

20-01 Resources view – Basics – Resources and allocations

This example shows how resources are added to the ResourceView and allocations appear as bars in the time area.

Each resource object requires a unique identifier, which is saved in the **ID** property. This allows resource objects to be distinguished from one another. The same applies to allocation objects. Each object type has its own namespace.

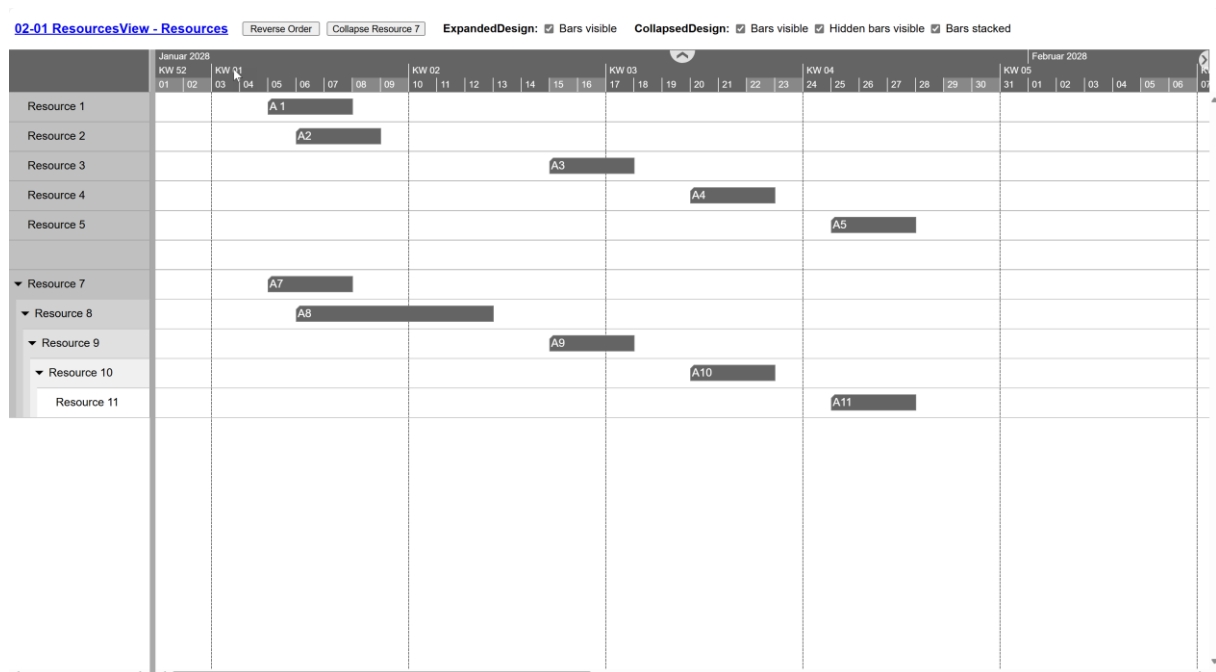
The start date (**Start**) and end date (**End**) in the allocation entries is required to display an allocation bar.

A text assignment for the table column can be made with **TableText** and for the labelling within the bar with **BarText**.

The **ParentID** property is required for hierarchical displays.

In the VSWidget, the principle applies that all objects are created and managed by the user of the library. If information with the specified designations is available in the object, it is used. All other entries are irrelevant for VSWidget. The VSWidget does not change any entries.

VSWidget is informed about the existing objects by the **AddResources()** and **AddAllocations()** methods. Exactly one row is created in the resource view for each resource object and one bar is created for each allocation object. The display sequence of resources corresponds to the specified sequence. Hierarchical subordinate resources are lined up directly behind ParentRows.



Bars within the rows are only ever visible if several conditions are met:

- The resource object must have assigned a valid allocation entry for **Start** and **End**.
- The time range for the bar must lie at least partially within the time range selected for the display.
- The **ExpandedRowDesign** and the **CollapseRowDesign** must be set to display bars.

There are special features for collapsed rows. Here you can set whether all bars of the non-visible rows should be displayed within the collapsed row:

BarsInHiddenDescendantRows



If overlaps occur, you can also set whether these bars should be displayed in their own sub-rows without overlaps:

BarsStacked



The **resourceBarSortModeForStackedRowDesign** option defines how the bars are arranged. The **BarSortMode** is set to **StartAndEnd** by default: The bars are sorted according to the start date. If the start date is the same, the longer bar is taken first.

In this example, the standard marking of bars has been prevented by setting the corresponding options:

```
defaultAllocationBarSelectable: false,
defaultResourceRowSelectable: false,
multipleSelectionEnabled: false,
```

The option to move bars has also been deactivated:

```
defaultAllocationAllowedBarDragModes: BarDragModes.None,
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <link href="../../VSWidget/nwaf-apptools.min.css" rel="stylesheet"/>
  <link href="../../VSWidget/nwaf-table.min.css" rel="stylesheet"/>
  <link href="../../VSWidget/nwaf-gantt.min.css" rel="stylesheet"/>
  <link href="../../VSWidget/nwaf-rab.min.css" rel="stylesheet"/>
  <script src="https://cdn.jsdelivr.net/npm/hammerjs@2.0.8/hammer.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/d3@7.9.0/dist/d3.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/tinycolor2@1.6.0/cjs/tinycolor.min.js"></script>
  <script src="../../VSWidget/nwaf-apptools.min.js"></script>
  <script src="../../VSWidget/nwaf-table.min.js"></script>
  <script src="../../VSWidget/nwaf-gantt.min.js"></script>
  <script src="../../VSWidget/nwaf-rab.min.js"></script>
  <script src="../../LicenseKey.js"></script>
  <script src="Miscellaneous.js"></script>
</head>
<body>
  <div id="toolbar">
    <input id="reverseOrder" type="button" value="Reverse Order"
      style="margin-left: 15px; height: 20px; width: 110px"/>
    <input id="collapse" type="button" value="Collapse Resource 7"
      style="margin-left: 5px; height: 20px; width: 140px"/>
    <label style="margin-left: 20px">
      <b>ExpandedDesign:</b>
    </label>
    <input type="checkbox" id="expandBarVisibility" checked/>
    <label for="expandBarVisibility">Bars visible</label>
    <label style="margin-left: 20px">
      <b>CollapsedDesign:</b>
    </label>
    <input type="checkbox" id="collapseBarVisibility" checked/>
    <label for="collapseBarVisibility">Bars visible</label>
    <input type="checkbox" id="collapseHiddenBarVisibility" checked/>
    <label for="collapseHiddenBarVisibility">Hidden bars visible</label>
    <input type="checkbox" id="collapseStackedBarVisibility" checked/>
    <label for="collapseStackedBarVisibility">Bars stacked</label>
```

```

</div>
<div id="VSWidget"></div>
<script>
    Miscellaneous.ready(() => {
        const { BarDragModes, BarSortMode, CollapseState, ViewType, RowDesigns } = netronic.nVSW;

        document.getElementById("reverseOrder").addEventListener("click", reverseOrder);
        document.getElementById("collapse").addEventListener("click", collapseOrExpandResource7);
        document.getElementById("expandBarVisibility").addEventListener("click", updateRowDesigns);
        document.getElementById("collapseBarVisibility").addEventListener("click", updateRowDesigns);
        document.getElementById("collapseHiddenBarVisibility").addEventListener("click", updateRowDesigns);
        document.getElementById("collapseStackedBarVisibility").addEventListener("click", updateRowDesigns);

        const timeAreaStart = new Date(2028, 0, 1);
        const timeAreaEnd = new Date(2028, 3, 1);

        const options = {
            licenseKey: window.VSW_LICENSE_KEY,

            viewType: ViewType.Resources,

            start: timeAreaStart,
            end: timeAreaEnd,

            defaultResourceExpandedRowDesign: RowDesigns.Bars,
            defaultResourceCollapsedRowDesign: RowDesigns.Bars |
                RowDesigns.BarsInHiddenDescendantRows |
                RowDesigns.BarsStacked,

            allocationBarSortModeForStackedRowDesign: BarSortMode.StartAndEnd,

            defaultAllocationAllowedBarDragModes: BarDragModes.None,
            defaultAllocationBarSelectable: false,
            defaultResourceRowSelectable: false,
            multipleSelectionEnabled: false,

            decouplingOfAllocationPropertiesFromActivities: true,
            fixedTableColumnWidth: 0,

            onCollapseStateChanged: (args) => {
                if (args.object.ID === "Res7") {
                    const btn1 = document.getElementById("collapse");

                    btn1.value = args.newCollapseState === CollapseState.Expanded ? "Collapse Resource 7" :
                        "Expand Resource 7";

                    args.object.CollapseState = args.newCollapseState;
                }
            },
        };

        const vsWidget = Miscellaneous.instantiateVSWidget(document.querySelector("#VSWidget"), options);
        const resources = createResources();
        const allocations = createAllocations();
        vsWidget.addResources(resources);
        vsWidget.addAllocations(allocations);
        vsWidget.render();

        function createResources() {
            const resources = [];

            for (let i = 1; i <= 11; i++) {
                const resource = { ID: `Res${i}` };

                if (i !== 6)
                    resource.TableText = `Resource ${i}`;
                if (i > 7)
                    resource.ParentID = `Res${i - 1}`;

                resources.push(resource);
            }
            return resources;
        }
    });

```

```

function createAllocations() {
  const allocations = [];
  const startDays = [5, 6, 15, 20, 25, 0, 5, 6, 15, 20, 25]; //Start dates January
  const workingDays = [3, 3, 3, 3, 3, 0, 3, 7, 3, 3, 3];

  for (let i = 0; i < startDays.length; i++) {
    const allocation = {
      ID: `Alloc${i + 1}`,
      ResourceID: `Res${i + 1}`,
      BarText: `A${i + 1}`,
      my_WorkingDays: workingDays[i],
      Entries: [
        {
          Start: new Date(2028, 0, startDays[i]),
          End: new Date(2028, 0, startDays[i] + workingDays[i]),
        },
      ],
    };
    allocations.push(allocation);
  }
  return allocations;
}

function reverseOrder() {
  vsWidget.removeResources(resources);
  vsWidget.removeAllocations(allocations);
  vsWidget.addResources(resources.reverse());
  vsWidget.addAllocations(allocations);
  vsWidget.render();
}

function collapseOrExpandResource7() {
  const resource = resources.find((resource) => resource.ID === "Res7");

  resource.CollapseState = resource.CollapseState === CollapseState.Collapsed ?
    CollapseState.Expanded :
    CollapseState.Collapsed;

  vsWidget.updateResources([resource]);
  vsWidget.render();
}

function updateRowDesigns() {
  let expandedRowDesign = RowDesigns.None;
  let collapsedRowDesign = RowDesigns.None;

  if (document.getElementById("expandBarVisibility").checked)
    expandedRowDesign = RowDesigns.Bars;

  if (document.getElementById("collapseBarVisibility").checked)
    collapsedRowDesign = collapsedRowDesign | RowDesigns.Bars;

  if (document.getElementById("collapseHiddenBarVisibility").checked)
    collapsedRowDesign =
      collapsedRowDesign | RowDesigns.BarsInHiddenDescendantRows;

  if (document.getElementById("collapseStackedBarVisibility").checked)
    collapsedRowDesign = collapsedRowDesign | RowDesigns.BarsStacked;

  vsWidget.option({
    defaultResourceExpandedRowDesign: expandedRowDesign,
    defaultResourceCollapsedRowDesign: collapsedRowDesign,
  });
}
});
</script>
</body>
</html>

```