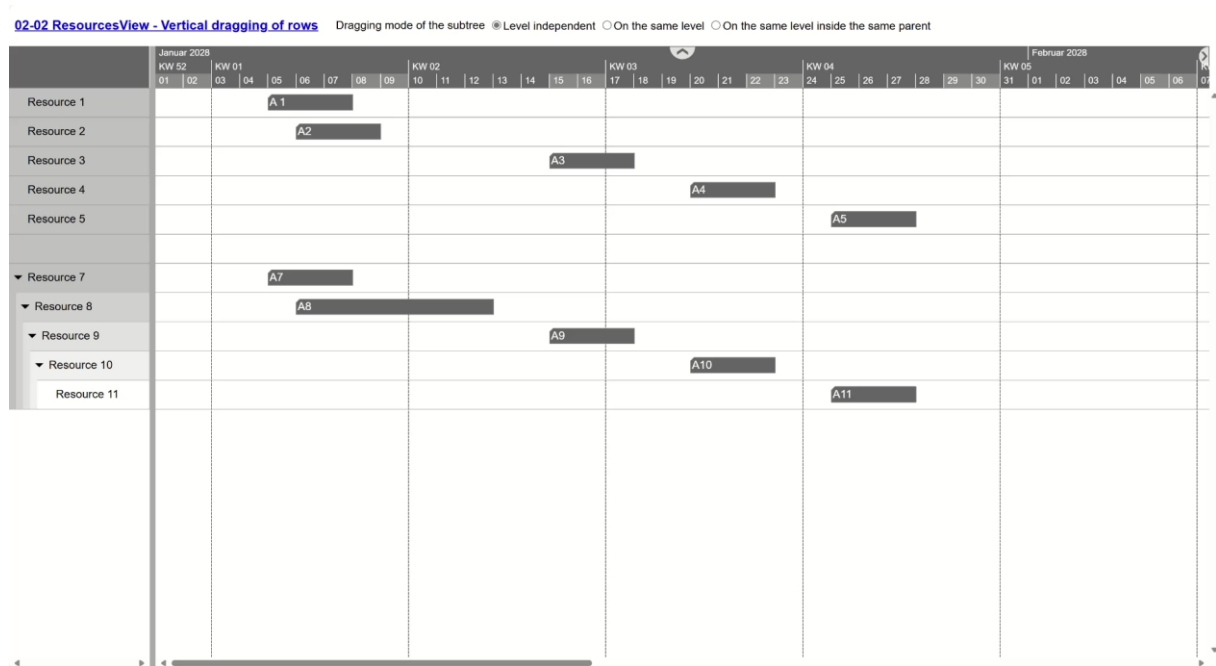


22-01 Resources view – Interactions – Moving rows via drag&drop

This example shows how moving a row changes the position of the associated subtree.

To allow rows to be dragged, the **defaultResourceAllowedRowDragModes** option must be set for the **RowDragModes.DragVertically** enum value.

In addition, the **resourceRowSortMode** option must be set to the enum value **RowSortMode.Ascending** so that the row sequence can be managed.



The way in which the movement affects the subtree can be selected.

Dragging mode of the subtree ☒ Level independent ☐ On the same level ☐ On the same level inside the same parent

- **Level independent**
The **defaultResourceAllowedRowDragModes** option must be set to the enum value **RowDragModes.DragVertically**. A subtree can be inserted into an existing hierarchy at any level and at any position. However, it is not possible to insert it into your own subtree. If a subtree is to be assigned as a child of an activity that does not yet have any children, the **Shift** key must be held down when dragging.
- **On the same level**
The **defaultResourceAllowedRowDragModes** option must be set to the enum value **RowDragModes.DragVertically | Enum.RowDragModes.DragOnSameLevelOnly**. A subtree can be inserted into the same level at any position.
- **On the same level inside the same parent**
The **defaultResourceAllowedRowDragModes** option must be set to the enum value **RowDragModes.DragVertically | Enum.RowDragModes.DragInSameTableParentOnly**. A subtree can be inserted into the same level inside the same parent at any position.

The **processOnDrop()** method, which is called within the **onDrop** callback function, updates the resource objects. The method simplifies application development if no additional changes need to be made to the resource objects by the application.

```
<!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">
    <h3>
      <a href="docs/22-01_ResourcesView-Interactions-Moving_rows_via_drag&drop.pdf"
        target="_blank">22-01 Resources view - Interactions - Moving rows via drag&drop</a>
    </h3>
    <label style="margin-left: 20px">Dragging mode of the subtree</label>
    <label>
      <input id="levelIndependent" type="radio" name="moveMode" value="levelIndependent" checked/>
      Level independent
    </label>
    <label>
      <input id="onSameLevel" type="radio" name="moveMode" value="onSameLevel"/>
      On the same level
    </label>
    <label>
      <input id="inSameTableParent" type="radio" name="moveMode" value="inSameTableParent"/>
      On the same level inside the same parent
    </label>
  </div>
  <div id="VSWidget"></div>
  <script>
    Miscellaneous.ready(() => {
      const { BarDragModes, RowDragModes, RowSortMode,
        RowDesigns, ViewType } = netronic.nVSW;

      document.getElementById("levelIndependent").addEventListener("click", changeRowDragDropOptions);
      document.getElementById("onSameLevel").addEventListener("click", changeRowDragDropOptions);
      document.getElementById("inSameTableParent").addEventListener("click", changeRowDragDropOptions);

      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,

        defaultCalendarID: "Cal1",
        defaultResourceExpandedRowDesign: RowDesigns.Bars,
        defaultResourceCollapsedRowDesign: RowDesigns.Bars |
          RowDesigns.BarsInHiddenDescendantRows |
          RowDesigns.BarsStacked,

        ignoreCalendarOnAllocationBarInteractions: true,
        defaultResourceAllowedRowDragModes: RowDragModes.DragVertically,

        defaultAllocationAllowedBarDragModes: BarDragModes.None,
        resourceRowSortMode: RowSortMode.Ascending,

        defaultAllocationBarSelectable: false,
        defaultResourceRowSelectable: false,
        multipleSelectionEnabled: false,
        fixedTableColumnWidth: 0,
      };
    });
  </script>
</body>
</html>
```

```

        onDrop: (args) => vsWidget.processOnDrop(args),
    };

    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];
        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 changeRowDragDropOptions(event) {
        const optionBox = event.target;

        if (optionBox.value === "levelIndependent" && optionBox.checked)
            vsWidget.option("defaultResourceAllowedRowDragModes", RowDragModes.DragVertically);
        else if (optionBox.value === "onSameLevel" && optionBox.checked)
            vsWidget.option("defaultResourceAllowedRowDragModes", RowDragModes.DragVertically |
                RowDragModes.DragOnSameLevelOnly);
        else if (optionBox.value === "inSameTableParent" && optionBox.checked)
            vsWidget.option("defaultResourceAllowedRowDragModes", RowDragModes.DragVertically |
                RowDragModes.DragInSameTableParentOnly);
    }
});
</script>
</body>
</html>

```