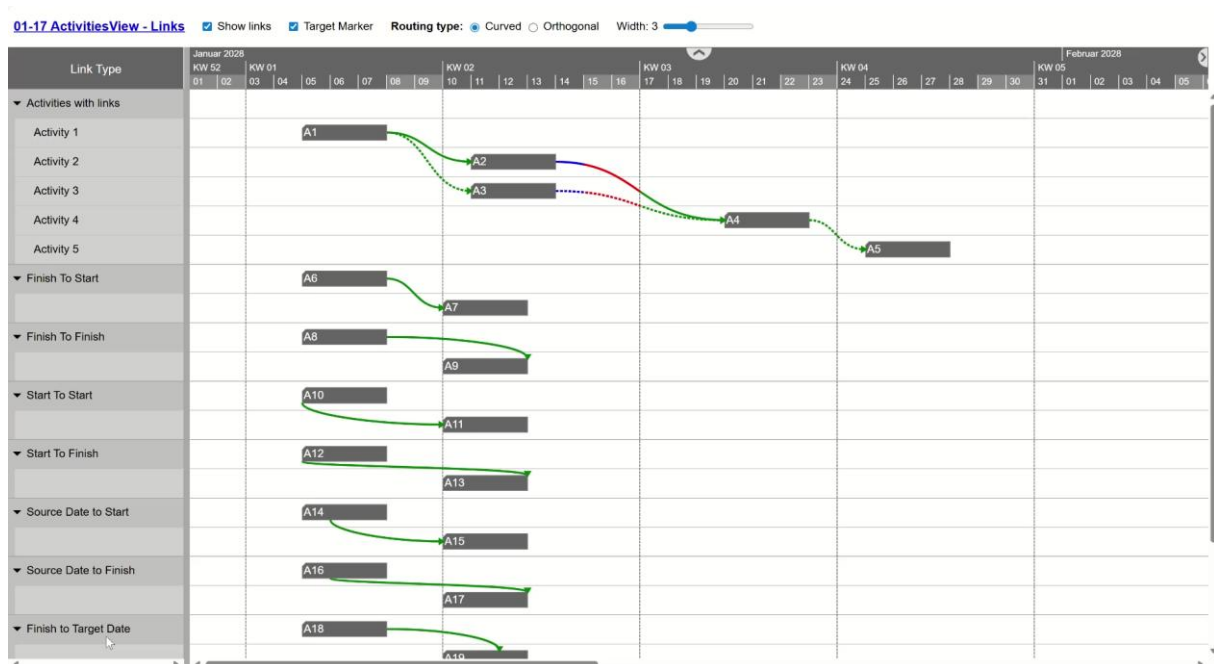


10-04 Activities view – Basics - Links

This example illustrates how to utilise links between activities in the ActivitiesView.



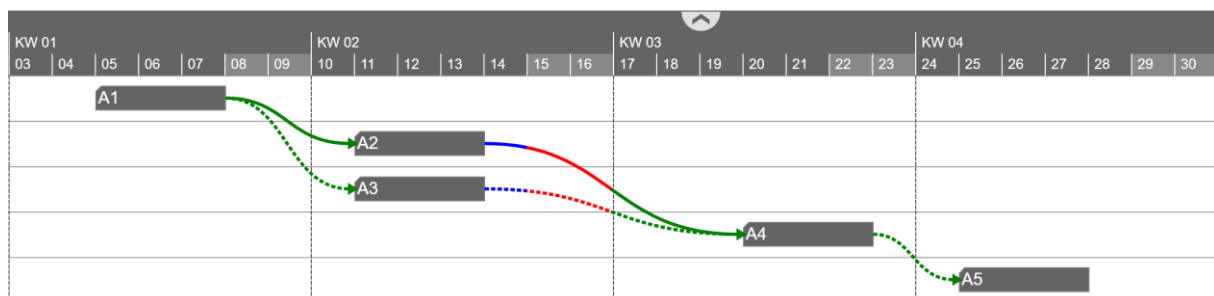
Links are used to establish connections between activities. These connections are directional (with a defined source and target), and the target end may optionally be indicated with an arrow-head (**TargetMarker**). Only one connection can exist between any two elements at a time.

Links serve a purely graphical purpose and are not semantically constrained within the library. It is the responsibility of the implementing application to utilize them appropriately according to the intended use case.

The general visibility of links within the **ActivitiesView** can be controlled via an option:

```
linksVisibleInActivitiesView: true,
```

The appearance of a link can be customized through properties such as **Color**, **Width**, and **DashArray**. The routing of the connection (**RoutingType**) can be either curved or orthogonal (right-angled).



The **Selectable** property of Links allows links to be made selectable, enabling actions to be performed on the selected set — for example, deleting a link via a context menu.

Some properties can be defined globally for all links via the Options settings, and can be overridden on an individual basis when needed:

```
defaultLinkRoutingType: LinkRoutingType.Curved,
defaultLinkSelectable: false,
defaultLinkTargetMarker: LinkMarker.FilledArrow,
defaultLinkTooltipTemplateID: "",
```

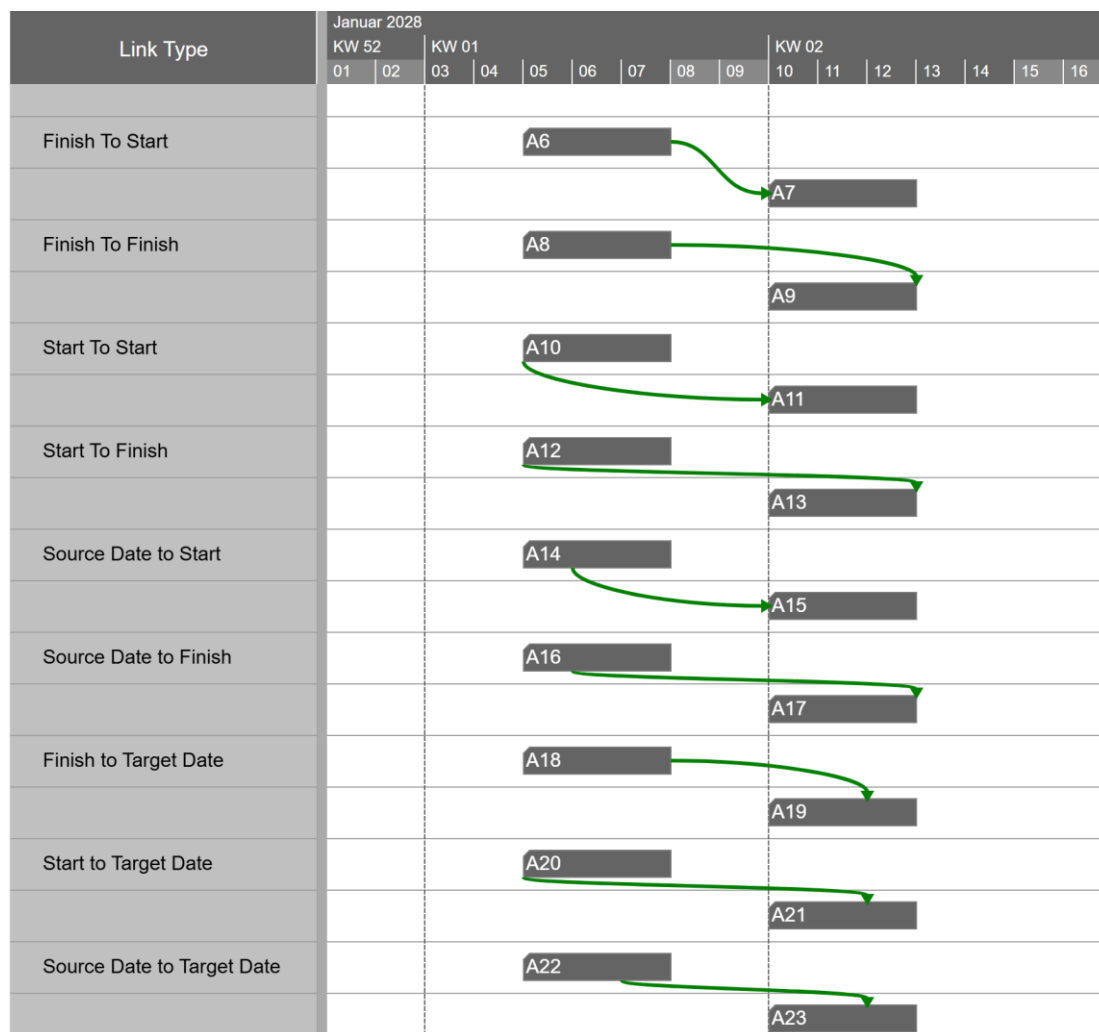
Other properties have fixed default values, which can only be modified individually for each link:

```
Color: black,
Width: 1,
DashArray: "none",
RelationType: RelationType.FinishToStart,
```

These properties are specific and must be set individually for each link:

```
ID: "L1",
SourceActivityID: "Act1",
TargetActivityID: "Act2",
```

The overview shows how different routing types affect the display of the links. It should be noted that some link types require the **LinkSourceDate** and **LinkTargetDate** properties to be set for Activity.



```

<!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 style="margin-left: 20px" id="showLinks" type="checkbox" checked/>
    <label for="showLinks">Show links</label>
    <input style="margin-left: 20px" id="showTargetMarker" type="checkbox" checked/>
    <label for="showTargetMarker">Target Marker</label>
    <label style="margin-left: 20px">
      <b>Routing type:</b>
    </label>
    <input type="radio" id="linkTypeCurved" name="routingType" checked/>
    <label for="linkTypeCurved">Curved</label>
    <input type="radio" id="linkTypeOrthogonal" name="routingType"/>
    <label for="linkTypeOrthogonal">Orthogonal</label>
    <label style="margin-left: 20px" for="linkWidth">
      Width:
      <span class="slider-output" id="output1">3</span>
    </label>
    <input type="range" id="linkWidth" min="1" max="8" value="3"
      data-output="output1" data-option="Width"/>
  </div>
<div id="VSWidget"></div>
<script>
  Miscellaneous.ready(() => {
    const { BarSortMode, BarDragModes, LinkRoutingType,
      LinkMarker, RelationType, RowDesigns } = netronic.nVSW;

    document.getElementById("showLinks").addEventListener("click", changeLinkDesign);
    document.getElementById("showTargetMarker").addEventListener("click", changeLinkDesign);
    document.getElementById("linkTypeCurved").addEventListener("click", changeLinkDesign);
    document.getElementById("linkTypeOrthogonal").addEventListener("click", changeLinkDesign);
    document.getElementById("linkWidth").addEventListener("input", changeLinkWidth);

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

    const options = {
      licenseKey: window.VSW_LICENSE_KEY,

      start: timeAreaStart,
      end: timeAreaEnd,

      defaultActivityExpandedRowDesign: RowDesigns.Bars,
      defaultActivityCollapsedRowDesign: RowDesigns.Bars |
        RowDesigns.BarsInHiddenDescendantRows |
        RowDesigns.BarsStacked,

      activityBarSortModeForStackedRowDesign: BarSortMode.StartAndEnd,
      defaultActivityAllowedBarDragModes: BarDragModes.None,
      defaultActivityBarSelectable: false,
      defaultActivityRowSelectable: false,
      multipleSelectionEnabled: false,
      defaultActivityTableRowDefinitionID: "CustomRowDef1",
      tableViewWidth: 300,
    };

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

```

```

const tableRowDefinitions = createTableRowDefinitions();
const activities = createActivities();
const links = createLinks();
vsWidget.addTableRowDefinitions(tableRowDefinitions);
vsWidget.addActivities(activities);
vsWidget.addLinks(links);
vsWidget.render();

function createTableRowDefinitions() {
  const tableRowDefinitions = [
    {
      ID: "CustomRowDef1",
      CellDefinitions: [
        {
          TitleText: "Link Type",
          Width: 250,
        },
      ],
    },
  ];

  return tableRowDefinitions;
}

function createActivities() {
  const activities = [
    {
      ID: "Act0",
      TableText: "Activities with links",
    },
    {
      ID: "Act1",
      ParentID: "Act0",
      TableText: "Activity 1",
      BarText: "A1",
      Start: new Date(2028, 0, 5),
      my_WorkingDays: 3,
    },
    {
      ID: "Act2",
      ParentID: "Act0",
      TableText: "Activity 2",
      BarText: "A2",
      Start: new Date(2028, 0, 11),
      my_WorkingDays: 3,
    },
    {
      ID: "Act3",
      ParentID: "Act0",
      TableText: "Activity 3",
      BarText: "A3",
      Start: new Date(2028, 0, 11),
      my_WorkingDays: 3,
    },
    {
      ID: "Act4",
      ParentID: "Act0",
      TableText: "Activity 4",
      BarText: "A4",
      Start: new Date(2028, 0, 20),
      my_WorkingDays: 3,
    },
    {
      ID: "Act5",
      ParentID: "Act0",
      TableText: "Activity 5",
      BarText: "A5",
      Start: new Date(2028, 0, 25),
      my_WorkingDays: 3,
    },
    {
      ID: "Act6",
      TableText: "Finish To Start",
      BarText: "A6",
    },
  ];
}

```

```

        Start: new Date(2028, 0, 5),
        my_WorkingDays: 3,
    },
    {
        ID: "Act7",
        ParentID: "Act6",
        TableText: "",
        BarText: "A7",
        Start: new Date(2028, 0, 10),
        my_WorkingDays: 3,
    },
    {
        ID: "Act8",
        TableText: "Finish To Finish",
        BarText: "A8",
        Start: new Date(2028, 0, 5),
        my_WorkingDays: 3,
    },
    {
        ID: "Act9",
        ParentID: "Act8",
        TableText: "",
        BarText: "A9",
        Start: new Date(2028, 0, 10),
        my_WorkingDays: 3,
    },
    {
        ID: "Act10",
        TableText: "Start To Start",
        BarText: "A10",
        Start: new Date(2028, 0, 5),
        my_WorkingDays: 3,
    },
    {
        ID: "Act11",
        ParentID: "Act10",
        TableText: "",
        BarText: "A11",
        Start: new Date(2028, 0, 10),
        my_WorkingDays: 3,
    },
    {
        ID: "Act12",
        TableText: "Start To Finish",
        BarText: "A12",
        Start: new Date(2028, 0, 5),
        my_WorkingDays: 3,
    },
    {
        ID: "Act13",
        ParentID: "Act12",
        TableText: "",
        BarText: "A13",
        Start: new Date(2028, 0, 10),
        my_WorkingDays: 3,
    },
    {
        ID: "Act14",
        TableText: "Source Date to Start",
        BarText: "A14",
        Start: new Date(2028, 0, 5),
        my_WorkingDays: 3,
        LinkSourceDate: new Date(2028, 0, 6),
    },
    {
        ID: "Act15",
        ParentID: "Act14",
        TableText: "",
        BarText: "A15",
        Start: new Date(2028, 0, 10),
        my_WorkingDays: 3,
    },
    {
        ID: "Act16",

```

```

        TableText: "Source Date to Finish",
        BarText: "A16",
        Start: new Date(2028, 0, 5),
        my_WorkingDays: 3,
        LinkSourceDate: new Date(2028, 0, 6),
    },
    {
        ID: "Act17",
        ParentID: "Act16",
        TableText: "",
        BarText: "A17",
        Start: new Date(2028, 0, 10),
        my_WorkingDays: 3,
    },
    {
        ID: "Act18",
        TableText: "Finish to Target Date",
        BarText: "A18",
        Start: new Date(2028, 0, 5),
        my_WorkingDays: 3,
    },
    {
        ID: "Act19",
        ParentID: "Act18",
        TableText: "",
        BarText: "A19",
        Start: new Date(2028, 0, 10),
        my_WorkingDays: 3,
        LinkTargetDate: new Date(2028, 0, 12),
    },
    {
        ID: "Act20",
        TableText: "Start to Target Date",
        BarText: "A20",
        Start: new Date(2028, 0, 5),
        my_WorkingDays: 3,
    },
    {
        ID: "Act21",
        ParentID: "Act20",
        TableText: "",
        BarText: "A21",
        Start: new Date(2028, 0, 10),
        my_WorkingDays: 3,
        LinkTargetDate: new Date(2028, 0, 12),
    },
    {
        ID: "Act22",
        TableText: "Source Date to Target Date",
        BarText: "A22",
        Start: new Date(2028, 0, 5),
        my_WorkingDays: 3,
        LinkSourceDate: new Date(2028, 0, 7),
    },
    {
        ID: "Act23",
        ParentID: "Act22",
        TableText: "",
        BarText: "A23",
        Start: new Date(2028, 0, 10),
        my_WorkingDays: 3,
        LinkTargetDate: new Date(2028, 0, 12),
    },
    ],
];

const millisecondsPerDay = 24 * 60 * 60 * 1000;
for (const activity of activities)
    if (activity.Start) {
        activity.End = new Date(activity.Start.getTime() +
            (activity.my_WorkingDays ?? 1) * millisecondsPerDay);
    }

return activities;
}

```

```

function createLinks() {
  const millisecondsPerDay = 24 * 60 * 60 * 1000;
  const links = [
    {
      ID: "L1",
      SourceActivityID: "Act1",
      TargetActivityID: "Act3",
      Color: "green",
      DashArray: "4 2",
      Width: 3,
    },
    {
      ID: "L2",
      SourceActivityID: "Act1",
      TargetActivityID: "Act2",
      Color: "green",
      Width: 3,
    },
    {
      ID: "L3",
      SourceActivityID: "Act3",
      TargetActivityID: "Act4",
      Color: "green",
      Width: 3,
      DashArray: "4 2",
      Entries: [
        {
          Color: "blue",
          Duration: millisecondsPerDay,
        },
        {
          Color: "red",
          Duration: 2 * millisecondsPerDay,
        },
      ],
    },
    {
      ID: "L4",
      SourceActivityID: "Act2",
      TargetActivityID: "Act4",
      Color: "green",
      Width: 3,
      Entries: [
        {
          Color: "blue",
          Duration: millisecondsPerDay,
        },
        {
          Color: "red",
          Duration: 2 * millisecondsPerDay,
        },
      ],
    },
    {
      ID: "L5",
      SourceActivityID: "Act4",
      TargetActivityID: "Act5",
      Color: "green",
      DashArray: "4 2",
      Width: 3,
    },
    {
      ID: "L6",
      SourceActivityID: "Act6",
      TargetActivityID: "Act7",
      Color: "green",
      Width: 3,
      RelationType: RelationType.FinishToStart,
    },
    {
      ID: "L7",
      SourceActivityID: "Act8",
      TargetActivityID: "Act9",
    },
  ]
}

```

```

        Color: "green",
        Width: 3,
        RelationType: RelationType.FinishToFinish,
    },
    {
        ID: "L8",
        SourceActivityID: "Act10",
        TargetActivityID: "Act11",
        Color: "green",
        Width: 3,
        RelationType: RelationType.StartToStart,
    },
    {
        ID: "L9",
        SourceActivityID: "Act12",
        TargetActivityID: "Act13",
        Color: "green",
        Width: 3,
        RelationType: RelationType.StartToFinish,
    },
    {
        ID: "L10",
        SourceActivityID: "Act14",
        TargetActivityID: "Act15",
        Color: "green",
        Width: 3,
        RelationType: RelationType.SourceDateToStart,
    },
    {
        ID: "L11",
        SourceActivityID: "Act16",
        TargetActivityID: "Act17",
        Color: "green",
        Width: 3,
        RelationType: RelationType.SourceDateToFinish,
    },
    {
        ID: "L12",
        SourceActivityID: "Act18",
        TargetActivityID: "Act19",
        Color: "green",
        Width: 3,
        RelationType: RelationType.FinishToTargetDate,
    },
    {
        ID: "L13",
        SourceActivityID: "Act20",
        TargetActivityID: "Act21",
        Color: "green",
        Width: 3,
        RelationType: RelationType.StartToTargetDate,
    },
    {
        ID: "L14",
        SourceActivityID: "Act22",
        TargetActivityID: "Act23",
        Color: "green",
        Width: 3,
        RelationType: RelationType.SourceDateToTargetDate,
    },
    ],
    return links;
}

function changeLinkDesign(event) {
    const checkboxOrRadio = event.target;

    if (checkboxOrRadio.id == "showLinks")
        vsWidget.option("linksVisibleInActivitiesView", checkboxOrRadio.checked);
    else if (checkboxOrRadio.id == "showTargetMarker") {
        vsWidget.option("defaultLinkTargetMarker", checkboxOrRadio.checked ?
            LinkMarker.FilledArrow : LinkMarker.None);
    } else if (checkboxOrRadio.name == "routingType") {

```



```

        vsWidget.option("defaultLinkRoutingType", document.getElementById("linkTypeCurved").checked ?
                                                                    LinkRoutingType.Curved : LinkRoutingType.Orthogonal);
    }
}

function changeLinkWidth(event) {
    const slider = event.target;

    const value = slider.value;

    document.getElementById(slider.dataset.output).innerText = value.toString().padStart(3, " ");

    for (const link of links)
        link.Width = +value;

    vsWidget.updateLinks(links);
    vsWidget.render();
}
});
</script>
</body>
</html>

```