

# opCharts - Customising Table Columns

- [Feature Description](#)
- [Views That Support Custom Columns](#)
- [Available Columns](#)
  - [Interface View](#)
  - [Node View & Node Context - Node Info Widget](#)
  - [Scheduled Outages View](#)
- [Configuration](#)
  - [Configuration Files](#)
  - [Enabling the Feature](#)
  - [Configuration](#)
  - [Property Attributes](#)
    - [Attribute Descriptions](#)
  - [Adding and Removing Columns](#)
- [Verification](#)
  - [JSON File Syntax Check](#)
  - [Testing View Functionality](#)

## Feature Description

This feature is available from opCharts version 3.2.2 and newer. This is an advanced feature that allows customers to modify the data presented in tables. The default table content has been carefully selected based on Opmantek's numerous years of network management experience. This said there are cases where customers would like to alter the table data to display information that is important to their organisation. Most opCharts pages with table data can be customised to display different content. This feature allows table columns to be removed or added as required.



If when following these steps the View that is being modified does not behave normally; remove the custom configuration file and normality should be restored.

## Views That Support Custom Columns

- [Interfaces](#)
- [Nodes](#)
- [Scheduled Outages](#)
- [Node Context - Node Info Widget](#)

## Available Columns

### Interface View

Executing an API call such as the following will reveal the available properties that can be made into columns for the Interface view.

```
### API call for interface properties
http://<server_name>/omk/opCharts/v1/nodes/<node_uuid>?properties=["info.system.name","info.interface"]

### API call for node UUID's
http://<server_name>/omk/opCharts/v1/nodes?properties=["info.system.name","info.system.uuid"]
```

### Node View & Node Context - Node Info Widget

Executing an API call such as the following will reveal the available properties that can be made into columns for the Node view as well as the Node Context - Node Info Widget.

```
### API call for node properties
http://<server_name>/omk/opCharts/v1/nodes/9b3bd935-9a7c-11e7-81dd-cfe2a8622c76?properties=["info.system"]

### API call for node UUID's
http://<server_name>/omk/opCharts/v1/nodes?properties=["info.system.name","info.system.uuid"]
```

For opCharts 4, the API call will be different:

```
http://<server_name>/omk/opCharts/v2/nodes/dcab7ffb-2c81-4ba7-91be-f85325d94f41?properties=["configuration"]
```

Here you can check the [common node properties](#).

## Scheduled Outages View

The properties that may be used for columns for the Scheduled Outages view may be found in the `/usr/local/nmis8/conf/Outages.nmis` file.

# Configuration

## Configuration Files

Each view has a separate configuration file that will be found in the following directory (by default this directory does not exist; reference the next step):

`/usr/local/omk/conf/table_schemas`

### opCharts-3.x

View	Configuration file
Interfaces	opCharts_interface-list.json
Nodes	opCharts_node-list.json
Scheduled Outages	opCharts_outage-schema.json
Node Context - Node Info Widget	opCharts_node-summary-table.json

### opCharts-4.x

View	Configuration file
Interfaces	opCharts_interface-list.json
Nodes	opCharts_node-list.json
Scheduled Outages	opCharts_outage-schema.json
Node Context - Node Info Widget	opCharts_node-summary-table.json
Business Service - Nodes	opCharts_business-services-nodes.json

## Enabling the Feature

In order to enable this feature the following must be done.

- Create a directory called `/usr/local/omk/conf/table_schemas`
- Copy the specific view configuration file that requires modification from `/usr/local/omk/lib/json/opCharts/table_schemas/` into `/usr/local/omk/conf/table_schemas`.
  - Only the necessary json files should be copied to the `/usr/local/omk/conf/table_schemas` directory as having unnecessary config files in this directory will result in future upgrades being unpredictable.



### Caution!

Enable this feature with care.

Future opCharts upgrades will need to be watched carefully as tables and node properties can change across versions. Based on this an upgrade has the potential to break the functionality of a custom table configuration. If this feature is enabled it is highly recommend to upgrade in a test environment prior to upgrading the production environment.

## Configuration

The configuration files are json files that have a specific syntax that must be observed. Add the desired property to the json file in the order it should appear in. The table will be constructed left to right based on attributed that are read from the top down.

## Property Attributes

Each property that is added will require a set of attributes. This is an example of the attributes that belong to the 'configuration.country' property.

```
{ "name": "configuration.country",  
  "label": "Country",  
  "cell": "String"  
},
```

## Attribute Descriptions

- name: Name of the node property
- label: The column name that will render in the web page.
- cell: The cell type, usually this will be "string". For other options please contact [support@opmantek.com](mailto:support@opmantek.com).

## Adding and Removing Columns

To remove a column simply remove the associated section from the applicable json file. To add a column add a new section in the json file. The column placement will be relative to the order it is put in the json file.

The example below is the opCharts\_node-list.json file. The version on the left is the default version. The version on the right adds the Country column between the Node Status and Group columns, it also removes the Links column.

### Default - opCharts\_node-list.json

```
// VERSION=1.8.0
[
  {
    "name": "node_name",
    "label": "Name",
    "cell": "NodeLinkCell",
    "renderable": 1,
    "comment": "must be present for NodeLinkCell
to work on any column, use 'renderable': 0 to hide"
  },
  {
    "name": "configuration.host",
    "label": "Host",
    "cell": "String"
  },
  {
    "name": "",
    "label": "Links",
    "cell": "UriButtonGroup",
    "links": [
      {
        "name": "remote_connection_details.url",
        "label": "remote_connection_details.label",
        "glyph": "fa fa-shield"
      },
      {
        "name": "custom_context_details.url",
        "label": "custom_context_details.label", "glyph":
"fa fa-link"
      }
    ],
    "sort": false,
    "button_group_class": "btn-group btn-group-xs"
  },
  {
    "name": "node_summary.nodestatus",
    "label": "Node Status",
    "cell": "NodeStatusCell"
  },
  {
    "name": "configuration.group",
    "label": "Group",
    "cell": "String"
  },
  {
    "name": "node_summary.nodeType",
    "label": "Node Type",
    "cell": "String"
  },
  {
    "name": "configuration.roleType",
    "label": "Role",
    "cell": "String"
  },
  {
    "name": "node_summary.nodeVendor",
    "label": "Vendor",
    "cell": "String"
  },
  {
    "name": "node_summary.sysLocation",
    "label": "Location",
    "cell": "String"
  },
  {
    "name": "node_summary8.health",
    "label": "Health",
    "cell": "ColouredByLevelCell",
    "levels": [ "green", 100, "yellow", 99,
"orange", 80, "red", 0 ]
  },
  {
    "name": "node_summary.lastUpdateSec",
    "label": "Last Update",
    "cell": "String",
    "formatter": "UnixTimeFormatter"
  }
]
]
```

### Example - opCharts\_node-list.json

```
// VERSION=1.8.0
[
  {
    "name": "node_name",
    "label": "Name",
    "cell": "NodeLinkCell",
    "renderable": 1,
    "comment": "must be present for NodeLinkCell
to work on any column, use 'renderable': 0 to hide"
  },
  {
    "name": "configuration.host",
    "label": "Host",
    "cell": "String"
  },
  {
    "name": "node_summary.nodestatus",
    "label": "Node Status",
    "cell": "NodeStatusCell"
  },
  {
    "name": "configuration.country",
    "label": "Country",
    "cell": "String"
  },
  {
    "name": "configuration.group",
    "label": "Group",
    "cell": "String"
  },
  {
    "name": "node_summary.nodeType",
    "label": "Node Type",
    "cell": "String"
  },
  {
    "name": "configuration.roleType",
    "label": "Role",
    "cell": "String"
  },
  {
    "name": "node_summary.nodeVendor",
    "label": "Vendor",
    "cell": "String"
  },
  {
    "name": "node_summary.sysLocation",
    "label": "Location",
    "cell": "String"
  },
  {
    "name": "node_summary8.health",
    "label": "Health",
    "cell": "ColouredByLevelCell",
    "levels": [ "green", 100, "yellow", 99,
"orange", 80, "red", 0 ]
  },
  {
    "name": "node_summary.lastUpdateSec",
    "label": "Last Update",
    "cell": "String",
    "formatter": "UnixTimeFormatter"
  }
]
]
```

# Verification

## JSON File Syntax Check

The syntax of a json file may be tested with the following command.

```
json_xs -t null < somefile.json
```

Please note that the version line at the top of the file will need to be removed in order to test in the manner.

```
### Test with version line
[root@opmantek table_schemas]# json_xs -t null < opCharts_node-list.json
malformed JSON string, neither array, object, number, string or atom, at character offset 0 (before "//
VERSION=1.8.0\n[\n...") at /usr/bin/json_xs line 138, <STDIN> line 1.

### Successful test without the version line
[root@opmantek table_schemas]# json_xs -t null < opCharts_node-list.json.checksyntax
[root@opmantek table_schemas]#
```

## Testing View Functionality

It is not necessary to restart any daemons. After editing the associated json file simply load (or reload) the view in question.

For more information on customizing fields with NMIS integration view our Wiki page Here: [NMIS/opCharts Integration - Custom Fields](#)