Object	**Optional, default: null** – Specifies an object with default values used for all properties of Resource objects except ID.
editable	boolean	**Optional, default: true** – If set to false, nothing can be edited.
end	Date|string	**Required** – End date of the considered time area.
entitiesTableTreeVisualization-Mode	number (see enum TreeVisualizationMode)	**Optional, default: ColoredIndentation** – Determines how the tree of objects is visualized in the entities table. See also options pm_entitiesTableTreeViewLineColor and pm_entitiesTableTreeViewLineDashArray.
entitiesTableViewWidth	number	**Optional, default: null (means current table width)** – This setting defines the width of the entities table view when it becomes visible initially.
entitiesTableVisibleInActivities-View	boolean	**Optional, default: false** – This option lets appear/disappear the entities table on the right side in the Activities View.
entitiesTableVisibleInResources-View	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.

Widget Option Name	Type	Description
		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).
entityRowSortCodePropertyName	string	**Optional, default: "PM_SortCode"** – Name of a data property to be used as sort criteria while sorting entity rows. The values of the addressed property in the entities can contain strings, numbers, or date values. If using interactive vertical row dragging, the specified data property must contain values of number type. See also option entityRowSortMode.
entityRowSortMode	number (see enum RowSortMode)	**Optional, default: None** – If set to a mode unequal to None allocation rows are sorted ascending or descending. It is a prerequisite to use the ascending mode for dragging entity rows vertically. See also option entityRowSortCodePropertyName.
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.
firstDayOfWeek	number (allowed values: 0, 1, 2, 3, 4, 5, 6)	**Optional, default: undefined** – Specifies the first day of a week. 0 = Sunday, 1 = Monday, … If defined, this option overwrites the settings of the options "weekNumbering" and "locale", respectively. See also option "weekNumbering".

Widget Option Name	Type	Description
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" = "da-DK", "de" = "de-DE", "en-GB", "en" = "en-US", "es" = "es-ES", "fi" = "fi-FI", "fr" = "fr-FR", "it" = "it-IT", "ja" = "ja-JP", "nl" = "nl-NL", "no" = "no-NO", "pl" = "pl-PL", "pt-BR", "pt" = "pt-PT", "ru" = "ru-RU", "sv" = "sv-SV", "th" = "th-TH", "zh" = "zh-CN")	**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. You must specify the language at least and can append a country. If the country is not known, then the universal language texts for that locale are used automatically as a fallback. Also it is allowed to use both uppercase and lowercase for all letters.
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**.

Widget Option Name	Type	Description
loggingVerboseLevel	number	**Optional, default: 0** – The default verbose level for logging omits loggings of callback triggerings that occur often. If set to 10, then the callbacks canDrag and onShowTooltip are logged additionally. If set to 20, also the callback onDrag is logged. If set to 30, also the callbacks visibilityFilter and compareObjects are logged.
maximumTimeResolutionUnit	string (one of "second", "minute", "hour", "day", "week", "month", "quarter", "year") or number (see enumeration [TimeUnit](#))	**Optional, default: value of option timeStepUnit or "second"** – Unit for maximum time resolution in the time area. Used together with option maximumTimeResolutionUnitFactor. Neither interactively nor by using the method setTimeResolutionForView can the time area display a finer time resolution than defined here. When you set this option and do not set the options timeStepUnit/-Factor, this value here also changes the default value of timeStepUnit! This is done for compatibility reasons.
maximumTimeResolutionUnitFactor	number (>= 1)	**Optional, default: value of option timeStepUnitFactor or 1** – Number of units for maximum time resolution in the time area. Integer values are recommended. Used together with option maximumTimeResolutionUnit. When you set this option and do not set the options timeStepUnit/-Factor, this value here also changes the default value of timeStepUnitFactor! This is done for compatibility reasons.
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 SuitableResourceIDs is supported. When dragging starts, the allowed drag modes are inherited by default from the allocation/activity that is

Widget Option Name	Type	Description
		being dragged directly. 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.
onCollapseStateChangedTriggered ByUpdateCalls	boolean	**Optional, default: true** – Defines whether the callback onCollapseStateChanged is also triggered by calling update methods.
pm_activityBarTopOffsetAndHeig htScaleFactor	number > 0 and <= 10	**Optional, default: 1** – This option modifies the top offset and height of activity bars. It can help to implement a compact layout.
pm_activityBaselineBarsVisible	boolean	**Optional, default: true** – If set to false, no baseline bars are displayed for the activities.
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_activityHierarchySupplementa ryDefinitionID	string	**Optional, default: undefined** – ID of a HierarchySupplementaryDefinition object that will be used to specify grouping parameters for hierarchy of activity objects.

Widget Option Name	Type	Description
pm_activityTableRowDefinitionIDForTitle	string	**Deprecated, see renamed option pm_tableRowDefinitionIDForTitleInActivitiesView.**
pm_allocationBarTopOffsetAndHeightScaleFactor	number > 0 and <= 10	**Optional, default: 1** – This option modifies the top offset and height of allocation bars in activities view and resources view. It can help to implement a compact layout.
pm_allocationRowsVisibleInActivitiesView	boolean	**Optional, default: false** – If set to true, then allocations are shown as own rows below the row of the referenced activity.
pm_allocationRowsVisibleInResourcesView	boolean	**Optional, default: false** – If set to true, then allocations are shown as own rows below the row of the referenced resource.
pm_barSortModeForOptimizedRowDesign	number (see enum [BarSortMode](#))	**Optional, default: StartAndEnd** – Determines how the bars are sorted in a row where the bars are shown vertically optimized, means that they do not overlap each other: **StartAndEnd (0):** Bars are sorted such, that the earlier the start, the earlier the bar is placed. If two bars have the same start, the longer bar is placed first. **ByCompareObjects (1):** Bars are sorted by using the callback compareObjects. **ByCompareObjectsOnSameStart (2):** Bars are sorted as in StartAndEnd, but for two bars with the same start, the result of callback compareFunc is used.
pm_bottomRowMarginInTimeArea	number > 0	**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 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

Widget Option Name	Type	Description
		one for not mentioned view types. Example: { 0 /*activities view*/: "yellow", -1 /*other views*/: null }
pm_clickCallbackTriggeringOnRow InTimeArea	boolean	**Optional, default: false** – When set to true, then the callbacks onClicked and onDoubleClicked are triggered on time area background of a row and in the curve pane area of a row.
pm_curvePanesCollapsibleInReso urcesView	boolean	**Optional, default: true** – Specifies whether the curve panes can be interactively collapsed or expanded.
pm_dateLineCaptionOptimizedPos itioningEnabled	boolean	**Optional, default: false** – Specifies whether the captions of date lines should be arranged optimized to avoid overlapping.
pm_dateLineGridColor	string (CSS color value, e.g. "#ff0000", "rgb(255, 0, 0)", or "red")	**Optional, default: "#505050" (on weekly or daily grid) and "646464" (on automatic grid)** – Color of the date line grid (see also option dateLineGridMode).
pm_dateLineGridDashArray	string	**Optional, default: "4,1" (on weekly or automatic grid) and "2,1" (on daily grid)** – Pattern of dashes and gaps for drawing the date line grid. 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 link is drawn solid.
pm_dateLineGridWidth	number	**Optional, default: 1** – Width of the date line grid in pixels.
pm_defaultActivityAllocation-RowsCollapsible	boolean	**Optional, default: true** – This option holds the default for the property PM_AllocationRowsCollapsible of Activity objects.
pm_defaultActivityAllowedBar-DragModes	number (see enum ActivityBarDragModes)	**Optional, default: ActivityBarDragModes.Drag-Horizontally** – This option holds the default for the attribute PM_Allowed-BarDragModes of Activity objects.
pm_defaultActivityAllowedRow-DragModes	number (see enum RowDragModes)	**Optional, default: RowDragModes.None** – This option holds the default for the attribute

Widget Option Name	Type	Description
		PM_AllowedRowDragModes 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_defaultActivityBarTooltip-TemplateID	string	**Optional, default: ""** – ID of a TooltipTemplate object that will be used when an activity object has set the property PM_BarTooltipTemplateID to "".
pm_defaultActivityCollapsedRow Design	number (see enum RowDesigns)	**Optional, default: 11** – This option holds the default for the attribute PM_CollapsedRowDesign of Activity objects. See explanation for possible values in enumerations chapter.
pm_defaultActivityConstraintSym bolColor	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_defaultActivityExpandedRow Design	number (see enum RowDesigns)	**Optional, default: 11** – This option holds the default for the attribute PM_ExpandedRowDesign of Activity objects. See explanation for possible values in enumerations chapter.
pm_defaultActivityMinimumRow Height	number	**Optional, default: 42** – Default minimum height of the activity rows in pixels. See also Activity.PM_MinimumRowHeight.
pm_defaultActivityProgressBackgr oundColor	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.

Widget Option Name	Type	Description
pm_defaultActivityRowSelectable	boolean	**Optional, default: true** – This option holds the default for the attribute PM_RowSelectable of Activity objects
pm_defaultActivityRowTooltipTemplateID	string	**Optional, default: ""** – ID of a TooltipTemplate object that will be used when an activity object has set the property PM_RowTooltipTemplateID to "".
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_defaultActivityStatusFrameColor	string	**Optional, default: "red"** – This option holds the default color for the property pm_statusFrameColor 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: DragAutoHorOrVer** – This option holds the default for the attribute PM_AllowedBarDragModes of Allocation objects. See also option pm_forcedAllocationAllowedBarDrag-Modes.
pm_defaultAllocationAllowedBarDragModesInActivitiesView	number (see enum AllocationBarDragModes)	**Optional, default: DragHorizontally** – This option holds the default for the attribute PM_AllowedBarDragModes-InAllocationView of Allocation objects. See also option pm_forcedAllocation-AllowedBarDragModesInActivities-View.
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.

Widget Option Name	Type	Description
pm_defaultAllocationBarTooltipTemplateID	string	**Optional, default: ""** – ID of a TooltipTemplate object that will be used when an allocation object has set the property PM_BarTooltipTemplateID to "".
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).
pm_defaultAllocationMinimumRowHeight	number (≥ 0)	**Optional, default: 42** – Default minimum height of the allocation rows in pixels. See also Allocation.PM_MinimumRowHeight.
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_defaultAllocationRowDesign	number (see enum RowDesigns)	**Optional, default: 9** – This option holds the default for the attribute PM_RowDesign of Allocation objects. See explanation for possible values in enumerations chapter.
pm_defaultAllocationRowSelectable	boolean	**Optional, default: true** – This option holds the default for the property PM_RowSelectable of Allocation objects.
pm_defaultAllocationRowTooltipTemplateID	string	**Optional, default: ""** – ID of a TooltipTemplate object that will be used when an allocation object has set the property PM_RowTooltipTemplateID to "".
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.
pm_defaultAllocationStatusFrameColor	string	**Optional, default: "red"** – This option holds the default color for the property pm_statusFrameColor of Allocation objects.
pm_defaultAllocationTableRowDefinitionID	string	**Optional, default: one table cell with value of property TableText of the referenced allocation object in it** – ID of a TableRowDefinition object that

Widget Option Name	Type	Description
		will be used in allocation rows when an allocation object has set the property PM_TableRowDefinitionID to "". See option pm_allocationRowsVisibleInActivitiesView.
pm_defaultAllowedActivityBarDragModes	number (see enum [ActivityBarDragModes](#))	**Deprecated, see renamed option pm_defaultActivityAllowedBarDragModes**
pm_defaultAllowedAllocationBarDragModes	number (see enum [AllocationBarDragModes](#))	**Deprecated, see renamed option pm_defaultAllocationAllowedBarDragModes**
pm_defaultAllowedEntityRowDragModes	number (see enum [RowDragModes](#))	**Deprecated, see renamed option defaultEntityAllowedRowDragModes**
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_defaultEntityRowTooltipTemplateID	string	**Optional, default: ""** – ID of a TooltipTemplate object that will be used when an entity object has set the property PM_ RowTooltipTemplateID to "".
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 "".
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_defaultLinkTooltipTemplateID	string	**Optional, default: ""** – ID of a TooltipTemplate object that will be used when a link object has set the

Widget Option Name	Type	Description
		property PM_ TooltipTemplateID to "".
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_defaultResourceAllocationRowsCollapsible	boolean	**Optional, default: true** – This option holds the default for the property PM_AllocationRowsCollapsible of Resource objects.
pm_defaultResourceAllowedRowDragModes	number (see enum RowDragModes)	**Optional, default: RowDragModes.None** – This option holds the default for the attribute PM_AllowedRowDragModes of Resource objects.
pm_defaultResourceCollapsedRowDesign	number (see enum RowDesigns)	**Optional, default: 11** – This option holds the default for the attribute PM_CollapsedRowDesign of Resource objects. See explanation for possible values in enumerations chapter.
pm_defaultResourceCurveTooltipTemplateID	string	**Optional, default: ""** – ID of a TooltipTemplate object that will be used when a resource object has set the property PM_ CurveTooltipTemplateID to "".
pm_defaultResourceExpandedRowDesign	number (see enum RowDesigns)	**Optional, default: 11** – This option holds the default for the attribute PM_ExpandedRowDesign of Resource objects. See explanation for possible values in enumerations chapter.
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_defaultResourceRowTooltipTemplateID	string	**Optional, default: ""** – ID of a TooltipTemplate object that will be used when a resource object has set

Widget Option Name	Type	Description
		the property PM_ RowTooltipTemplateID to "".
pm_defaultResourceTableRowDef initionID	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_definedAllocationLinksVisibleI nActivitiesView	boolean	**Optional, default: false** – If set to true and the options pm_linksVisibleInActivitiesView and pm_allocationRowsVisibleInActivitiesV iew are also true, then links that are defined between allocations are shown additionally.
pm_definedAllocationLinksVisibleI nResourcesView	boolean	**Optional, default: false** – If set to true and the option pm_linksVisibleInResourcesView is also true, then links that are defined between allocations are shown instead of calculated allocation links defined by activity links.
pm_detailedActivityConstraintSy mbolsEnabled	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_detailedAllocationConstraintS ymbolsEnabled	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_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_entitiesTableCellContentTopOffset	number > 0	**Optional, default: 21** – Top offset for cell content in table cells of entities table. This number is valid for the base line of the first line of text inside the table cell and is only taken into account when it is lower than half of default row height and half of an optional row maximum height.
(Deprecated!) **pm_entitiesTableHeaderBackgroundColor**	string (CSS color value) or Object	**Deprecated, see option pm_entitiesTableHeaderBackgroundColor.**
(Deprecated!) **pm_entitiesTableHeaderColumnSeparatorColor**	string (CSS color value) or Object	**Deprecated, see option pm_entitiesTableHeaderColumnSeparatorColor.**
(Deprecated!) **pm_entitiesTableHeaderHighlightingColor**	string (CSS color value) or Object	**Deprecated, see option pm_entitiesTableHeaderHighlightingColor.**
(Deprecated!) **pm_entitiesTableHeaderTextColor**	string (CSS color value) or Object	**Deprecated, see option pm_entititiesTableHeaderTextColor.**
pm_entitiesTableSymbolColumnBackgroundColor	string (CSS color value)	**Optional, default: "white"** – If set then the symbol column of the entities table will show this color in the background.
pm_entitiesTableSymbolColumnTitleBackgroundColor	string (CSS color value)	**Optional, default: value of option pm_entitiesTableSymbolColumnBackgroundColor** – If set then the symbol column title of the entities table will show this color in the background when the option pm_entitiesTableSymbolColumnTitleVisible is set to true.
pm_entitiesTableSymbolColumnTitleSymbolIDs	string[]	**Optional, default: undefined** – Array of identifiers of the symbols to be shown in the entities table in the title cell of the symbol column. They will only appear when the option pm_entitiesTableSymbolColumnTitleVisible is set to true.

Widget Option Name	Type	Description
		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 later. 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_entitiesTableSymbolColumnTitleVisible	boolean	**Optional, default: false** – If set to true, the symbols specified in the option pm_entitiesTableSymbol-ColumnTitleSymbolIDs will be displayed in the title cell of the symbol column, provided the option pm_entitiesTableSymbolColumnVisible is also set to true. Otherwise, the title cell will have the same color as defined in the pm_entitiesTableTitleBackground-Color option.
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 (≥ 22)	**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 to ensure that the symbols always remain visible as long as the symbol column is visible.
pm_entitiesTableTitleBackgroundColor	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

Widget Option Name	Type	Description
		types. Example see at option pm_calendarGridColor.
pm_entitiesTableTitleColumnSeparatorColor	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_entitiesTableTitleHeight	number > 0	**Optional, default: 60** – Specifies the height of the entities table.
pm_entitiesTableTitleHighlightingColor	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_entitiesTableTitleTextColor	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_entitiesTableTreeViewLineColor	string (CSS color value)	**Optional, default: "black"** – Determines the color of tree view lines in the entities table. See option entitiesTableTreeVisualizationMode.
pm_entitiesTableTreeViewLineDashArray	string (SVG stroke dash array value)	**Optional, default: ""** – Determines the dash array of tree view lines in the entities table. See option entitiesTableTreeVisualizationMode.
pm_entityHierarchySupplementaryDefinitionID	string	**Optional, default: undefined** – ID of a HierarchySupplementaryDefinition object that will be used to specify grouping parameters for hierarchy of entity objects.
(Deprecated!) **pm_entityTableRowDefinitionIDForTitle**	string	**Deprecated, see option pm_tableRowDefinitionIDForTitleInEntitiesTable.**
pm_forcedActivityAllowedBar-DragModes	number|null (see enum [ActivityBarDragModes](#))	**Optional, default: null** – If set to a number, then this value overrides any setting in option pm_defaultActivityAllowedBarDragModes, Activity property

Widget Option Name	Type	Description
		PM_AllowedBarDragModes, property allowedDragModes in canDrag callback. This option is only important when your application cannot use the other mentioned settings.
pm_forcedActivityAllowedRow-DragModes	number (see enum [RowDragModes](#))	**Optional, default: null** – If set to a number, then this value overrides any setting in option pm_defaultActivityAllowedRow-DragModes, Activity property PM_AllowedRowDragModes, property allowedDragModes in canDrag callback. This option is only important when your application cannot use the other mentioned settings.
pm_forcedAllocationAllowedBar-DragModes	number|null (see enum [AllocationBarDragModes](#))	**Optional, default: null** – If set to a number, then this value overrides any setting in option pm_defaultAllocationAllowedBarDrag-Modes, Allocation property PM_AllowedBarDragModes, property allowedDragModes in canDrag callback. This option is only important when your application cannot use the other mentioned settings.
pm_forcedAllocationAllowed-BarDragModesInActivitiesView	number|null (see enum [AllocationBarDragModes](#))	**Optional, default: null** – If set to a number, then this value overrides any setting in option pm_defaultAllocationAllowedBarDrag-ModesInActivitiesView, Allocation property PM_AllowedBarDragModesInActivitiesView, property allowedDragModes in canDrag callback. This option is only important when your application cannot use the other mentioned settings.
pm_forcedEntityAllowedRow-DragModes	number (see enum [RowDragModes](#))	**Optional, default: null** – If set to a number, then this value overrides any setting in option pm_defaultEntityAllowedRow-DragModes, Entity property PM_AllowedRowDragModes, property allowedDragModes in canDrag callback. This option is only important when your application cannot use the other mentioned settings.
pm_forcedResourceAllowedRow-DragModes	number (see enum [RowDragModes](#))	**Optional, default: null** – If set to a number, then this value overrides any setting in option

Widget Option Name	Type	Description
		pm_defaultResourceAllowedRowDragModes, Resource property PM_AllowedRowDragModes, property allowedDragModes in canDrag callback. This option is only important when your application cannot use the other mentioned settings.
pm_ignoreCalendarOnActivityBarI nteractions	boolean	**Optional, default: false** – If set to true, then the activity calendar is not taken into account when dragging an activity bar.
pm_ignoreCalendarOnAllocationB arInteractions	boolean	**Optional, default: false** – If set to true, then the resource calendar is not taken into account when dragging an allocation bar.
pm_linesShownInLoadCurvePanes	boolean	**Option, default: false** – If this option is set to true, in all load curve panes horizontal auxiliary lines are displayed for each tick mark of the numerical scales in the table. These lines help the user to read the curve values.
pm_linksVisibleInActivitiesView	boolean	**Option, default: true** – If set to false, the activities view does not show links. When true, it shows at least activity links. See also option pm_definedAllocationLinksVisibleInActivitiesView.
pm_linksVisibleInResourcesView	boolean	**Option, default: false** – If set to true, the resources view shows links. See also option pm_definedAllocationLinksVisibleInResourcesView.
pm_mainViewAreaVisible	boolean	**Deprecated, see renamed option pm_mainViewVisibleInResourcesView**.
pm_mainViewAreaVisibleInActivitiesView	boolean	**Optional, default: true** – When set to false, then in activities view the main view area is invisible. The main view area contains the rows for activities with PM_ViewArea set to Main. If pm_topViewAreaVisibleInActivitiesView is also false, then the main view area will be visible nevertheless.
pm_mainViewAreaVisibleInLoadsView	boolean	**Optional, default: true** – When set to false, then in loads view the main view area is invisible. The main view area contains the rows for resources with PM_ViewArea set to Main. If pm_topViewAreaVisibleInLoadsView is also false, then the main view area will be visible nevertheless.

Widget Option Name	Type	Description
pm_mainViewAreaVisibleIn-ResourcesView	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 Main. If pm_topViewAreaVisibleInResources-View 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_maximumTopViewAreaHeight Ratio	number (-0.8 ≤ n ≤ 0.8)	**Optional, default: 0.5** – If positive, this value determines the maximum height of the top view area expressed as a fraction of the full view height. If negative, the absolute value instead determines the maximum height of the common view area (so the common view area can be used for unassigned resource allocations alternatively. Vertical scroll bars are shown in both view areas if necessary.
pm_objectHighlightFlashingEnabled	boolean	**Optional, default: true** – Specifies whether or not the frame displayed around an object that has been scrolled to by using the method scrollToObject should flash or around objects after using the method highlightObjects.
pm_objectHighlightingColor	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 or around objects after using the method highlightObjects. 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_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

Widget Option Name	Type	Description
		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 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_progressBarHeight	number >= 0	**Optional, default: 4** – Determines the height of all progress bars within allocation bars and activity bars.
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

Widget Option Name	Type	Description
		ReleaseDate and a DueDate, a line will be displayed to visually connect both dates:
pm_resourceHierarchySupplementaryDefinitionID	string	**Optional, default: undefined** – ID of a HierarchySupplementaryDefinition object that will be used to specify grouping parameters for hierarchy of resource objects
pm_resourceHierarchySupplementaryDefinitionIDInLoadsView	string	**Optional, default: undefined** – ID of a HierarchySupplementaryDefinition object that will be used to specify grouping parameters for hierarchy of resource objects in loads view.
pm_resourceTableRowDefinitionIDForTitle	string	**Deprecated, see renamed option pm_tableRowDefinitionIDForTitleInResourcesView.**
pm_scrollOffsetsChangedCallbackTimeDelay	number >= 0	**Optional, default: 500** – This value determines the time delay in milliseconds for triggering the callbacks onVerticalScrollOffsetChanged and onTimeAreaViewParametersChanged.
pm_scrollToObjectAnimation-Enabled	boolean	**Optional, default: false** – If set to true, then scrolling to the target position is animated when using the method scrollToObject.
pm_scrollToObjectHighlight-FlashingEnabled	boolean	**Deprecated, see renamed option pm_objectHighlightFlashingEnabled.**
pm_scrollToObjectHighlighting-Color	string (CSS color value) or Object	**Deprecated, see renamed option pm_objectHighlightingColor.**
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

Widget Option Name	Type	Description
		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 > 0	**Optional, default: 5** – Vertical distance between two bars in pixels. See also pm_topRowMarginInTimeArea and pm_bottomRowMarginInTimeArea. 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_symbolColumnBackgroundColor	string CSS color value)	**Optional, default: "white"** – If set then the symbol column of the activities/resources table will show this color in the background.
pm_symbolColumnTitleBackgroundColor	string (CSS color value)	**Optional, default: value of option pm_symbolColumnBackgroundColor** – If set then the symbol column title of the activities/resources table will show this color in the background when the option pm_symbolColumnTitleVisible is set to true.
pm_symbolColumnTitleSymbolIDs	string[]	**Optional, default: undefined** – Array of identifiers of the symbols to be shown in the table in the title cell of the symbol column. They will only appear when the option pm_symbolColumnTitleVisible is set to true and option titleText is not set, so that the table title shows columns. 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.

Widget Option 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 later. 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_symbolColumnTitleVisible	boolean	**Optional, default: false** – If set to true, the symbols specified in the option pm_symbolColumnTitle-SymbolIDs will be displayed in the title cell of the symbol column, provided the option pm_symbolColumnVisible is also set to true. Otherwise, the title cell will have the same color as defined in the pm_tableTitle-BackgroundColor option.
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 (≥ 22)	**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 to ensure that the symbols always remain visible as long as the symbol column is visible.
pm_tableCellContentTopOffset	number > 0	**Optional, default: 21** – Top offset for cell content in table cells of left table. This number is valid for the base line of the first line of text inside the table cell and is only taken into account when it is lower than half of default row height and half of an optional row maximum height.
(Deprecated!) **pm_tableHeaderBackgroundColor**	string (CSS color value) or Object	**Deprecated, see option pm_tableTitleBackgroundColor.**
(Deprecated!) **pm_tableHeaderColumnSeparator Color**	string (CSS color value) or Object	**Deprecated, see option pm_tableTitleColumnSeparatorColor.**
(Deprecated!) **pm_tableHeaderHighlightingColor**	string (CSS color value)	**Deprecated, see option pm_tableTitleHighlightingColor.**

Widget Option Name	Type	Description
	or Object	
(Deprecated!) **pm_tableHeaderTextColor**	string (CSS color value) or Object	**Deprecated, see option pm_tableTitleTextColor.**
pm_tableRowDefinitionIDForTitleInActivitiesView	string	**Optional, default: value of option pm_defaultActivityTableRowDefinitionID** – 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_tableRowDefinitionIDForTitleInEntitiesTable	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_tableRowDefinitionIDForTitleInLoadsView	string	**Optional, default: value of option pm_tableRowDefinitionIDForTitleInResourcesView or, when undefined, 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 widths for the TableRowDefinition object that is referenced here.
pm_tableRowDefinitionIDForTitleInResourcesView	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 widths for the TableRowDefinition object that is referenced here.
pm_tableTitleAndTimescaleHeight	number >= 0	**Optional, default: 60** – Specifies the height of the left table and the timescale.
pm_tableTitleBackgroundColor	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

Widget Option Name	Type	Description
		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_tableTitleColumnSeparatorColor	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_tableTitleHighlightingColor	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_tableTitleTextColor	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 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.

Widget Option Name	Type	Description
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.
(Deprecated! Use option pm_timescaleInteractionMode instead.) 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.
pm_timescaleInteractionMode	number (see enum TimescaleInteractionModes)	**Optional, default: TimescaleInteractionModes.Default** – Specifies which interactions are allowed on the timescale.
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

Widget Option Name	Type	Description
		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 > 0	**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	**Deprecated, see renamed option pm_topViewAreaVisibleInResources-View.**
pm_topViewAreaVisibleInActivitiesView	boolean	**Optional, default: false** – If set to true, then activities in the activities view are shown in a separate top view area, that have the attribute PM_ViewArea set to Top. See also options pm_mainViewAreaVisibleInActivitiesView and pm_maximumTopViewArea-HeightRatio.
pm_topViewAreaVisibleInLoadsView	boolean	**Optional, default: false** – If set to true, then resources in the loads view are shown in a separate top view area, that have the attribute PM_ViewArea set to Top. See also options pm_mainViewAreaVisibleInLoadsView and pm_maximumTopViewArea-HeightRatio.
pm_topViewAreaVisibleInResourcesView	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 options

Widget Option Name	Type	Description
		pm_mainViewAreaVisibleInResources View and pm_maximumTopViewArea-HeightRatio.
pm_treeViewLineColor	string (CSS color value)	**Optional, default: "black"** – Determines the color of tree view lines in the left table. See option treeVisualizationMode.
pm_treeViewLineDashArray	string (SVG stroke dash array value)	**Optional, default: ""** – Determines the dash array of tree view lines in the left table. See option treeVisualizationMode.
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.
pm_watermarkOpacity	number (>= 0.0, <= 1.0)	**Optional, default: 0.2** – Opacity of the watermark. See also option pm_watermarkSymbolID.
pm_watermarkSymbolID	string	**Optional, default: undefined** – Identifier of the symbol to be shown in the time area of the Gantt chart. The symbol is stretched while maintaining the ratio between width and height of the symbol so that it is as large as possible. See also option pm_watermarkOpacity. ATTENTION: Option not usable with IE11!
resetValueForDifferentialUpdate	any	**Optional, default: null** – Determines a value that will be replaced by "undefined" for differential updates when using an update method with flag UpdateModes.DifferentialValues set.
resourceRowSortCodePropertyName	string	**Optional, default: "PM_SortCode"** – Name of a data property to be used as sort criteria while sorting resource rows. The values of the addressed property in the resources can contain strings, numbers, or date values. If using interactive vertical row dragging, the specified data property must contain values of number type.

Widget Option Name	Type	Description
		See also option resourceRowSortMode.
resourceRowSortMode	number (see enum [RowSortMode](#))	**Optional, default: None –** If set to a mode unequal to None resource rows are sorted ascending or descending. It is a prerequisite to use the ascending mode for dragging resource rows vertically. See also option resourceRowSortCodePropertyName.
start	Date|string	**Required** – Start of the considered time area.
tableViewWidth	number	**Optional, default: null (means table width)** – This option defines the width of the table view in all views. A change to the vertical splitter is not changing this option.
tableViewWidthInActivitiesView	number	**Optional, default: null (means table width)** – This option defines the width of the table view in activities view. A change to the vertical splitter is not changing this option.
tableViewWidthInLoadsView	number	**Optional, default: null (means table width)** – This option defines the width of the table view in loads view. A change to the vertical splitter is not changing this option.
tableViewWidthInResourcesView	number	**Optional, default: null (means table width)** – This option defines the width of the table view in resources view. A change to the vertical splitter is not changing this option.
tableViewWidthsSynchronized	boolean	**Optional, default: true** – This option defines whether an interactive change of the table view width sets the view width of all views or not.
(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

Widget Option Name	Type	Description
		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", "week", "month", "quarter", "year") or number (see enumeration [TimeUnit](#))	**Optional, default: value of option maximumTimeResolutionUnit or "second"** – Unit for time steps on horizontal drag interactions of bars. Used together with option timeStepUnitFactor. When using a time unit "day" or above, the stepping is done day-by-day without concerning nonworking times within the period. When you set this option and do not set the options maximumTime-ResolutionUnit/-Factor, the value here also changes the default value of maximumTimeResolutionUnit! This is done for compatibility reasons. **Attention!** Currently, the dates of the bars as well as the dates in the calendar must not be defined finer than this unit together with the option timeStepUnitFactor indicate. Otherwise, unexpected jumps may occur when moving bars.
timeStepUnitFactor	number (>= 1)	**Optional, default: value of option maximumTimeResolutionUnitFactor or 1** – Number of units for a single time step on horizontal drag interactions of bars. Used together with option timeStepUnit. Integer values are recommended. When you set this option and do not set the options maximumTime-ResolutionUnit/-Factor, the value here also changes the default value of maximumTimeResolutionUnitFactor! This is done for compatibility reasons. **Attention!** Currently, the dates of the bars as well as the dates in the calendar must not be defined finer than this factor together with the option timeStepUnit indicate.

Widget Option Name	Type	Description
		Otherwise, unexpected jumps may 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).
tooltipDelay	number	**Optional, default: 500** – This option determines delay in milliseconds until a tooltip gets visible.
treeVisualizationMode	number (see enum TreeVisualizationMode)	**Optional, default: ColoredIndentation** – Determines how the tree of objects is visualized in the left table. See also options pm_treeViewLineColor and pm_treeViewLineDashArray.
version	string	**Read only** – This option holds the version number of the widget set by

Widget Option Name	Type	Description
		NETRONIC. Usually it is formatted using the semantic versioning format "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: activities view, resources view, or loads view.
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 week numbering scheme (ISO8601: January 4 must be in the first week of the year, USA: January 1 must be in the first week of the year). This option determines also the first day of the week (ISO8601: Monday, USA: Sunday). If set to null, then the implicit setting of the option "locale" is used. And that setting can also be overwritten by the option "firstDayOfWeek" (see there for more details).
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

Widget Option Name	Type	Description
		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.

4.2 Callbacks

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

When we speak of a Promise object within of the callbacks, you can use a standard Promise object or a jQuery Promise (see http://api.jquery.com/promise/).

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/entity or touches an activity/allocation/entity with a finger.<br><br>Profile:<br>```
function (args)
args = {
  "objectType" : ObjectType,
  "object" : Object,
  "visualType" : VisualType,
  "entry"¹ : AllocationEntry,
  "entryIndex"¹ : number,
  "allowedDragModes" :  ActivityBarDragModes
     |AllocationBarDragModes
     |RowDragModes, //[in/out]
  "selectedObjects" : Object[]|undefined,
  "startPropertyName" : string²,
  "endPropertyName" : string²
}
```<br><br>If the application sets args.allowedDragModes to None, then no dragging will be possible. The same is possible by setting option pm_forcedActivity/AllocationAllowedBarDragModes or pm_forcedActivity/Entity/Resource-AllowedRowDragModes, resp., to None. On input, args.allowedDragModes contains the value of the property PM_Allowed(Row/Bar)DragModes of the object to drag. |

---

[1] Available only if objectType == ObjectType.Allocation or if visualType == VisualType.PeriodHighlighter.
[2] Only set if touching/dragging an activity bar.

Callback Name	Type	Description
		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. If the mouse touches a date symbol or bar of an activity, then the properties start/endPropertyName contain the name of the property to be modified when a dragging or dropping the symbol or bar, resp. 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]`
		`}`

compareActivities	Function	**Optional, default: undefined** – This function is called when an activity is added or when its parent is changed during its update. Currently, only objects that appear as table rows can be sorted using this callback. The comparison is always performed only between siblings of same object type. The result will determine the sorting of the rows in the view.
		
		Profile:
		`function (args)`
		`args = {`
		`  "objectType" : ObjectType.Activity,`
		`  "objectA" : Object,`
		`  "objectB" : Object,`
		`  "viewType" : ViewType,`
		`  "hierarchySupplementaryDefinitionID"`[3]` : string,`
		`"hierarchyLevelSupplementaryDefinitionIndex"`Fehler! Textmarke nicht definiert.` : number,`
		`  "groupingLevelDefinitionIndex"`Fehler! Textmarke nicht definiert.` : number,`
		`  "groupingCodeA"`Fehler! Textmarke nicht definiert.` : string,`
		`  "groupingCodeB"`Fehler! Textmarke nicht definiert.` : string,`
		`  "isALowerThanB": boolean //[in/out]`
		`}`

---

[3] Only set when this callback is referencing grouping rows. The properties objectA and objectB then always are null.

Callback Name	Type	Description
		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. In case of grouping rows objectA and objectB are null and instead groupingCodeA and groupingCodeB together with the properties hierarchySupplementaryDefinitionID, hierarchyLevelSupplementaryDefinitionIndex, and groupingLevelDefinitionIndex are set.

compareAllocations	Function	**Optional, default: undefined** – This function is called when an allocation is added or when its referenced activity row or resource row is changed during its update and allocation rows are visible. Currently, only objects that appear as table rows can be sorted using this callback. The comparison is always performed only between siblings of same object type. The result will determine the sorting of the rows in the view.
		
		Profile:
		`function (args)`
		`args = {`
		`  "objectType" : ObjectType.Allocation,`
		`  "objectA" : Object,`
		`  "objectB" : Object,`
		`  "viewType" : ViewType,`
		`  "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.

compareEntities	Function	**Optional, default: undefined** – This function is called when an entity is added or when its parent is changed during its update. The comparison is always performed only between siblings of same object type. The result will determine the sorting of the rows in the view.
		
		Profile:
		`function (args)`
		`args = {`
		`  "objectType" : ObjectType.Entity,`
		`  "objectA" : Object,`
		`  "objectB" : Object,`
		`  "viewType" : ViewType,`
		`  "hierarchySupplementaryDefinitionID"`[4]` : string,`

---

[4] Only set when this callback is referencing grouping rows. The properties objectA and objectB then always are null.

Callback Name	Type	Description
		`"hierarchyLevelSupplementaryDefinitionIndex"`[Fehler! Textmarke nicht definiert.] `: number,` `"groupingLevelDefinitionIndex"`[Fehler! Textmarke nicht definiert.] `: number,` `"groupingCodeA"`[Fehler! Textmarke nicht definiert.] `: string,` `"groupingCodeB"`[Fehler! Textmarke nicht definiert.] `: string,` `"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. In case of grouping rows objectA and objectB are null and instead groupingCodeA and groupingCodeB together with the properties hierarchySupplementaryDefinitionID, hierarchyLevelSupplementaryDefinitionIndex, and groupingLevelDefinitionIndex are set.
compareObjects **DEPRECATED!**	Function	**Deprecated for performance reasons, please see options compareActivities/Allocations/Entities/ Resources.**

compareResources	Function	**Optional, default: undefined** – This function is called when a resource is added or when its parent is changed during its update. The comparison is always performed only between siblings of same object type. The result will determine the sorting of the rows in the view.
		
		Profile:
		`function (args)` `args = {` `"objectType" : ` <u>`ObjectType`</u>`.Resource,` `"objectA" : Object,` `"objectB" : Object,` `"viewType" : ViewType,` `"hierarchySupplementaryDefinitionID"`[5] `: string,`
		
		`"hierarchyLevelSupplementaryDefinitionIndex"`[Fehler! Textmarke nicht definiert.] `: number,` `"groupingLevelDefinitionIndex"`[Fehler! Textmarke nicht definiert.] `: number,` `"groupingCodeA"`[Fehler! Textmarke nicht definiert.] `: string,` `"groupingCodeB"`[Fehler! Textmarke nicht definiert.] `: string,` `"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.

[5] Only set when this callback is referencing grouping rows. The properties objectA and objectB then always are null.

Callback Name	Type	Description
		In case of grouping rows objectA and objectB are null and instead groupingCodeA and groupingCodeB together with the properties hierarchySupplementaryDefinitionID, hierarchyLevelSupplementaryDefinitionIndex, and groupingLevelDefinitionIndex are set.
determineGroupingCode	Function	**Optional, default: undefined** – This function is called to determine grouping information like grouping code and long text. Profile: ```function (args)\nargs = {\n "objectType" : ObjectType,\n "object" : Object,\n "hierarchySupplementaryDefinitionID" : string,\n "hierarchyLevelSupplementaryDefinitionIndex"\n: number,\n "groupingLevelDefinition" : Object,\n "groupingLevelDefinitionIndex" : number,\n "code" : string // [in/out],\n "text" : string // [out]\n}```
onClicked	Function	**Optional, default: undefined** – This function is called when an object is clicked by the user. Profile: ```function (args)\nargs = {\n "objectType" : ObjectType,\n "object" : Object|null,\n "visualType" : VisualType,\n "date" : Date|undefined, // date at mouse cursor\n "entry"[1] : AllocationEntry|\n PeriodHighlighterEntry,\n "entryIndex"[1] : number,\n "curve" : Curve, // Only available when\n clicked on a curve; the "object" parameter\n will then hold the corresponding resource\n "periodHighlighter" : PeriodHighlighter,\n // Only available when clicked on a\n PeriodHighlighterEntry; the "object"\n parameter will then hold the corresponding\n resource/activity\n "cellIndex" : number, // Only available when\n clicked on a table cell; zero-based index\n of the cell.\n "symbolIndex" : number, // Only available when\n clicked on a symbol; zero-based index\n of the symbol.}\n "hierarchySupplementaryDefinitionID"[6] : string,\n "hierarchyLevelSupplementaryDefinitionIndex"[6]\n: number,\n "groupingLevelDefinitionIndex"[6] : number,\n "groupingCodes"[6] : string[]``` In case of a grouping row, object is null and instead groupingCodes together with the properties hierarchySupplementaryDefinitionID,

---

[6] Only set when this callback is referencing a grouping row. The object property then always is null.

Callback Name	Type	Description
		hierarchyLevelSupplementaryDefinitionIndex, and groupingLevelDefinitionIndex are set.
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 resource, activity or entity row
		• by automatic row expansion when dragging objects
		• by using the method scrollToObject
		• by setting the attribute PM_CollapseState on a resource, an activity, or entity object and option onCollapseStateChangedTriggeredByUpdate Calls is not set to false.
		
		Profile:
		`function (args)`
		`args = {`
		`  "objectType" : `<u>`ObjectType`</u>`,`
		`  "object" : Object|null,`
		`  "newCollapseState" : `<u>`CollapseState`</u>`,`
		`  "interactively" : boolean,`
		`  "isForAllocationRows" : boolean,`
		`  "hierarchySupplementaryDefinitionID"`[8]` : string,`
		`  "hierarchyLevelSupplementaryDefinitionIndex"`[8]
		`: number,`
		`  "groupingLevelDefinitionIndex"`[8]` : number,`
		`  "groupingCodes"`[8]` : string[],`
		`  "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" : `<u>`ObjectType`</u>`,`

Callback Name	Type	Description
| | | ```
  "object" : Object,
  "newCollapseState" : CollapseState,
  "resource" : Resource|undefined,
  "promise" : Promise [out]
}
```<br><br>The property "resource" is only set, when the object is not the resource itself.<br><br>The application can update the property PM_CurveCollapseState of the object if needed.<br><br>If the application sets the promise attribute, then the update of the DOM is delayed until the promise is resolved. |
| **(Deprecated!)**<br>**onDetermineColumnDefinitions** | Function | **Optional, default: undefined** – Please use object TableRowDefinition instead for same purpose.<br><br>This function is called to determine the definitions of the table columns.<br><br>Profile:<br>```
function (args)
 args = {
   "tableType" : TableType
   "level" : number,
   "objectType" : ObjectType,
   "object" : Object,
   "columns" : Object[], //[in/out]
   "hasColumnTitles" : boolean //[in/out]
}
```<br>The content of args.columns can be changed or replaced. For each column there is an object as follows:<br><br>```
{
  "initialWidth" : number, /*in pixels*/
  "horizontalTextAlignment" :
                  HorizontalAlignment,
  "textSource" : string /*property name*/,
  "title" : string,
  "wrapMode" : TextWrapMode
}
```<br>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.<br><br>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. |

Callback Name	Type	Description
		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|null,```
		``` "visualType" : VisualType,```
		``` "date" : Date|undefined, // date at mouse cursor```
		``` "entry"`[1]` : AllocationEntry|```
		``` PeriodHighlighterEntry,```
		``` "entryIndex"`[1]` : number,```
		``` "periodHighlighter" : PeriodHighlighter,```
		``` // Only available when clicked on a```
		``` PeriodHighlighterEntry; the "object"```
		``` parameter will then hold the corresponding```
		``` resource/activity```
		``` "cellIndex" : number, // Only available when```
		``` clicked on a table cell; zero-based index```
		``` of the cell.```
		``` "symbolIndex" : number, // Only available when```
		``` clicked on a symbol; zero-based index```
		``` of the symbol.```
		``` "hierarchySupplementaryDefinitionID"`[6]` : string,```
		``` "hierarchyLevelSupplementaryDefinitionIndex"`[6]`: number,```
		``` "groupingLevelDefinitionIndex"`[6]` : number,```
		``` "groupingCodes"`[6]` : { key: value }```
		```}```
		
		 In case of a grouping row, object is null and instead groupingCodes together with the properties hierarchySupplementaryDefinitionID, hierarchyLevelSupplementaryDefinitionIndex, and groupingLevelDefinitionIndex are set.

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

Callback Name	Type	Description

		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]
		`                |`[RowDragModes]`,`
		`    "rowInsertionMode" : `[RowInsertionMode]`,`
		`    "newRowObjectType" : `[ObjectType]`,`
		`    "newRowObject" : Object,`
		`    "newRowObjectIsSuitableActivity" : boolean,`
		`    "newRowObjectIsSuitableResource" : boolean,`
		`    "newStart" : Date, // not for date lines`
		`    "newEnd" : Date, // not for date lines`
		`    "newDate" : Date, // only for date lines,`
		`    "startPropertyName" : string`[2]`,`
		`    "endPropertyName" : string`[2]`,`
		`    "dropAllowed" : boolean [out],`
		`    "cancel" : boolean [out]`
		`}`
		
		If a date symbol or bar of an activity is dragged, then the properties start/endPropertyName contain the name of the property to be modified when a dragging or dropping the symbol or bar, resp.
		
		If a row is dragged, the property rowInsertionMode tells about the current insertion mode relative to the object in property newRowObject.

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]
		`                |`[RowDragModes]`,`

Callback Name	Type	Description

		`"startPropertyName" : string`[2]`,`
		`"endPropertyName" : string`[2]
		`}`
		
		If a date symbol or bar of an activity was dragged, then the properties start/endPropertyName contain the name of the property to be modified when a dragging or dropping the symbol or bar, resp.

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)`
		`args = {`
		`    "objectType" : `[ObjectType]`,`
		`    "object" : Object,`
		`    "visualType" : `[VisualType]`,`
		`    "entry"`[1]` : AllocationEntry,`
		`    "entryIndex"`[1]` : number,`
		`    "dragMode" : `[ActivityBarDragModes]
		`                 |`[AllocationBarDragModes]
		`                 |`[RowDragModes]`,`
		`    "startPropertyName" : string`[2]`,`
		`    "endPropertyName" : string`[2]`,`
		`    "cancel" : boolean [out]`
		`}`
		
		If a date symbol or bar of an activity will be dragged, then the properties start/endProperty-Name contain the name of the property to be modified when a dragging or dropping the symbol or bar, resp.

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 Promise object into args.promise, 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]
		`                 |`[RowDragModes]`,`
		`    "rowInsertionMode": `[RowInsertionMode]`,`
		`    "newRowObjectType" : `[ObjectType]`,`

Callback Name	Type	Description
		``
    "newRowObject" : Object,
    "newStart" : Date,
    "newEnd" : Date,
    "newSortCode": number,
    "startPropertyName" : string²,
    "endPropertyName" : string²,
    "cancel" : boolean, // [out]
    "promise" : Promise, // [out]
    "workingTimeDistance" : number, /*in
milliseconds*/
    "coupledObjects": Allocation[]|Activity[],
    "startsAndEndsOfCoupledObjects" : Object[],
    "otherNewSortCodesForSiblingObjects": 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.

If the option multipleBarDraggingEnabled is set to true and more than one object has been dragged, then the properties coupledObjects 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.

If a date symbol or bar of an activity is dropped, then the properties start/endPropertyName contain the name of the property to be modified when a dragging or dropping the symbol or bar, resp.

If the visualType is Row, then the property rowInsertionMode is set. If the sort mode for this type of row object is set to Ascending (see options activity/entity/resourceSortMode), then also the properties newSortCode and otherNewSortCodesForSiblingObjects is set. The latter is an array of objects of the following form: |

| Callback Name | Type | Description |
|---|---|---|
| | | <pre>{<br>    object : Activity\|Entity\|Resource,<br>    newSortCode : number<br>}.</pre>This array contains all sibling objects for which is new sort code is needed.<br><br>The code after a vertical row drag&drop could look like this: |

```
if (callbackArgs.rowInsertionMode !== RowInsertionMode.None) {
    // determine sort code property name
    let sortCodePropertyName = {
      [ObjectType.Activity]: "activityRowSortCodePropertyName",
      [ObjectType.Entity]: "entityRowSortCodePropertyName",
      [ObjectType.Resource]: "resourceRowSortCodePropertyName"
    }[callbackArgs.objectType];
    let sortCodePropertyName = vsw.option(sortCodePropertyName) || "PM_SortCode";

    let objectsToUpdate = [];

    // prepare update of parent ID and sort code of dropped object
    let objectToUpdate = Object.assign({}, callbackArgs.object); // clone object
    switch (callbackArgs.rowInsertionMode) {
      case RowInsertionMode.InsertAsChild:
        objectToUpdate.ParentID = callbackArgs.newRowObject.ID;
        break;

      case RowInsertionMode.InsertAsPreviousSibling:
      case RowInsertionMode.InsertAsNextSibling:
        objectToUpdate.ParentID = callbackArgs.newRowObject.ParentID;
        break;
    }
    objectToUpdate[sortCodePropertyName] = callbackArgs.newSortCode;

    // only for other object types than entities
    if (callbackArgs.objectType !== ObjectType.Entity)
      objectToUpdate.PM_ViewArea = callbackArgs.newRowObject.PM_ViewArea;

    // prepare update of sort code of other objects
    objectsToUpdate.push(objectToUpdate);
    if (callbackArgs.otherNewSortCodesForSiblingObjects) {
      callbackArgs.otherNewSortCodesForSiblingObjects.forEach(newSortCodeAndRowObject => {
        // clone object
        let otherObjectToUpdate = Object.assign({}, newSortCodeAndRowObject.object);

        otherObjectToUpdate[sortCodePropertyName] = newSortCodeAndRowObject.newSortCode;
        objectsToUpdate.push(otherObjectToUpdate);
      });
    }

    // update objects and render
    let updateObjectsMethodName = {
      [ObjectType.Activity]: "updateActivities",
      [ObjectType.Entity]: "updateEntities",
      [ObjectType.Resource]: "updateResources"
    }[callbackArgs.objectType];
    vsw[updateObjectsMethodName](objectsToUpdate);
    vsw.render();
}
```

Callback Name	Type	Description

onLogError	Function	**Optional, default: undefined** – If set, then this function is called when an exception occurs on any method described below that is called on the widget or on setting an option. By default, the exception is re-thrown afterwards.
		
		Profile:<pre>function (args)
		args = {
		  "commandName" : string,
		  "commandCounter": number,</pre>

Callback Name	Type	Description
		``` "error" : Error, "rethrow" : boolean /*out*/ } ``` The commandName property of the argument object contains one of the method or callback names. The commandCounter can be used to bundle errors of the same command. You can use this in your application to send the error from the client to the application server and make it persistent there. Normally no error exception should be triggered at all. See also callback onLogWarning. If the property "rethrow" is set to false, then the exception will not be re-thrown.
onLogWarning	Function	**Optional, default: undefined** – If set then this function is triggered when data is inconsistent among other incidents. Profile: ``` function (args) args = { "commandName" : string, "commandCounter": number, "description": string } ``` The commandName property of the argument object contains the pure current method name. The description contains an English text like "Option "xyz" is unknown", "Object ID empty", or "Object with ID "xyz" not unique". The commandCounter can be used to bundle errors of the same command. You can use this in your application to debug your application or to send the warning from the client to the application server and make it persistent there. Normally no warning should be triggered at all. See also callback onLogError.
onSaveAsPDFProgress	Function	**Optional, default: undefined** – This function is called constantly during the execution of the saveAsPDF method. Especially when saving a diagram to many pages, this callback is helpful for the application to be continuously informed about the progress of the processing. If a promise is returned by the application in the corresponding attribute, then VSW will wait for

Callback Name	Type	Description

		resolution before continuing the process. This serves to have the chance to show an updated progress dialog.
		
		Profile:
		`function (args)`
		`args = {`
		`  "pageCount" : number,`
		`  "currentPageNumber" : number,`
		`  "promise" : Promise /*out*/`
		`}`

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".
		
		The application can also decide whether to accept a selection change by validating the reason property eventually together with the causing object (e.g., clicking on an object or the background, showing a context menu, or starting a drag action).
		
		Profile:
		`function (args)`
		`args = {`
		`  "objectType" : ObjectType|0,`
		`  "object" : Object|null,`
		`  "selectedObjects" : Object[], //[in/out]`
		`  "visualType" : VisualType,`
		`  "previouslySelectedObjects" : Object[]|null,`
		`  "previouslySelectedObjectsType" : ObjectType`
		`                                 |null,`
		`  "reason": SelectionChangedReason,`
		`  "reasonObject": Object|undefined,`
		`  "reasonObjectType": ObjectType|undefined,`
		`  "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 Promise object at args.promise, 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,`

Callback Name	Type	Description

		`"object" : Object|null,`
		`"visualType" : VisualType,`
		`"date" : Date|undefined,`
		`"event" : DOMEvent or jQuery.Event,`
		`"entry"`[1]` : AllocationEntry|`
		`PeriodHighlighterEntry,`
		`"entryIndex"`[1]` : number,`
		`"hierarchySupplementaryDefinitionID"`[6]` : string,`
		`"hierarchyLevelSupplementaryDefinitionIndex"`[6]
		`: number,`
		`"groupingLevelDefinitionIndex"`[6]` : number,`
		`"groupingCodes"`[6]` : string[],`
		`"promise" : Promise //[out]`
		`}`
		
		In case of a grouping row object is null and instead groupingCodes together with the properties hierarchySupplementaryDefinitionID, hierarchyLevelSupplementaryDefinitionIndex, and groupingLevelDefinitionIndex are set.

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|null,`
		`  "visualType" : VisualType,`
		`  "visualSubtype" : VisualSubtype,`
		`  "event" : DOMEvent,`
		`  "date"`[7]` : Date // date at mouse cursor,`
		`  "capacity"`[8]` : number,`
		`  "load"`[8]` : number,`
		`  "singleLoads"`[8]` : Object,`
		`  "entry"`[1]` : AllocationEntry|`
		`          PeriodHighlighterEntry,`
		`  "entryIndex"`[1]` : number,`
		`  "periodHighlighter" : PeriodHighlighter,`
		`      // Only available when clicked on a`
		`      PeriodHighlighterEntry; the "object"`
		`      parameter will then hold the corresponding`
		`      resource/activity`
		`  "hierarchySupplementaryDefinitionID"`[6]` : string,`
		`  "hierarchyLevelSupplementaryDefinitionIndex"`[6]
		`      : number,`
		`  "groupingLevelDefinitionIndex"`[6]` : number,`
		`  "groupingCodes"`[6]` : string[],`

---

[7] Available only if objectType == ObjectType.Resource and the mouse cursor hovers over a curve area.

[8] 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.

Callback Name	Type	Description

		```"innerHTML"``` [9] ```: string //[in/out]```
		```"tooltipTemplateID" : string, // [in/out]```
		```"promise" : Object|null // [out]```
		```}```
		
		If you want to avoid showing a tooltip, you will have set the properties innerHTML and tooltipTemplateID to "" or null.
		
		In case of a grouping row object is null and instead groupingCodes together with the properties hierarchySupplementaryDefinitionID, hierarchyLevelSupplementaryDefinitionIndex, and groupingLevelDefinitionIndex are set.
		
		When the application sets a Promise object into args.promise, then the widget will re-new the text now provided as first parameter in the call to Promise.resolve.

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 resolution. There is an internal delay that is defined by option pm_scrollOffsetsChangedCallbackTimeDelay.
		
		Profile:
		```function (args)```
		```args = {```
		``` "scrollOffset" : number, /* in pixels */```

---

[9] Text to be displayed inside a tooltip window. This text has to be formatted compliant to the formatting rules for the contents of HTML <div> elements. **Line breaks** can be inserted by adding a <br> tag to the text. Embracing substrings by and tags will show **bold texts**. The same way you can use the <table> and the corresponding <tr> and <td> tags to **tabulate** the tooltip contents. If your original text contains the symbols "<" or ">" - i.e. those symbols should be displayed as they are and must not be interpreted as parts of HTML tag – then you have to replace the symbols by escape sequence codes (replace "<" by "<" and ">" by ">").

Callback Name	Type	Description
| | | ```
"width" : number, /* in pixels */
"start" : Date,
"end" : Date,
"timeResolutionUnit" : string (possible
    values: "seconds"/"minutes"/"hours"/
          "days"),
"timeResolutionUnitCount" : number,
"tableViewWidth" : number, /* current width,
  not to be confused with the option
  tableViewWidth */
"entitiesTableViewWidth" : number /* current
  width, not to be confused with the option
  entitiesTableViewWidth */
}
```<br><br>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 that is defined by option pm_scrollOffsetsChangedCallbackTimeDelay.<br><br>Profile:<br>```
function (args)
args = {
  "tableType" : TableType,
  "scrollOffset" : number, /* in pixels */
  "rowObjectTypeAtTop" : ObjectType,
  "rowObjectAtTop" : Object,
  "topViewScrollOffset" : number, /* in pixels */
  "topViewRowObjectAtTop" : Object
}
``` |
| **visibilityFilter**
DEPRECATED! | Function | **Deprecated for performance reasons, please see options visibilityFilterFor… below.** |
| **visibilityFilterForActivities** | Function | **Optional, default: undefined** – This function is called to hide objects. The result has to be set in the property named "result": true means visible and false means invisible. Setting the option with the same or another value again, triggers the visibility check for all activities immediately.<br><br>Profile:<br>```
function (args)
args = {
  "objectType" : ObjectType.Activity,
  "object" : Object,
  "result" : boolean /* [out], Default: true */
}
``` |
| **visibilityFilterForAllocations** | Function | **Optional, default: undefined** – This function is called to hide objects. The result has to be set in the property named "result": true means visible and false means invisible. Setting the option with the same or another value again, triggers |

Callback Name	Type	Description

		the visibility check for all allocations immediately.
		
		 Profile:
		```function (args)```
		```args = {```
		``` "objectType" : ObjectType.Allocation,```
		``` "object" : Object,```
		``` "result" : boolean /* [out], Default: true */```
		```}```

visibilityFilterForEntities	Function	**Optional, default: undefined** – This function is called to hide objects. The result has to be set in the property named "result": true means visible and false means invisible. Setting the option with the same or another value again, triggers the visibility check for all entities immediately.
		
		 Profile:
		```function (args)```
		```args = {```
		``` "objectType" : ObjectType.Entity,```
		``` "object" : Object,```
		``` "result" : boolean /* [out], Default: true */```
		```}```

visibilityFilterForResources	Function	**Optional, default: undefined** – This function is called to hide objects. The result has to be set in the property named "result": true means visible and false means invisible. Setting the option with the same or another value again, triggers the visibility check for all resources immediately.
		
		 Profile:
		```function (args)```
		```args = {```
		``` "objectType" : ObjectType.Activity,```
		``` "object" : Object,```
		``` "result" : boolean /* [out], Default: true */```
		```}```

4.3 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. [10]
addAllocations	allocations : Allocation[]	Adds allocations. [10]

Method Name : Result Type	Parameters	Description
addCalendars	calendars : Calendar[]	Adds calendars. [10]
addCurves	curves : Curve[]	Adds curves. [10]
addDateLines	dateLines : DateLine[]	Adds date lines. [10]
addEntities	entities : Entity[]	Adds entities. [10]
addHierarchy-Supplementary-Definitions	hierarchySupplementaryDefinitions : HierarchySupplementaryDefinition[]	Adds hierarchy supplementary definitions. [10]
addLinks	links : Link[]	Adds links. [10]
addPeriod-Highlighters	periodHighlighters : PeriodHighlighter[]	Adds period highlighters. [10]
addResources	resources : Resource[]	Adds resources. [10]
addSymbols	symbols : Symbol[]	Adds symbols. [10]
addTableRow-Definitions	tableRowDefinitions : TableRowDefinition[]	Adds table row definitions. [10]
addTooltip-Templates	tooltipTemplates : TooltipTemplate[]	Adds tooltip templates. [10]
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.
calculateWorking-Time : 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.
cancelSaveAsPDF	-	Cancels the execution of the saveAsPDF method.
fitTimeAreaInto-View : Promise	start : Date|undefined, end : Date|undefined	Fits the time area into the visible area. If start and/or end dates are given, then only the time between these are fitted into the visible area. Not given dates are internally replaced by start and end date of the complete time area.

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.

---

[10] 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
		See also selectObjects method.
highlightObjects	objectType : ObjectType, objectsOrIDs : string[]|object[], visualType : VisualType	Highlights the given objects or the objects addressed by the given IDs. In the activities view only activities and allocations can be highlighted. In the resources view only resources and allocations can be highlighted. In the loads view only resources can be highlighted.

The parameter visualType is only required for objects of type Activity or Allocation. In this case you can define whether the rows (VisualType.Row) or the bars (VisualType.Bar) should be highlighted.

Note: In the resources view, VisualType.Row can be applied only to allocations that reside in separate rows (allocation rows), but not to allocations inside resources rows.

The highlighting is shown by a flashing frame around the objects. The highlighting ends automatically when changing the data or with any user interaction or by using the method scrollToObject.

See also methods selectObjects and scrollToObject. See also options pm_objectHighlightingColor, pm_objectHighlightFlashingEnabled. |
removeActivities	activitiesOrIDs : string[]|Activity[]	Removes activities. [10]
removeAll	objectType : ObjectType|undefined	Removes all objects or just all objects of the given object type. [10]
removeAllocations	allocationsOrIDs : string[]|Allocation[]	Removes allocations. [10]
removeCalendars	calendarsOrIDs : string[]|Calendar[]	Removes calendars. [10]
removeCurves	curvesOrIDs : string[]|Curve[]	Removes curves. Resources have to be unused to be removable. [10]
removeDateLines	dateLinesOrIDs : string[]|DateLine[]	Removes date lines. [10]
removeEntities	entitiesOrIDs : string[]|Entity[]	Removes entities. [10]
removeHierarchy-Supplementary-Definitions	hierarchySupplementaryDefinitionsOrIDs : string[]|HierarchySupplementaryDefinition[]	Removes hierarchy supplementary definitions. [10]
removeLinks	linksOrIDs : string[]|Link[]	Removes links. [10]

Method Name : Result Type	Parameters	Description
removePeriod-Highlighters	periodHighlightersOrIDs : PeriodHighlighter[]|string[]	Removes period highlighters. [10]
removeResources	resourcesOrIDs : string[]|Resource[]	Removes resources. [10]
removeSymbols	symbolsOrIDs : string[]|Symbol[]	Removes symbols. [10]
removeTableRow-Definitions	tableRowDefinitionsOrIDs : string[]|TableRowDefinition[]	Removes table row definitions. [10]
removeTooltip-Templates	tooltipTemplatesOrIDs : string[]|TooltipTemplate[]	Removes tooltip templates. [10]
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.
saveAsPDF : Promise	fileName : string, options : Object	Saves the entire chart into a PDF document that is downloaded after creation. Possibly the browser asks whether to wait for completion or not. Additional libraries are needed: PDFKit, SVG-to-PDFKit, and blob-stream. For the properties bottomHTML and topHTML html2canvas is needed additionally. See chapter 2.2! The method returns a Promise object that the application can use, for instance, to react to the finish of the processing (e.g., to make a waiting screen disappear). The optional file name has to be pure (without any path information), and the file will be saved to the downloads folder of the browser by default. If no file name is specified, a new one is generated automatically. The optional options object can be used to specify additional properties for the export. The following attributes are allowed: • `"author"` : `string` (default: undefined) • `"bottomHTML"` : `string` (default: undefined) • `"bottomPageMargin"` : `number` (>= 0; default 10; in millimeters) • `"bottomText"` : `string` (default: undefined)

Method Name : Result Type	Parameters	Description
		• `"bottomTimescaleVisible"` : boolean (default: false) • `"cutMarksVisible"` : boolean (default: false) • `"horPageCountLimit"` : number (default: 0=not active, if "zoomFactorInPercent" is 0, then 1) * • `"keywords"` : string (default: undefined) • `"leftPageMargin"` : number (>= 0; default 10; in millimeters) • `"ownerPassword"` : string (default: undefined; if defined, you can edit the document in an appropriate application by entering this password) • `"pageFormat"` : string (default: "A4"; possible values "A0"/"A1"/"A2"/"A3"/"A4"/"A5"/"A6"/"Legal"/"Letter" or "w*h" with width and height in millimeters) • `"pageOrientation"` : number (default: Portrait; see enum [PageOrientation](#)) • `"permissionToAnnotate"` : boolean (default: true; if false, then it will not be possible to annotate text in the document) • `"permissionToAssembleDocument"` : boolean (default: true; if false, then it will not be possible to combine the document with others) • `"permissionToCopy"` : boolean (default: true; if false, then it will not be possible to copy text using the clipboard) • `"permissionToCopyForContentAccessibility"` : boolean (default: true; if false, then it will not be possible to copy content for accessibility) • `"permissionToModify"` : boolean (default: true; if false, then the PDF document can only be changed by the owner) • `"permissionToPrint"` : string (default: "highResolution"; possible values are "lowResolution", "highResolution", "none"; if not set to "lowResolution" or "highResolution", then it will not be possible to print the document) • `"printingMode"` : number (default: Cutting, see enum [PrintingMode](#)) • `"rightPageMargin"` : number (>= 0; default 10; in millimeters) • `"subject"` : string (default: undefined) • `"title"` : string (default: undefined) • `"topHTML"` : string (default: undefined) • `"topPageMargin"` : number (>= 0; default 10; in millimeters) • `"topText"`: string (default: undefined)

Method Name : Result Type	Parameters	Description
		• `"userPassword" : string` (default: undefined; if given, then it is possible to read the PDF document only by entering the password in an appropriate viewer application) • `"verPageCountLimit" : number` (default: 0=not active, if "zoomFactorInPercent" is 0, then 1) * • `"watermarkSymbolID" : string` (default: undefined) • `"zoomFactorInPercent" : number` (default: 0=not active, else > 0) * In printing mode **Single**, the widget content is placed in one single page (zoomFactorInPercent and hor/verPageCountLimit not respected). In printing mode **Paging**, table and timescale are repeated on each page. In printing mode **Cutting**, the pages are filled that way you can cut the pages and glue them. * If the zoomFactorInPercent is 0 and at least one of hor/verPageCountLimit are 0/undefined, then this limit value(s) will be set to 1. This way, only a minimum of parameters has to be set to get the expected output. Without setting any parameter, you will get a single page as output. The properties bottom/left/right/topPageMargin define the margins that are left blank on each page of the PDF document. This serves for a proper layout for printing it later. The properties bottomText/topText allow to specify additional texts for top/bottom frame area. The keywords {{#PageNo}}, {{#PageCount}}, {{#Date}} maybe used as placeholders. These texts are only usable if the corresponding properties topHTML/bottomHTML are not specified. The text is shown using the font family inherited from the div element of the widget and a font size of 10px. The properties bottomHTML/topHTML allow to specify additional HTML content for top/bottom frame area. The keywords {{#PageNo}}, {{#PageCount}}, {{#Date}} maybe used as placeholders. Also the keyword {{@*symbolID*}} is a placeholder for a defined symbol and can be used in <image src="…"> to show a symbol if needed (other URLs to external images are also possible). If topHTML or bottomHTML is used, then topText and bottomText are not usable, respectively. Text is shown using the font family and size inherited from

Method Name : Result Type	Parameters	Description
		the div element of the widget if the style is not modified within the HTML. Please ensure that the given HTML is valid. The property bottomTimescaleVisible determines showing an additional timescale at the bottom of the chart in the PDF document. The property cutMarksVisible determines showing marks at the four corners of each page in order to make it possible to cut the empty margins of printed pages and put the pages together. This only makes sense in printing mode Cutting. The properties horPageCountLimit and verPageCountLimit determine a zoom factor for the chart indirectly by setting the limits of page count. It is possible to set one of the limits only or to leave them both zero. The properties pageFormat and pageOrientation determine the size and orientation of each page in the PDF document. The property watermarkSymbolID allows to put a watermark on each page. The property zoomFactorInPercent determines the zoom factor for the chart, when not left zero. The properties author, keywords, subject, title, permissionTo… are put into the PDF document properties. See also method cancelSaveAsPDF and callback option onSaveAsPDFProgress. ATTENTION: Method is not usable with IE11!
scrollToDate : Promise	date: Date | string, offset : string|undefined	Scrolls to the given date. If the parameter offset is set, the view will be scrolled back by the given offset to get a distance between the left margin of the time area view and the given date. The offset can be a string with • a number that specifies a number of pixels (e.g. "50px"). • a percentage string that specifies the size of the offset as a percentage of the time area view width (e.g. "10%").
scrollToObject : Promise	objectType : ObjectType,	Scrolls to the object (activity/allocation/entity/ resource). If the object is not visible visible because

Method Name : Result Type	Parameters	Description
	objectOrID : Object|string, targetPositionInView : TargetPositions, highlightingEnabled : boolean	of being a hidden row or being within a hidden row, 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. Value Necessary means that the object will be made visible using the only necessary scrolling. 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_objectHighlightingColor, pm_objectHighlightFlashingEnabled, pm_scrollToObjectAnimationEnabled.
scrollViewArea-Horizontally	viewArea : HorizontallyScrollableViewArea, scrollPosition : HorizontalScrollPosition	Scrolls the specified view area horizontally to the left or right.
scrollViewArea-Vertically	viewArea : VerticallyScrollableViewArea, scrollPosition : VerticalScrollPosition	Scrolls the specified view area vertically to the top or bottom.
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 and links 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.
setTimeResolutionForView	unit : string ("seconds", "minutes", "hours", "days", ", "weeks", "months", "quarters",	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.

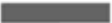| Method Name : Result Type	Parameters	Description
	"years")|TimeUnit, unitCount : number|undefined, start : Date|undefined	The time resolution cannot be set finer than the maximum time resolution defined by the options maximumTimeResolutionUnit and -Factor!
updateActivities	activities : Activity[], updateMode : UpdateModes	Update activities. Allowed changes are modification of all attributes besides ID. [10] 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. [10] 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. [10] 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. [10] 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. [10] 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. [10] updateMode is optional. See enum UpdateModes in the Enumerations chapter for details.
updateHierarchySupplementDefinitions	hierarchySupplementDefinitions : HierarchySupplementDefinition[], updateMode : UpdateModes	Updates hierarchy supplementary definitions. Allowed changes are modification of all attributes besides ID. [10] updateMode is optional. See enum UpdateModes in the Enumerations chapter for details.
updateLinks	links : Link[], updateMode : UpdateModes	Updates links. Allowed changes are modification of all attributes besides ID. [10] updateMode is optional. See enum UpdateModes in the Enumerations chapter for details.
updatePeriod-Highlighters	periodHighlighters : PeriodHighlighter[],	Updates period highlighters. [10]

Method Name : Result Type	Parameters	Description
	updateMode : UpdateModes	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. [10] 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. [10] updateMode is optional. See enum UpdateModes in the Enumerations chapter for details.
updateTableRow-Definitions	tableRowDefinitions: TableRowDefinition[], updateMode : UpdateModes	Updates table row definitions. Allowed changes are modification of all attributes besides ID. [10] updateMode is optional. See enum UpdateModes in the Enumerations chapter for details.
updateTooltip-Templates	tooltipTemplates: TooltipTemplate[], updateMode : UpdateModes	Updates tooltip templates. Allowed changes are modification of all attributes besides ID. [10] updateMode is optional. See enum UpdateModes in the Enumerations chapter for details.

5   Enumerations

The following enumerations are provided:

5.1   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,
};
```

5.2   ActivityBarShape

```
netronic.nVSW.ActivityBarShape = {
    Regular: 0,
    Summary: 1,
    Diamond: 2,
    Rectangle: 3,
    Symbol: 4
};
```

When the shapes Regular, Summary, or Rectangle are used, then the bar will span the time range defined by the allocation entries or defined by the assigned activity object when there is no allocation entry defined. When the shapes Diamond or Symbol are used the shape is set to the start date of the same time span.

When the shapes Diamond or Symbol are used, then the bar text is shown only when the bar is visible within an expanded row, otherwise the bar text is shown unconditionally.

5.3 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,
};
```

5.4 AllocationBarShape

```
netronic.nVSW.AllocationBarShape = {
    Regular: 0,
    Summary: 1,
    Diamond: 2,
    Rectangle: 3,
    Symbol: 4
};
```

The shapes Regular and Rectangle will show the allocation entries within the figure.

When the shapes Regular, Summary, or Rectangle are used, then the bar will span the time range defined by the allocation entries or defined by the assigned activity object when there is no allocation entry defined. When the shapes Diamond or Symbol are used the shape is set to the start date of the same time span.

When the shapes Diamond or Symbol are used, then the bar text is shown only when the bar is visible within an allocation row, otherwise the bar text is shown unconditionally.

5.5 BarSortMode

```
netronic.nVSW.BarSortMode = {
    StartAndEnd: 0,
    ByCompareObjects: 1,
    ByCompareObjectsOnSameStart: 2
};
```

5.6 CollapseState

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

```
};
```

5.7 CurveInterpolationType

```
netronic.nVSW.CurveInterpolationType = {
```

```
    StepAfter: 0,
```

```
    Linear: 1,
```

```
};
```

5.8 CurveType

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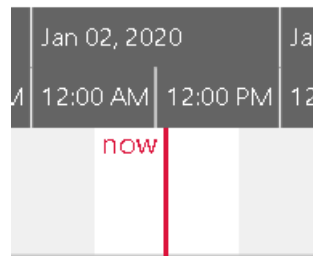
5.9 DateLineCaptionOrientation

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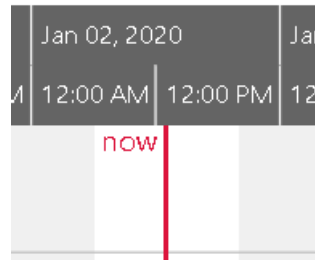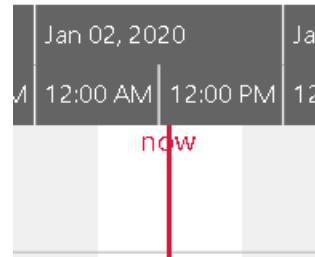
```
    Horizontal: 1,
```

```
    Vertical: 2
```

```
};
```

5.10 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

};

5.11 DateLineGridModes

```
netronic.nVSW.DateLineGridModes = {
    None: 0,
    Auto: 1,
    Weekly: 2,
    Daily: 4
};
```

5.12 HorizontalAlignment

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

5.13 HorizontallyScrollableViewArea

```
netronic.nVSW.HorizontallyScrollableViewArea = {
    Table: 0,
    TimeArea: 1,
    EntitiesTable: 2
}
```

5.14 HorizontalScrollPosition

```
netronic.nVSW.HorizontalScrollPosition = {
    Left: 1,
    Right: 2
}
```

5.15 LinkMarker

```
netronic.nVSW.LinkMarker = {
    None: 0,
    FilledArrow: 1
};
```

5.16 LinkRoutingType

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

5.17 ObjectType

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

```
    Entity: 13,
    PeriodHighlighter: 14,
    Symbol: 15,
    DateLine: 16,
    TooltipTemplate: 17,
    TableRowDefinition: 18,
    Calendar: 20
};
```

5.18 PageOrientation

```
netronic.nVSW.PageOrientation = {
    Portrait: 0,
    Landscape: 1
};
```

5.19 PanningMode

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

5.20 PatternType

```
netronic.nVSW.PatternType = {
    VerticalHatch: 0,
    ForwardHatch: 1,
    BackwardHatch: 2
};
```

5.21 PrintingMode

Note: See also here for an overview of the different modes.

```
netronic.nVSW.PrintingMode = {
    Single: 0,
    Cutting: 1,
    Paging: 2
};
```

5.22 ProgressBarWidthCalculationMode

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

5.23 RelationType

```
netronic.nVSW.RelationType = {
    FinishToStart: 0,
    FinishToFinish: 1,
    StartToStart: 2,
    StartToFinish: 3,
    SourceDateToStart: 4,
    SourceDateToFinish: 5,
    FinishToTargetDate: 8,
    StartToTargetDate: 10,
    SourceDateToTargetDate: 12
};
```

5.24 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 horizontal overlapping
    BarsInHiddenDescendantRows: 4,   // Shows bars of other hidden descendant rows
    CalendarGrid: 8                  // Shows calendar grid of row object
};
```

Further explanation:

- "Bars": In the activities view, the bar in an activity row represents the same object as the row itself. In the resources view, the bars represent the allocations that are assigned to the resource represented by the row. In the loads view, no bars are displayed as a matter of principle. For allocation rows in the activities view or resources view the bars represent the same object as the row itself. If this flag is set then the bars are shown, else they remain invisible.

- "BarsInHiddenDescendantRows": This flag is only effective when the row is shown in collapsed state but does no harm when the row is shown in expanded state. Bars of hidden rows below the collapsed row are projected into the row.

- "Optimized": When bars are shown in this row then they will be shown stacked vertically so that they do not overlap graphically when they allocate common time ranges. When bars are stacked the row is getting higher. If the flag is not set, the bars will graphically overlap and the row height is stable.

- "CalendarGrid": If this flag is set then the calendar assigned to the object represented by the row is made visible through a so-called calendar grid.

5.25 RowDragModes

```
netronic.nVSW.RowDragModes = {

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

    None: 0,
    DragVertically: 8,
```

```
    DragOutside: 32,
    DragOnSameLevelOnly: 64
};
```

5.26 RowInsertionMode

```
netronic.nVSW.RowInsertionMode = {
    None: 0,
    InsertAsChild: 1,
    InsertAsNextSibling: 2,
    InsertAsPreviousSibling: 3
};
```

5.27 RowSortMode

```
netronic.nVSW.RowSortMode = {
    None: 0,
    Ascending: 1,
    Descending: 2
};
```

5.28 SelectionChangedReason

```
netronic.nVSW.SelectionChangedReason = {
    Click: 0,
    BackgroundClick: 1,
    ContextMenu: 2,
    DragStart: 3
};
```

5.29 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
};
```

5.30 TableType

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

5.31 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
};
```

5.32 TextWrapMode

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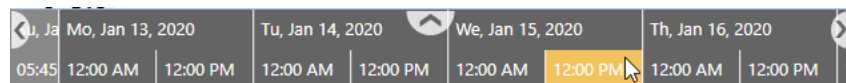
5.33 TimescaleInteractionModes

```
netronic.nVSW.TimescaleInteractionModes = {
    // Note: Values are flags,
    //       i.e. they can be combined by using bitwise OR operators.

    None: 0,                 // no interactions at all
    ScrollingByButtons: 1,   // horizontal scrolling by using scrolling buttons
    Rescaling: 2,            // rescaling by time period selection, up-button or
                            // mouse wheel
    Default: 3              // all interactions
};
```
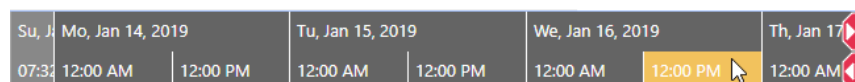
5.34 TimescaleNavigationMode

```
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.

```
};
```

5.35 TimeType

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

5.36 TimeUnit

```
netronic.nVSW.TimeUnit = {
    Seconds: 0,
    Minutes: 1,
    Hours: 2,
    Days: 3,
    Weeks: 4,
    Months: 5,
    Quarters: 6,
    Years: 7
};
```

5.37 TreeVisualizationMode

```
netronic.nVSW.TreeVisualizationMode = {
    ColoredIndentation: 0,
    TreeViewLines: 1
};
```

5.38 UpdateModes

```
netronic.nVSW.UpdateModes = {
    // Note: Values are flags,
    //       i.e. they can be combined by using bitwise OR operators.

    UpdateOnly: 0,          // deprecated, use Default
    Default: 0,
    ImplicitAddObjects: 1, // If an object to be updated does not exist,
                           // it will be added automatically.
    DifferentialValues: 2  // If set, then the object data given in the update
                           // method can contain only changed property values. If a
                           // property is omitted, it will be supplemented by the
                           // value of the property in the current object. If a
                           // property value shall be set to undefined explicitly,
                           // please use a null value instead or another value that
                           // can be set by the option
                           // resetValueForDifferentialUpdate.*
};
```

* When using this flag, the application will have to use simple objects in the update methods, because the VSW will modify them and supplement missing property values. Also, these objects have to be different to the ones given in former calls to the appropriate add or update method.

Please be aware that sub objects in arrays like entries must be fully defined despite of the active update mode.

5.39 VerticallyScrollableViewArea

```
netronic.nVSW.VerticallyScrollableViewArea = {
    Top: -1,
    Main: 0,
    EntitiesTable: 2
}
```

5.40 VerticalScrollPosition

```
netronic.nVSW.VerticalScrollPosition = {
    Top: 1,
    Bottom: 2
}
```

5.41 ViewArea

```
netronic.nVSW.ViewArea = {
    Top: -1,
    Main: 0,
    Default: 0 // for compatibility reasons
};
```

5.42 ViewType

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

5.43 VisualSubtype

```
netronic.nVSW.VisualSubtype = {
    ActivityDueDateSymbol: 1,
    ActivityReleaseDateSymbol: 2
};
```

5.44 VisualType

```
netronic.nVSW.VisualType = {
    Background: -1,
    Bar: 0,
    Row: 1,
    Curve: 2,
    Link: 3,
```

```
    PeriodHighlighter: 4,
    DateLine: 5
};
```

5.45 WorldViewPosition

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

APPENDIX

Information Material on Specific Topics

- **Hello Gantt World** - Build your first HTML5/JavaScript Gantt Chart within few minutes with the VSW (video)

- Update 2 of the Model for Resource Planning HTML5 Gantt Charts (blog post)

- What is an **activities** and a **resources view**? (blog post)

- **Calendars** – Individual resource working times in a HTML5 Gantt chart (blog post)

- **Links** as the visualization of **dependencies** (blog post

- The art of **designing bars** to map semantics to Gantt charts (video)

- From awful to awesome **progress visualization** in Gantt charts (blog post)

- **Backlogs** in HTML5 Gantt charts (blog post)

- **Hierarchy** vs. **grouping** (blog post)

Release Notes

Version 6.3
Version 6.2
Version 6.1
Version 6.0
Version 5.3
Version 5.2
Version 5.0
Version 4.0
Version 3.2
Version 3.1