

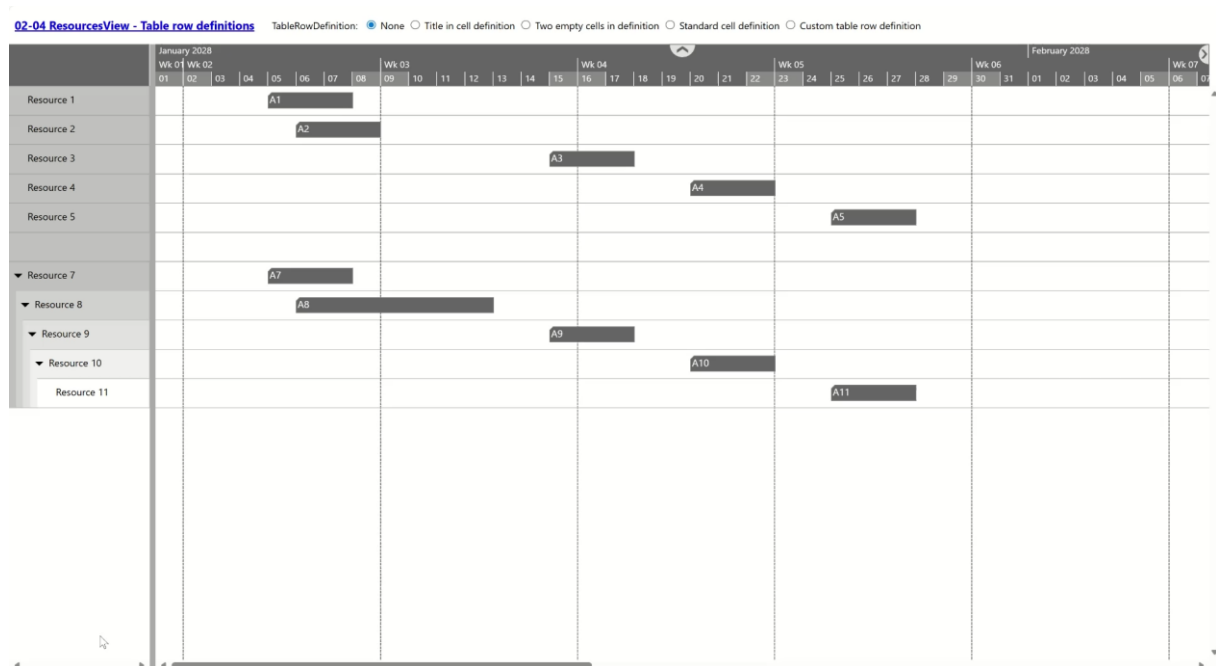
20-02 Resources view – Basics – Row definitions.html

This example shows how `TableRowDefinitions` are created and used.

The **TableRowDefinition** determines how a table row is divided into data fields and where the displayed data comes from.

Setting the **defaultResourceTableRowDefinitionID** option determines which table row definition is to be used.

Although the option is not set at the beginning, an empty table row definition is used internally, which consists of a data field and displays the content of the **TableText** data field.



However, this definition cannot be addressed or changed as there is no assigned ID. In terms of content, this is the equivalent:

```
const rowDef = {
  ID: "CustomRowDef",
  CellDefinitions: [{}]
};

vsWidget.addTableRowDefinitions([rowDef]);
vsWidget.option("defaultResourceTableRowDefinitionID", " CustomRowDef");
vsWidget.render();
```

It is now possible to add a header text to the table:

```
const rowDef = {
  ID: "CustomRowDef",
  CellDefinitions: [{TitleText: "Resource"}]
};
```

Resource
Resource 1
Resource 2
Resource 3
Resource 4
Resource 5

The complete definition behind **CellDefinitions: [{ }]** looks like this:

```
const rowDef = {
  ID: "CustomRowDef",
  CellDefinitions: [
    {
      TitleText: "",
      HorizontalTitleAlignment: HorizontalAlignment.Center,
      TextSource: "TableText",
      Width: 200,
      MinimumWidth: 3,
      VerticalAlignment: VerticalAlignment.FirstLineOnBaseline,
      HorizontalAlignment: HorizontalAlignment.Left,
      WrapMode: TextWrapMode.NoWrap,
    }
  ]
};
```

It becomes interesting when you arrange two empty cells one behind the other:

CellDefinitions: [{ }, { }]

Resource 1	Resource 1
Resource 2	Resource 2
Resource 3	Resource 3
Resource 4	Resource 4
Resource 5	Resource 5
▼ Resource 7	Resource 7
▼ Resource 8	Resource 8
▼ Resource 9	Resource 9
▼ Resource 10	Resource 10
Resource 11	Resource 11

In this case, you can see that the first cell has a special role because the drop-down symbols for the hierarchy are added there.

Finally, the **TextFormat** property must be used instead of **TextSource**:

```
{
  TitleText: "ParentID",
  HorizontalTitleAlignment: HorizontalAlignment.Center,
  TextFormat: "{{ParentID}}",
  HorizontalAlignment: HorizontalAlignment.Center,
  Width: 100,
  MinimumWidth: 100,
```

```

    MaximumWidth: 100,
},

```

If **MinimumWidth** and **MaximumWidth** are set to the value of **Width**, you get a column whose size cannot be changed interactively.

Resource	ParentID
Resource 1	
Resource 2	
Resource 3	
Resource 4	
Resource 5	
▼ Resource 7	
▼ Resource 8	Res7
▼ Resource 9	Res8
▼ Resource 10	Res9
Resource 11	Res10

Good to know:

- Column widths of the cells can be changed interactively.
- It is not possible to move columns interactively.
- Data fields are purely display fields that do not allow any input.
- Each row can have its own **TableRowDefinition**.

```

<!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">
    <form id="myForm">
      <label style="margin-left: 20px">TableRowDefinition:&nbsp;</label>
      <label>
        <input id="rowDef1" type="radio" name="rowDef" checked/>
        None
      </label>
    </form>
  </div>

```

```

        <input id="rowDef2" type="radio" name="rowDef" value="CustomRowDef1"/>
        Title in cell definition
    </label>
    <label>
        <input id="rowDef3" type="radio" name="rowDef" value="CustomRowDef2"/>
        Two empty cells in definition
    </label>
    <label>
        <input id="rowDef4" type="radio" name="rowDef" value="CustomRowDef3"/>
        Standard cell definition
    </label>
    <label>
        <input id="rowDef5" type="radio" name="rowDef" value="CustomRowDef4"/>
        Custom table row definition
    </label>
</form>
</div>
<div id="VSWidget"></div>
<script>
    Miscellaneous.ready(() => {
        const { BarDragModes, HorizontalAlignment, VerticalAlignment,
            RowDesigns, TextWrapMode, ViewType } = netronic.nVSW;

        document.getElementById("rowDef1").addEventListener("click", changeTableRowDefinition);
        document.getElementById("rowDef2").addEventListener("click", changeTableRowDefinition);
        document.getElementById("rowDef3").addEventListener("click", changeTableRowDefinition);
        document.getElementById("rowDef4").addEventListener("click", changeTableRowDefinition);
        document.getElementById("rowDef5").addEventListener("click", changeTableRowDefinition);

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

        let options = {
            licenseKey: window.VSW_LICENSE_KEY,

            viewType: ViewType.Resources,
            start: timeAreaStart,
            end: timeAreaEnd,

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

            defaultAllocationAllowedBarDragModes: BarDragModes.None,
            defaultAllocationBarSelectable: false,
            defaultResourceRowSelectable: false,
            multipleSelectionEnabled: false,
            fixedTableColumnWidth: 0,
        };

        const vsWidget = Miscellaneous.instantiateVSWidget(document.querySelector("#VSWidget"), options);

        const tableRowDefinitions = createTableRowDefinitions();
        const resources = createResources();
        const allocations = createAllocations();
        vsWidget.addTableRowDefinitions(tableRowDefinitions);
        vsWidget.addResources(resources);
        vsWidget.addAllocations(allocations);
        vsWidget.render();

        function createTableRowDefinitions() {
            const tableRowDefinitions = [
                {
                    ID: "CustomRowDef1",
                    CellDefinitions: [
                        {
                            TitleText: "Resource",
                        },
                    ],
                },
                {
                    ID: "CustomRowDef2",
                    CellDefinitions: [{}, {}],
                },
            ];
        }
    });

```

```

    },
    {
      ID: "CustomRowDef3",
      CellDefinitions: [
        {
          TitleText: "",
          HorizontalTitleAlignment: HorizontalAlignment.Center,
          TextSource: "TableText",
          Width: 200,
          MinimumWidth: 3,
          VerticalAlignment: VerticalAlignment.FirstLineOnBaseline,
          HorizontalAlignment: HorizontalAlignment.Left,
          WrapMode: TextWrapMode.Nowrap,
        },
      ],
    },
  ],
  {
    ID: "CustomRowDef4",
    CellDefinitions: [
      {
        TitleText: "Resource",
        HorizontalTitleAlignment: HorizontalAlignment.Center,
        TextSource: "TableText",
        Width: 170,
      },
      {
        TitleText: "ParentID",
        HorizontalTitleAlignment: HorizontalAlignment.Center,
        TextFormat: "{{ParentID}}",
        HorizontalAlignment: HorizontalAlignment.Center,
        Width: 100,
        MinimumWidth: 100,
        MaximumWidth: 100,
      },
    ],
  },
];

return tableRowDefinitions;
}

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]; // Tageszahlen im Januar 2028
  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 changeTableRowDefinition(event) {
    const radio = event.target;

    vsWidget.option({
        defaultResourceTableRowDefinitionID: radio.value,
        tableViewWidth: 400,
    });
}
});
</script>
</body>
</html>
```