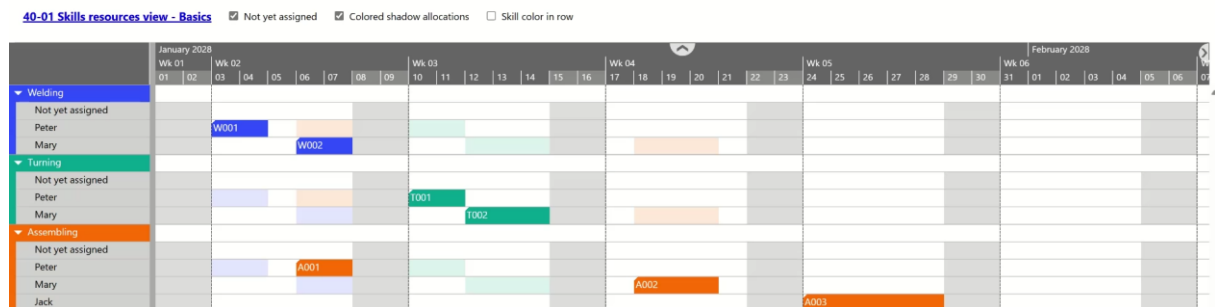


40-01 Skilled resources view - Basics

The Skilled Resources View is designed for displaying resources with multiple skills. All skills appear as main rows, with the associated resources listed below them. A resource can therefore appear multiple times — once under each skill it possesses.



- Peter, Mary have three skills: Welding, Turning, Assembling
- Jack has only one skill: Assembling

The skilled resources view is enabled through the option

`viewType: ViewType.SkilledResources,`

Skill Objects

Skills are managed as separate objects of the type **Skill**.

```
const skillColors = ["#304ffe", "#009e8e", "#ff6d00"];
const skills = [
  { ID: "Skill_Welding", TableText: "Welding", TableColor: skillColors[0], TableTextColor: "white"},
  { ID: "Skill_Turning", TableText: "Turning", TableColor: skillColors[1], TableTextColor: "white"},
  { ID: "Skill_Assembling", TableText: "Assembling", TableColor: skillColors[2], TableTextColor: "white"}
];
vsw.addSkills(skills);
```

All skills are displayed at the top level; a hierarchical subdivision of skills is not supported.

Resource Objects

In this context, the **SkillIDs** attribute is the most important property of the resource. It defines which skills the resource possesses. The resource is listed in the diagram exactly once for each of the specified skills.

```
const resources = [
  { ID: "Res_Peter", TableText: "Peter", _Color: "#FFD27F", SkillIDs: ["Skill_Welding", "Skill_Turning", "Skill_Assembling"] },
  { ID: "Res_Mary", TableText: "Mary", _Color: "#7EF5E0", SkillIDs: ["Skill_Welding", "Skill_Turning", "Skill_Assembling"] },
  { ID: "Res_Jack", TableText: "Jack", _Color: "#7EF5E0", SkillIDs: ["Skill_Assembling"] }
];
```

A hierarchical subdivision of resources is not possible in this view; any specified **ParentID** attribute will be ignored. Resources **without** skills and resources with **unknown** SkillIDs are not displayed.

An additional resource for each skill can be defined to cover the use case where it has not yet been determined which resource should finally carry out the task.

```
const additionalResources = [
```

```

{ ID: "Res_Welding_unassigned", TableText: "Not yet assigned", _Color: "#FFD27F",
  SkillIDs: ["Skill_Welding"] },
{ ID: "Res_Turning_unassigned", TableText: "Not yet assigned", _Color: "#FFD27F",
  SkillIDs: ["Skill_Turning"] },
{ ID: "Res_Assembling_unassigned", TableText: "Not yet assigned", _Color: "#FFD27F",
  SkillIDs: ["Skill_Assembling"] },
];

```

Allocation Objects

An allocation describes the assignment of a task that requires specific skills to a resource that possesses these skills and is capable of performing the task.

```

let allocation =
{
  ID: `AL_ Res_Peter_1`,
  SkillID: "Skill_Assembling",
  ResourceID: "Res_Peter",
  Start: new Date(2028, 0, 6),
  End: new Date(2028, 0, 8),
  Color: "#ff6d00"
}

```

If the skill of the allocation matches the skill of the resource, the bar is displayed in the color specified for the allocation. Otherwise, the color is dimmed according to the **lightTonedDownOverlayColor** option.

```
lightTonedDownOverlayColor = "rgba(255,255,255,0.85)";
```

```

<!DOCTYPE html>
<html lang="de">
<head>
  <meta charset="utf-8"/>
  <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>
  <style>
    body { font-family: system-ui, Segoe UI, Arial, sans-serif; margin: 0; }
    #toolbar { padding: 10px 12px; border-bottom: 1px solid #e0e0e0; }
  </style>
</head>
<body>
  <div id="toolbar">
    <input style="margin-left: 20px" id="notYetAssigned" type="checkbox" checked/>
    <label for="notYetAssigned">Not yet assigned</label>

    <input style="margin-left: 20px" id="coloredShadowAllocations" type="checkbox" checked/>
    <label for="coloredShadowAllocations">Colored shadow allocations</label>

    <input style="margin-left: 20px" id="showSkillColorInRow" type="checkbox"/>
    <label for="showSkillColorInRow">Skill color in row</label>
  </div>
<div id="VSWidget"></div>
<script>
  Miscellaneous.ready(() => {
    const { ViewType, TimeType, BarShape, BarDesigns } = netronic.nVSW;

    document.getElementById("notYetAssigned").addEventListener("click", notYetAssigned);
    document.getElementById("showSkillColorInRow").addEventListener("click", showSkillColorInRow);
    document.getElementById("coloredShadowAllocations").addEventListener("click", coloredShadowAllocations);
  });

```

```

const start = new Date(2028, 0, 1);
const end = new Date(2028, 2, 1);

grayTonedDownOverlayColor = "rgba(240,240,240, 1.0)";
lightTonedDownOverlayColor = "rgba(255,255,255,0.85)";

const options = {
    licenseKey: window.VSW_LICENSE_KEY,
    start, end,
    viewType: ViewType.SkilledResources,
    tableViewWidth: 340,
    defaultCalendarID: "Cal_MonFri",
    firstDayOfWeek: 1,
    calendarGridColor: "rgba(220,220,220,1.0)",
    fixedTableColumnWidth: 0,
    topRowMarginInTimeArea: 1,
    bottomRowMarginInTimeArea: 0,
    defaultResourceMinimumRowHeight: 24,
    defaultAllocationBarHeight: 24,
    defaultSkillMinimumRowHeight: 24,
    subRowDistanceInTimeArea: 1,
    defaultAllocationBarDesign : BarDesigns.Text | BarDesigns.ComplexShape,

    tonedDownOverlayColor: lightTonedDownOverlayColor,
    allocationBarDesignOfOtherSkill: shadowDesign = netronic.nVSW.BarDesigns.Entries
    |

netronic.nVSW.BarDesigns.TonedDownColoring,
    onDrop: function (args) {
        vsw.processOnDrop(args);
    }
};

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

//region Calendar
function createMonFriEntries(rangeStart, rangeEnd) {
    const entries = [];
    const dayMs = 24 * 60 * 60 * 1000;
    let d = new Date(rangeStart.getFullYear(), rangeStart.getMonth(), rangeStart.getDate());
    const endDate = new Date(rangeEnd.getFullYear(), rangeEnd.getMonth(), rangeEnd.getDate());
    while (d < endDate) {
        const dow = d.getDay(); // 0=So, 1=Mo, ... 6=Sa
        if (dow >= 1 && dow <= 5) {
            entries.push({
                Start: new Date(d.getFullYear(), d.getMonth(), d.getDate(), 0, 0, 0),
                End: new Date(d.getFullYear(), d.getMonth(), d.getDate() + 1, 0, 0, 0),
                TimeType: TimeType.WorkingTime
            });
        }
        d = new Date(d.getTime() + dayMs);
    }
    return entries;
}
vsw.addCalendars([
    { ID: "Cal_MonFri", Entries: createMonFriEntries(start, end) }
]);
//endregion

//region Skills
const skillColors = ["#304ffe", "#009e8e", "#ff6d00"];
const skills = [
    { ID: "Skill_Welding", TableText: "Welding",
      TableColor: skillColors[0], TableTextColor: "white"},
    { ID: "Skill_Turning", TableText: "Turning",
      TableColor: skillColors[1], TableTextColor: "white"},
    { ID: "Skill_Assembling", TableText: "Assembling",
      TableColor: skillColors[2], TableTextColor: "white"}
];
vsw.addSkills(skills);
//endregion
//region Resources
const resources = [
    { ID: "Res_Peter", TableText: "Peter", _Color: "#FFD27F",
      SkillIDs: ["Skill_Welding", "Skill_Turning", "Skill_Assembling"] },
    { ID: "Res_Mary", TableText: "Mary", _Color: "#7EF5E0",

```

```

        SkillIDs: ["Skill_Welding", "Skill_Turning", "Skill_Assembling"] },
    { ID: "Res_Jack", TableText: "Jack", _Color: "#7EF5E0",
      SkillIDs: ["Skill_Assembling"]}
];

const additionalResources = [
    { ID: "Res_Welding_unassigned", TableText: "Not yet assigned",
      _Color: "#FFD27F", SkillIDs: ["Skill_Welding"] },
    { ID: "Res_Turning_unassigned", TableText: "Not yet assigned",
      _Color: "#FFD27F", SkillIDs: ["Skill_Turning"] },
    { ID: "Res_Assembling_unassigned", TableText: "Not yet assigned",
      _Color: "#FFD27F", SkillIDs: ["Skill_Assembling"] },
];
vsw.addResources([...additionalResources, ...resources]);
//endregion

//region Allocations
const allocations = [];
const startTimes = [
    // Peter
    [ new Date(2028, 0, 3),
      new Date(2028, 0, 10),
      new Date(2028, 0, 6)
    ],
    // Mary
    [ new Date(2028, 0, 6),
      new Date(2028, 0, 12),
      new Date(2028, 0, 18)
    ],
    // Jack
    [ new Date(2028, 0, 10),
      new Date(2028, 0, 18),
      new Date(2028, 0, 24)
    ]
];

const endTimes = [
    // Peter
    [new Date(2028, 0, 5),
      new Date(2028, 0, 12),
      new Date(2028, 0, 8)
    ],
    // Mary
    [new Date(2028, 0, 8),
      new Date(2028, 0, 15),
      new Date(2028, 0, 21)
    ],
    // Jack
    [new Date(2028, 0, 12),
      new Date(2028, 0, 22),
      new Date(2028, 0, 29)
    ]
];

const prefix = ["W", "T", "A"];
const counter = [1, 1, 1];

for (var i=0; i < resources.length; i++)
{
    for (var j=0; j < (startTimes[i]).length; j++)
    {
        const number = counter[j%3].toString().padStart(3, "0");
        const code = prefix[j%3] + number;
        counter[j%3]++;

        let allocation = {
            ID: `AL_${resources[i].ID}_${j}`,
            ResourceID: resources[i].ID,
            SkillID: resources[i].SkillIDs[j % 3],
            Start: startTimes[i][j],

```

```

        End: endTimes[i][j],
        TableText: `${resources[i].TableText}`,
        Color: skillColors[j % 3],
        BarText: code,
        BarTextColor: "white",
        BorderWidth: 0,
    }

    // Special case: Jack
    if (i == 2) {
        if (j == 2) // Jack skill is only Assembling
            allocation.SkillID = resources[i].SkillIDs[0]
        else
            continue;
    }
    allocations.push(allocation);
}
}
//endregion

vsw.addAllocations(allocations);
vsw.render();

function showSkillColorInRow(event) {
    skills.forEach(skill => {skill.TableColorVisibleInTimeArea = event.target.checked;});

    const newColors = event.target.checked ? ["#304ffe", "#009e8e", "#ff6d00"] :
                                              ["#304ffe", "#009e8e", "#ff6d00"];

    skills.forEach((skill, i) => {
        skill.TableColor = newColors[i];
    });

    vsw.updateSkills(skills);
    vsw.render();
}

function coloredShadowAllocations(event) {
    vsw.option(event.target.checked ? "tonedDownOverlayColor" :
                                         "tonedDownOverlayColor", lightTonedDownOverlayColor);
}

function notYetAssigned(event) {
    if (event.target.checked)
    {
        vsw.removeResources(resources)
        vsw.addResources([...additionalResources, ...resources]);
    }
    else
        vsw.removeResources(additionalResources);

    vsw.render();
}
});
</script>
</body>
</html>

```