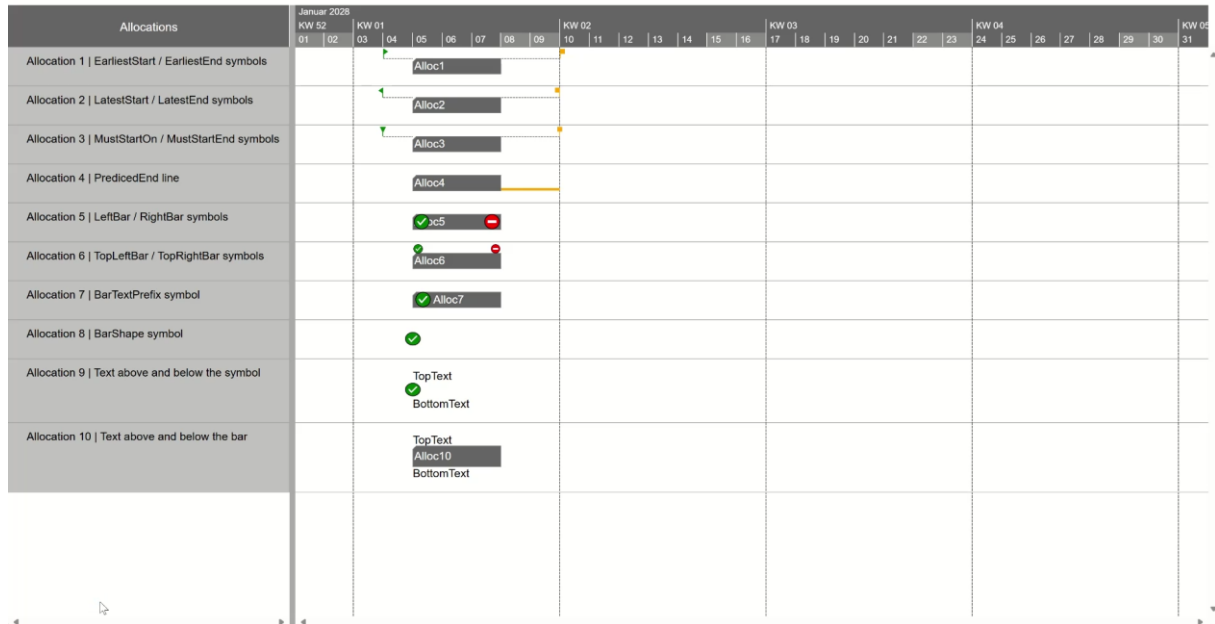


21-02 Resources view – Visualization - Symbols

This example shows which different symbols can be attached to the bars.

02-14 ResourcesView - Allocation symbols



To ensure that the symbols for **EarliestStartDate**, **LatestStartDate** and **MustStartOnDate** are completely within the row, the **topRowMarginInTimeArea** option must be increased from 10 to 17 pixels.

`topRowMarginInTimeArea: 17,`

To maintain the centred position of the bar, it is recommended to set the **bottomRowMarginInTimeArea** option to the same value.

`bottomRowMarginInTimeArea: 17,`

	Refers to	DateSource	SymbolID	Color	Height	Width	X-Offset
1.	EarliestStartDate	EarliestStart					right
2.	EarliestEndDate	EarliestEnd					right
3.	LatestStartDate	LatestStart					left
4.	LatestEndDate	LatestEnd					left
5.	MustStartOnDate	MustStartOn					center
6.	MustEndOnDate	MustStartOn					center
7.	PredictedEndDate	PredictedEnd	No symbol.				center
8.	LeftBarSymbol	Start					right + Δ
9.	RightBarSymbol	End					left - Δ
10.	TopLeftBarSymbol	Start					right
11.	TopRightBarSymbol	End					left
12.	BarTextPrefixSymbol	Start					right + Δ

13.	BarShapeSymbol	Start					center
-----	----------------	-------	--	--	--	--	--------

*Colours can only be selected for the built-in Diamond symbol, but not for self-defined symbols.

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>02-14 NETRONIC VSW SE</title>
  <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/21-02_ResourcesView-Visualization-Symbols.pdf"
        target="_blank">21-02 Resources view - Visualization - Symbols</a>
    </h3>
  </div>
  <div id="VSWidget"></div>

  <script>
    Miscellaneous.ready(() => {
      const { BarShape, BarDragModes, RowDesigns, ViewType } = netronic.nVSW;

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

      const options = {
        licenseKey: window.VSW_LICENSE_KEY,

        viewType: ViewType.Resources,

        start: timeAreaStart,
        end: timeAreaEnd,

        multipleSelectionEnabled: false,
        topRowMarginInTimeArea: 17,
        bottomRowMarginInTimeArea: 17,
        fixedTableColumnWidth: 0,

        defaultResourceExpandedRowDesign: RowDesigns.Bars,
        defaultResourceTableRowDefinitionID: "CustomRowDef",
        defaultResourceRowSelectable: false,
        defaultAllocationBarSelectable: false,
        defaultAllocationAllowedBarDragModes: BarDragModes.None,
      };

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

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

      function createTableRowDefinitions() {

```

```

const tableRowDefinitions = [
  {
    ID: "CustomRowDef",
    CellDefinitions: [
      {
        TitleText: "Allocations",
        Width: 400,
      },
    ],
  },
];
return tableRowDefinitions;
}

function createSymbols() {
  const symbols = [
    { ID: "Symbol1", URL: "images/warning.svg" },
    { ID: "Symbol2", URL: "images/checkmark.svg" },
    { ID: "Symbol3", URL: "images/document.svg" },
    { ID: "Symbol4", URL: "images/closed.svg" },
    { ID: "Symbol5", URL: "images/opened.svg" },
    { ID: "Symbol6", URL: "images/one-way.svg" },
  ];
  return symbols;
}

function createResources() {
  const total = 10;
  const resources = [];

  for (let i = 1; i <= total; i++) {
    const resource = {
      ID: `Res${i}`,
    };
    resources.push(resource);
  }

  resources[0].TableText = "Allocation 1 | EarliestStart / EarliestEnd symbols";
  resources[1].TableText = "Allocation 2 | LatestStart / LatestEnd symbols";
  resources[2].TableText = "Allocation 3 | MustStartOn / MustStartEnd symbols";
  resources[3].TableText = "Allocation 4 | PredictedEnd line";
  resources[4].TableText = "Allocation 5 | LeftBar / RightBar symbols";
  resources[5].TableText = "Allocation 6 | TopLeftBar / TopRightBar symbols";
  resources[6].TableText = "Allocation 7 | BarTextPrefix symbol";
  resources[7].TableText = "Allocation 8 | BarShape symbol";
  resources[8].TableText = "Allocation 9 | Text above and below the symbol";
  resources[9].TableText = "Allocation 10 | Text above and below the bar";

  return resources;
}

function createAllocations() {
  const allocations = [];

  for (let i = 1; i <= 10; i++) {
    let allocation = {
      ID: `Alloc${i}`,
      ResourceID: `Res${i}`,
      TableText: `Allocation ${i}`,
      BarText: `Alloc${i}`,
    };

    if (i == 1) {
      allocation.EarliestStart = new Date(2028, 0, 4);
      allocation.EarliestStartColor = "green";
      allocation.EarliestEnd = new Date(2028, 0, 10);
      allocation.EarliestEndColor = "orange";
    }

    if (i == 2) {
      allocation.LatestStart = new Date(2028, 0, 4);
      allocation.LatestStartColor = "green";
      allocation.LatestEnd = new Date(2028, 0, 10);
      allocation.LatestEndColor = "orange";
    }
  }
}

```

```

    if (i == 3) {
        allocation.MustStartOn = new Date(2028, 0, 4);
        allocation.MustStartOnColor = "green";
        allocation.MustEndOn = new Date(2028, 0, 10);
        allocation.MustEndOnColor = "orange";
    }
    if (i == 4) {
        allocation.PredictedEnd = new Date(2028, 0, 10);
        allocation.PredictedEndColor = "orange";
    }
    if (i == 5) {
        allocation.LeftBarSymbolID = "Symbol2";
        allocation.LeftBarSymbolHeight = 25;
        allocation.LeftBarSymbolWidth = 25;
        allocation.RightBarSymbolID = "Symbol6";
        allocation.RightBarSymbolHeight = 25;
        allocation.RightBarSymbolWidth = 25;
    }
    if (i == 6) {
        allocation.TopLeftBarSymbolID = "Symbol2";
        allocation.TopLeftBarSymbolHeight = 15;
        allocation.TopLeftBarSymbolWidth = 15;
        allocation.TopLeftBarSymbolYOffset = -4;
        allocation.TopRightBarSymbolID = "Symbol6";
        allocation.TopRightBarSymbolHeight = 15;
        allocation.TopRightBarSymbolWidth = 15;
        allocation.TopRightBarSymbolYOffset = -4;
    }
    if (i == 7) {
        allocation.BarTextPrefixSymbolID = "Symbol2";
        allocation.BarTextPrefixSymbolHeight = 25;
        allocation.BarTextPrefixSymbolWidth = 25;
    }

    if (i == 8) {
        allocation.BarShape = BarShape.Symbol;
        allocation.BarShapeSymbolID = "Symbol2";
        allocation.BarShapeSymbolWidth = 25;
        allocation.BarShapeSymbolHeight = 25;
    }
    if (i == 9) {
        allocation.BarTopOutsideText = "TopText";
        allocation.BarBottomOutsideText = "BottomText";
        allocation.BarShape = BarShape.Symbol;
        allocation.BarShapeSymbolID = "Symbol2";
        allocation.BarShapeSymbolWidth = 25;
        allocation.BarShapeSymbolHeight = 25;
    }
    if (i == 10) {
        allocation.BarTopOutsideText = "TopText";
        allocation.BarBottomOutsideText = "BottomText";
        allocation.BarHeight = 30;
    }

    // Entries are set for the 10 allocations
    if (i <= 11) {
        allocation.Entries = [
            {
                Start: new Date(2028, 0, 5),
                End: new Date(2028, 0, 8),
            },
        ];
    }
    allocations.push(allocation);
}

return allocations;
}
});
</script>
</body>
</html>

```