opCharts Form Schema

Table of Contents

Summary

> **Version**
>
> opCharts 4.4.2

We are providing ways for customers to produce custom form using form schema which can be read by certain components in opCharts to show additional fields which can be configured to the users liking.

These files are stored in `conf/form_schemas` and are json encoded schema which will tell our form system how to render.

Currently there is no way to get dynamic data for helping drive select lists for example but this could change in the future, let us know what you want to see.

Form Schemas

Supported Administration Components

- Nodes

Supported opCharts Components

- Element Lists

Example Form Schema

conf/form_schemas/opCharts_demo.json

```
{
    "label": "Demo",
    "description": "Demo form schema",
    "component": ["element_lists"],
    "tags": ["interface"],
    "schema": {
        "demo1": {
            "title": "My Title",
            "type": "Text"
        },
        "demo2": {
            "title": "Another Title",
            "type": "Select",
            "options": ["option1", "option2"]
        }
    }
}
```

Filename

The filename must be prefixed with the application name, eg opCharts_

The filename must have .json at the end and only be a-Z 0-9 ascii characters.

Part of the filename will be saved to schema documents so changing the filename at a later date will break linkage.

Form Metadata

key	type	required	description
label	String	yes	Title of the form schema which will be shown in the gui
description	String	no	Give context to your users about what this form is for
component	array [string]	no	Allow the form schema to show for different components, is an array so a form could be shared over more than one component, if this is not defined then it will show over more than type of component.
tags	array [string]	no	We use this in element_inventory to then filter down form types for different inventory, its an array so you can share this form over multiple types of inventory.
schema	object	yes	Deep structure to represent the form
schema.<key>	nested key	yes	Defines what the value will be saved under
schema.<key>.type	string	no	Defines the type of field type to be rendered, if not defined it will default to Text
schema.<key>.title	string	no	Defines the text that appears in a form field's label
schema.<key>.help	string	no	Help text to add next to the editor.
schema.<key>.validators	array[]	no	A list of validators, see validation below
schema.<key>.order	integer	no	Defines the order of the fields to be displayed on the form. It is recommended you use this for every key so that fields have a fixed order.

Supported Field Types

Schema Value	Additional Options
`Text`	
`Number`	
`Password`	
`TextArea`	
Checkbox	
Select	
Radio	

Date	`yearStart`
	First year in the list. Default: 100 years ago.
	`yearEnd`
	Last year in the list. Default: current year.

DateTime	

Validation

type	description

required	Checks the field has been filled in.
number	Checks it is a number, allowing a decimal point and negative values.
range	Checks it is a number in a range defined by `min` and `max` options. Message if it is not a number can be set with the `numberMessage` option.
email	Checks it is a valid email address.
url	Checks it is a valid URL.
match	Checks that the field matches another. The other field name must be set in the `field` option.
regexp	Runs a regular expression. Requires the `regexp` option, which takes a compiled regular expression or a string value. Setting the `match` option to `false` ensures that the regexp does NOT pass.

See Backbone Form Validation for examples and further docs.

https://github.com/powmedia/backbone-forms#validation

```
{
    "label": "Customer Data",
    "description": "Edit in conf/form_schemas/opCharts_element_inventory_customer.json",
    "component": ["element_lists"],
    "tags": ["interface"],
    "schema": {
        "customerId": {
            "title": "Customer ID",
            "type": "Text",
            "order": 1
        },
        "upStream": {
            "title": "Upstream Contract Speed",
            "type": "Number",
            "validators": [
                { "type": "range", "min": 1, "max": 10000, "message": "Mbps should be between 1 and 10000" }
            ],
            "order": 2
        },
        "downStream": {
            "title": "Downstream Contract Speed",
            "type": "Number",
            "validators": [
                { "type": "range", "min": 1, "max": 10000, "message": "Mbps should be between 1 and 10000" }
            ],
            "order": 3
        }
    }
}
```

Example for adding Maintenance Tracking to entityMib Data

The following is an example form schema so you can track the maintenance renewals of your serial number items using opCharts Element Lists.

The following would be located in /usr/local/omk/conf/form_schemas and in testing was named opCharts_element_lists_serial_numbers.json

```
{
    "label": "Serial number tracking for EntityMIB Data",
    "description": "Edit in conf/form_schemas/opCharts_element_lists_serial_numbers.json",
    "component": ["element_lists"],
    "tags": ["entityMib"],
    "schema": {
        "under_maintenance": {
            "title": "Under Maintenance",
            "help": "Do we have a maintenance contract for this element or not?",
            "type": "Select",
            "options": ["","Yes", "No"]
        },
        "maintenance_contract": {
            "title": "Maintenance Contract",
            "type": "Text"
        },
        "start_date": {
            "title": "Start Date",
            "yearStart": 2010,
            "yearEnd": 2030,
            "type": "Date"
        },
        "renewal_date": {
            "title": "Renewal Date",
            "yearStart": 2020,
            "yearEnd": 2050,
            "type": "Date"
        }
    }
}
```