

Integration Guide for the NETRONIC Visual Scheduling Widget

This little document will show you how to create your first application including the VSW. Only a few steps are needed as described below! If you want to use the ESM approach, jump to chapter 2.

1 Classic approach with or without jQuery UI

To integrate the Visual Scheduling Widget (VSW) into an HTML page the following steps must be done:

1. Add the following references to CSS and JavaScript files in this order:

```
<link href="nwaf-apptools.min.css" rel="stylesheet" />
<link href="nwaf-table.min.css" rel="stylesheet" />
<link href="nwaf-gantt.min.css" rel="stylesheet" />
<link href="nwaf-rab.min.css" rel="stylesheet" />

<script
src="https://cdnjs.cloudflare.com/ajax/libs/hammer.js/2.0.8/hammer.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/d3/7.9.0/d3.min.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/tinycolor/1.6.0/tinycolor.min.js"></script>
<script src="nwaf-apptools.min.js"></script>
<script src="nwaf-table.min.js"></script>
<script src="nwaf-gantt.min.js"></script>
<script src="nwaf-rab.min.js"></script>
```

For more details, please refer to chapter **System Requirements** of the Interface Definition Document (see VSW_SE_IDD.pdf). You can either request the 3rd party libraries from a CDN server on the Internet or from your local web server. Optionally additional 3rd party libraries are needed that you will have to include if you want to use some special features of VSW like time zone support or PDF export.

2. Add a <div> element to your HTML page. This element will be extended by the VSW to generate the diagram:

```
<div id="VSWidget" style="width:1000px; height:300px"></div>
```

3. Inside your code inside of a JavaScript file you instantiate the widget:

```
const { VSWidget, ViewType } = netronic.nVSW;

const options = {
  "licenseKey": "--- PLEASE INSERT YOUR LICENSE KEY HERE ---",
```

```

    "start": new Date("2030-01-01"),
    "end": new Date("2030-07-01"),
    "viewType": ViewType.Activities /* or alternatively ViewType.Resources */
  };
  const vsWidgetInstance = new VSWidget(document.querySelector("#VSWidget"), options);
  vsWidgetInstance.render(); /* now a very first diagram without any data inside
                             will be shown */

```

You will have to fill in the license key that is provided in an additional license file inside the package. If something goes wrong here, you will only see a placeholder image! In this case or if your license has expired, please contact your sales partner at Boyum IT.

Alternatively, you can use the VSW as a jQuery UI Widget. Then you will have to include the jQuery libraries and instantiate the widget by using

```
const vsWidgetInstance = $("#VSWidget").nVSWidget(options).nVSWidget("instance");
```

For explanation of the usage of jQuery UI Widgets please see the official documentation: <http://api.jqueryui.com/jquery.widget/>.

4. Now, you can add activities as follows:

```

vsWidgetInstance.addActivities([
  {
    "ID": "Act1",
    "BarText": "Activity 1",
    "TableText": "Activity 1",
    "Start": new Date("2030-01-01"),
    "End": new Date("2030-01-03")
  }
]);
vsWidgetInstance.render();

```

If you want to work with the resources view, then you can add resources and allocations:

```

vsWidgetInstance.addResources([
  {
    "ID": "Res1",
    "TableText": "Resource 1"
  }
]);
vsWidgetInstance.addAllocations([
  {
    "ID": "Alloc1",
    "BarText": "Allocation 1",
    "ResourceID": "Res1",
    "Entries": [
      {
        "Start": new Date("2030-01-01"),
        "End": new Date("2030-01-03")
      }
    ]
  }
]);
vsWidgetInstance.render();

```

Please note: To display the resources view you either must change the option *viewType*:

```
vsWidgetInstance.option("viewType", ViewType.Resources);
```

or of course, you can set that option when instantiating the VSW.

The file index.html can be started from the file system by double-clicking on the file index.html e.g. in the File Explorer (Windows) or Finder (macOS) for your first tests.

Good luck!

2 ESM approach

To integrate the Visual Scheduling Widget (VSW) into an HTML page the following steps must be done:

1. Add the following import map with references to libraries:

```
<script type="importmap">
  {
    "imports": {
      "hammerjs": "https://cdn.jsdelivr.net/npm/hammerjs@2.0.8/+esm",
      "d3-array": "https://cdn.jsdelivr.net/npm/d3@7.9.0/+esm",
      "d3-axis": "https://cdn.jsdelivr.net/npm/d3@7.9.0/+esm",
      "d3-ease": "https://cdn.jsdelivr.net/npm/d3@7.9.0/+esm",
      "d3-format": "https://cdn.jsdelivr.net/npm/d3@7.9.0/+esm",
      "d3-interpolate": "https://cdn.jsdelivr.net/npm/d3@7.9.0/+esm",
      "d3-scale": "https://cdn.jsdelivr.net/npm/d3@7.9.0/+esm",
      "d3-shape": "https://cdn.jsdelivr.net/npm/d3@7.9.0/+esm",
      "d3-selection": "https://cdn.jsdelivr.net/npm/d3@7.9.0/+esm",
      "d3-time": "https://cdn.jsdelivr.net/npm/d3@7.9.0/+esm",
      "d3-time-format": "https://cdn.jsdelivr.net/npm/d3@7.9.0/+esm",
      "d3-transition": "https://cdn.jsdelivr.net/npm/d3@7.9.0/+esm",
      "tinycolor2": "https://cdn.jsdelivr.net/npm/tinycolor2@1.6.0/+esm",
      "nwaf_apptools": "nwaf-apptools.esm.min.mjs",
      "nwaf_table": "nwaf-table.esm.min.mjs",
      "nwaf_gantt": "nwaf-gantt.esm.min.mjs",
      "nwaf_rab": "nwaf-rab.esm.min.mjs"
    }
  }
</script>
<script src="main.mjs" type="module"></script>
```

For more details, please refer to chapter **System Requirements** of the Interface Definition Document (see VSW_SE_IDD.pdf). You can either request the 3rd party libraries from a CDN server on the Internet or from your local web server. Optionally additional 3rd party libraries are needed that you will have to include if you want to use some special features of VSW like time zone support or PDF export.

2. Add a <div> element to your HTML page. This element will be extended by the VSW to generate the diagram:

```
<div id="VSWidget" style="width:1000px; height:300px"></div>
```

3. Inside your code (e.g. a file named main.mjs) you instantiate the widget:

```
import { VSWidget, ViewType } from "nwaf_rab";

const options = {
  "licenseKey": "--- PLEASE INSERT YOUR LICENSE KEY HERE ---",
  "start": new Date("2030-01-01"),
  "end": new Date("2030-07-01"),
  "viewType": ViewType.Activities /* or alternatively ViewType.Resources */
};
const vsWidgetInstance = new VSWidget(document.querySelector("#VSWidget"), options);
vsWidgetInstance.render(); /* now a very first diagram without any data inside
                             will be shown */
```

You will have to fill in the license key that is provided in an additional license file inside the package. If something goes wrong here, you will only see a placeholder image! In this case or if your license has expired, please contact your sales partner at Boyum IT.

4. Now, you can add activities as follows:

```
vsWidgetInstance.addActivities([{\n  "ID": "Act1",\n  "BarText": "Activity 1",\n  "TableText": "Activity 1",\n  "Start": new Date("2030-01-01"),\n  "End": new Date("2030-01-03")\n}]);\nvsWidgetInstance.render();
```

If you want to work with the resources view, then you can add resources and allocations:

```
vsWidgetInstance.addResources([{\n  "ID": "Res1",\n  "TableText": "Resource 1"\n}]);\nvsWidgetInstance.addAllocations([{\n  "ID": "Alloc1",\n  "BarText": "Allocation 1",\n  "ResourceID": "Res1",\n  "Entries": [{\n    "Start": new Date("2030-01-01"),\n    "End": new Date("2030-01-03")\n  }],\n}]);\nvsWidgetInstance.render();
```

Please note: To display the resources view you either must change the option *viewType*:

```
vsWidgetInstance.option("viewType", ViewType.Resources);
```

or of course, you can set that option when instantiating the VSW.

While the classic approach can be started from the file system by double-clicking on the file `index.html` e.g. in the File Explorer (Windows) or Finder (macOS), the ESM approach can only be started from a web server!

Good luck!