

Table of contents

- [Table of contents](#)
- [JSON-Wizard](#)
 - [Description](#)
 - [Compatibility](#)
 - [Getting Started](#)
 - [Output Connections](#)
 - [Flow Element](#)
 - [General Properties](#)
 - [Working Mode Dependend Properties](#)
 - [Actions Syntax](#)
 - [Examples](#)
 - [Create](#)
 - [Create Key in Object](#)
 - [Create Key in Array](#)
 - [Insert Element in Array](#)
 - [Update](#)
 - [Update Key in Object](#)
 - [Update Key in Array](#)
 - [Update Element in Array](#)
 - [Create or Update](#)
 - [Create or Update Key in Array](#)
 - [Create or Update Element in Array](#)
 - [Delete](#)
 - [Remove Key from Object](#)
 - [Remove Key from Array](#)
 - [Remove Elements from Array](#)
 - [Additional Datasets / Files + JSONata](#)
 - [Fallback](#)
 - [Access Current Element](#)
 - [Error handling](#)
 - [Private data](#)

JSON-Wizard

Description

JSON Wizard makes it easy to modify and process any type of JSON in an Enfocus Switch environment.

Whether you want to:

- Add new keys to an object or array,
- Push elements into an array,

- Update existing keys in objects or arrays,
- Delete keys, objects, or arrays,
- Update or delete keys, objects, or arrays,

-- all this is possible with a little JSON magic 🤖

Compatibility

Switch Fall 2022 and higher.

Getting Started

Use one of our sample flows and drop a sample file into the flow. Compare the input JSON with the output JSON and checkout the executed actions.

Output Connections

This app requires one incoming connection - more incoming connections are allowed. The app supports traffic light outgoing connections of the following types:

- Data error: carries the incoming job if the operation fails at the first attempt.
- Data success: carries the incoming or injected job after the operation succeeds. If there are no data success connections the output is simply suppressed (with logging a warning).
- Log error: carries a small .log file named after the incoming job containing relevant information about the operations.
- Log success: carries a small .log file named after the incoming job containing relevant information about the operations.

Flow Element

General Properties

Property	Value	Description
<i>Working mode</i>	enum [JSON-File JSON-Dataset External JSON-File]	
<i>Actions</i>	String[]	A list of actions that should be executed. Go to <i>Actions Syntax</i>
<i>Set private data</i>	String[]	Allows to set private data using a JSON Path expression of the resulting JSON; e.g.: <code>myOrderID=\$..[0].OrderID</code>
<i>Additional datasets</i>	String[]	The content of the dataset will be added to the master JSON and can be accessed by a JSONPATH expression like \$... The added objects will be removed after all queries are done.
<i>Additional file</i>	String	Allows to define an additional JSON file which will be appended to the master JSON. This can be useful if information from this file must be retrieved. The added objects will be removed after all queries are done.
<i>Object name</i>	String	Defines the name of the appended object of the additional file.

Working Mode Dependend Properties

JSON-Dataset

Property	Value	Description
<i>Master dataset name</i>	String	Name of the dataset that should be processed

External JSON-File

Property	Value	Description
<i>File path</i>	String	Path to external JSON file

Property	Value	Description
<i>Delete after injection</i>	Boolean	Defines if the external file should be deleted or not after it is injected as Switch job
<i>Append as dataset</i>	Boolean	Defines if the JSON should be appended as dataset to the incoming job
<i>Dataset name</i>	String	If <code>Append as Dataset = true</code> ; the dataset name of the JSON
<i>Job name</i>	String	Default: The name proper of the incoming job; If <code>Append as Dataset = false</code> ; the name of the injected job.

Actions Syntax

Comment: `//` at the beginning of a line **Separator:** `||`

Actions

- **Create:** `+` => Creates new key-value pairs in objects or arrays; pushes new elements to array.
- **Update:** `&` => Changes values of existing keys in objects or arrays
- **Delete:** `-` => Removes key-value pairs; removes elements from array
- **Create or Update:** `?` => Combines the functionality of Create and Update
 - if a key does not exist it will be created
 - if a key exists it will be updated

JSONPath: A valid JSONPath expression; we recommend using <https://jsonpath.com/> to test your queries; checkout <https://www.npmjs.com/package/jsonpath> and <https://www.npmjs.com/package/jsonpath-plus> for more information

- **Data types:** string|boolean|number|json|jsonata
- **Value:**

In case of `DataType=json` a stringified json object/array, else any string (it will be parsed according the selected `DataType`).

The value can also define a `JSONPath` that queries the current JSON. In this case the value has to start with `$`

If `DataType=jsonata` the value has to define a valid JSONata expression, which can be tested by the JSON Exerciser: <https://try.jsonata.org/>

If `DataType=jsonata` or if the value is defined as JSONPath a fallback can be defined using `##` after the main query

- **Key:** The key to be created in the object/array - only needed for action 'Create'

Examples

Example JSON:

```
{
  "Account": {
    "Account Name": "Firefly",
    "Order": [
      {
        "OrderID": "order103",
        "Product": [
          {
            "Product Name": "Bowler Hat",
            "ProductID": 858383,
            "Quantity": 2
          },
          {
            "Product Name": "Trilby hat",
            "ProductID": 858236
          }
        ]
      },
      {
        "OrderID": "order104",
        "Product": [
          {
            "Product Name": "Bowler Hat",
            "ProductID": 858383,
            "Quantity": 4
          },
          {
            "ProductID": 345664,
            "Product Name": "Cloak",
            "Quantity": 1
          }
        ]
      }
    ]
  }
}
```

Create

- Create new key-value pairs in objects or arrays
- Push new elements to array

Create Key in Object

Configuration

```
+||$. [?(@.OrderID==\'order103\')]||string||myValue||Test
```

Result

The search query returns the object of the **Order** collection, where the **OrderID='order103'**. For this object the key-value pair **Test:"myValue"** will be set if the key does not exist.

```
{
  "OrderID": "order103",
  "Product": [
    {
      "Product Name": "Bowler Hat",
      "ProductID": 858383,
      "Quantity": 2
    },
    {
      "Product Name": "Trilby hat",
      "ProductID": 858236
    }
  ],
  "Test": "myValue"
}
```

Create Key in Array

Configuration

```
+||$. [?(@.OrderID=="order103")].Product||string||myValue||Test
```

Result

The search query returns the all elements of the **Products** collection of the order where **OrderID='order103'**. This configuration sets the key-value pair **Test:"myValue"** in every element in the array.

```
{
  "OrderID": "order103",
  "Product": [
    {
      "Product Name": "Bowler Hat",
      "ProductID": 858383,
      "Quantity": 2,
      "Test": "myValue"
    },
    {
      "Product Name": "Trilby hat",
      "ProductID": 858236,
      "Test": "myValue"
    }
  ]
}
```

Insert Element in Array.

Configuration

```
+||$..[?(@.OrderID=="order103")].Product||json||"{\"Product Name\": \"Bowler  
Hat\", \"ProductID\": 858383, \"Quantity\": 2}"
```

Result

The search query returns all elements of the **Products** collection of the order where **OrderID='order103'** . If **key** is not defined, the value will be pushed into the array.


```
{
  "OrderID": "order103",
  "Product": [
    {
      "Product Name": "Bowler Hat",
      "ProductID": 858383,
      "Quantity": 2
    },
    {
      "Product Name": "Trilby hat",
      "ProductID": 858236
    },
    {
      "Product Name": "Bowler Hat",
      "ProductID": 858383,
      "Quantity": 2
    }
  ]
}
```

Update

- Change values of existing keys in objects or arrays

Update Key in Object

Configuration

```
&| |$.Account.Order[?(@.OrderID=="order104")].OrderID| |string| |myValue
```

or

```
&| |$.Account.Order[?(@.OrderID=="order104")]| |string| |myValue| |OrderID
```

Result

The search query returns the `OrderID` of the `Order` with `OrderID='order104'`. For this object the key `OrderID` will be set to `"myValue"`

```

{
  "OrderID": "myValue",
  "Product": [
    {
      "Product Name": "Bowler Hat",
      "ProductID": 858383,
      "Quantity": 4,
    },
    {
      "Product Name": "Cloak",
      "ProductID": 345664,
      "Quantity": 1,
    }
  ]
}

```

Update Key in Array

Configuration

```

&| |$. [?(@.OrderID=="order104")].Product | |number | |100 | |Quantity

```

Result

The search query returns the **Product** array of the **Order** with **OrderID='order103'** . For the elements in the array key **Quantity** will be set to **100** if it exists.

```

{
  "OrderID": "order104",
  "Product": [
    {
      "Product Name": "Bowler Hat",
      "ProductID": 858383,
      "Quantity": 100
    },
    {
      "ProductID": 345664,
      "Product Name": "Cloak",
      "Quantity": 100
    }
  ]
}

```

Update Element in Array

Configuration

```
&| |$. [?(@.OrderID=="order103")].Product[0] || json || 100 || {"myTest":{"x":1}}
```

Result

The search query returns first element of the **Product** array of the **Order** with **OrderID='order103'**. If the property **key** is not defined, the element will be replaced by the defined **value**.

```
{
  "OrderID": "order103",
  "Product": [
    {
      "myTest": {
        "x": 1,
      }
    },
    {
      "Product Name": "Trilby hat",
      "ProductID": 858236,
    }
  ]
}
```

Create or Update

- Combines the functionality of **Create** and **Update**
 - if a key does not exist it will be created
 - if a key exists it will be updated

Create or Update Key in Array

Configuration

```
?| |$. [?(@.OrderID=="order103")].Product | number | 100 | Quantity
```

Result

The search query returns the **Product** array of the **Order** with **OrderID='order103'**. For the elements in the array the key **Quantity** will be set to **100**.

```
{
  "OrderID": "order103",
  "Product": [
    {
      "Product Name": "Bowler Hat",
      "ProductID": 858383,
      "Quantity": 100
    },
    {
      "Product Name": "Trilby hat",
      "ProductID": 858236,
      "Quantity": 100
    }
  ]
}
```

Create or Update Element in Array

Configuration

```
?| |$.Account.Order[?(@.OrderID=="order103")].Product|json|[[{"id":"1","Product Name":"MyNewProduct XY","ProductID":858383}, {"id":"2","Product Name":"MyNewProduct Q","ProductID":858383}]
```

Result

The search query returns the **Product** array of the **Order** with **OrderID='order103'**. For the elements in the array the app checks if a product with the same **id** or **key** already exists and if so it will update the element in the array. Otherwise the element will be pushed to the array.

The sample JSON did not contain any **Products** with the an id, so the elements of the input array are pushed into the array.

```

{
  "OrderID": "order103",
  "Product": [
    {
      "Product Name": "Bowler Hat",
      "ProductID": 858383,
      "Quantity": 2
    },
    {
      "Product Name": "Trilby hat",
      "ProductID": 858236
    },
    {
      "id": "1",
      "Product Name": "MyNewProduct XY",
      "ProductID": 858383
    },
    {
      "id": "2",
      "Product Name": "MyNewProduct Q",
      "ProductID": 858383
    }
  ]
}

```

Delete

- Removes key-value pairs
- Removes elements from array

Remove Key from Object

Configuration

```

- | |$.Account.Order[?(@.OrderID=="order104")].OrderID"

```

Result

The search query returns the `OrderID` of the `Order` with `OrderID='order104'`. The key `OrderID` will be removed from the order object.

```
{
  "Product": [
    {
      "Product Name": "Bowler Hat",
      "ProductID": 858383,
      "Quantity": 4
    },
    {
      "Product Name": "Cloak",
      "ProductID": 345664,
      "Quantity": 1
    }
  ]
}
```

Remove Key from Array

Configuration

```
-| |$..[?(@.OrderID=="order104")].Product..Quantity
```

Result

The search query returns the **Quantity** of the **Product** array of the **Order** with **OrderID='order104'**. The key **Quantity** will be removed from every element in the array.

```
{
  "OrderID": "order104",
  "Product": [
    {
      "Product Name": "Bowler Hat",
      "ProductID": 858383
    },
    {
      "Product Name": "Cloak",
      "ProductID": 345664
    }
  ]
}
```

Remove Elements from Array

Configuration

```
-||$..[?(@.OrderID=="order104")].Product[0]"
```

Result

The search query returns the first element of the **Product** array of the **Order** with **OrderID='order104'**.

The element will be removed from the array.

```
{
  "OrderID": "order104",
  "Product": [
    {
      "Product Name": "Cloak",
      "ProductID": 345664
    }
  ]
}
```

Additional Datasets / Files + JSONata

The JSON Wizard also supports adding the content of any other JSON dataset to the master JSON.

This allows you to access the content with a JSONata or JSONPath expression.

After performing all actions, the content of the additional dataset or file is removed from the master JSON again.

Dataset

Content of dataset 'lookup'

```
[
  {
    "device": "inline",
    "gutterGap": 10,
    "height": 297,
    "nUp": 1,
    "orientation": "Portrait",
    "sheetSize": {
      "height": 320,
      "width": 460,
    },
    "width": 210,
  },
  {
    "device": "inline",
    "gutterGap": 0,
    "height": 90,
    "nUp": 1,
    "orientation": "Portrait",
    "sheetSize": {
      "height": 320,
      "width": 460,
    },
    "width": 60,
  },
]
```

JSON during runtime


```

{
  "lookup": [
    {
      "width": 210,
      "height": 297,
      "device": "inline",
      "nUp": 1,
      "sheetSize": {
        "width": 460,
        "height": 320
      },
      "orientation": "Portrait",
      "gutterGap": 10
    },
    {
      "width": 60,
      "height": 90,
      "device": "inline",
      "nUp": 1,
      "sheetSize": {
        "width": 460,
        "height": 320
      },
      "orientation": "Portrait",
      "gutterGap": 0
    }
  ]
}

```

Configuration

```

+||$||jsonata||lookup[height=297 and width=210 and device='inline']||impose
+||$||jsonata||$.impose.orientation = 'Portrait' ? $.impose.gutterGap : 0||columnGap
+||$||jsonata||$.impose.orientation = "Landscape" ? $.impose.gutterGap : 0||rowGap

```

Result

Resolved JSON

```

{
  "columnGap": 10,
  "rowGap": 0,
  "impose": {
    "device": "inline",
    "gutterGap": 10,
    "height": 297,
    "nUp": 1,
    "orientation": "Portrait",
    "sheetSize": {
      "height": 320,
      "width": 460,
    },
    "width": 210,
  }
}

```

Fallback

Allows to define a fallback query / value if the regular query does not return a value. The regular query and the fallback query must be separated by **##** as follows:

Syntax: `<regular query>##<fallback>`

In this case the property `orientation` should be 'Landscape' if the object has no such property. Otherwise if the property exists, the value should be unchanged

```

{
  "device": "inline",
  "gutterGap": 10,
  "height": 297,
  "nUp": 1,
  "sheetSize": {
    "height": 320,
    "width": 460
  },
  "width": 210
}

```

Configuration

JSONata:

```
?||$||jsonata|orientation##"Landscape"|orientation
```

JSONPath:

```
?||$||string|$.orientation##Landscape|orientation
```

Result

```
{
  "device": "inline",
  "gutterGap": 10,
  "height": 297,
  "nUp": 1,
  "sheetSize": {
    "height": 320,
    "width": 460
  },
  "width": 210,
  "orientation": "Landscape"
}
```

Access Current Element

Allows to access the element that is selected by the JSONPath expression. This element will be temporarily added to the JSON as `currentElement` making it available for JSONata or JSONPath queries.

In that case the property `secondProductHasQuantity` should be set to true or false. The JSONPath selects the order and the jsonata makes use of this result in it's query as `currentElement`

The `currentElement` is removed from the resulting JSON at the end of every query.

The input JSON is the example JSON from above.

Configuration

```
+||$.Account.Order[?
(@.OrderID=="order103")]||jsonata||$length($string(currentElement.Product[1].Quantity)) > 0 ? true
: false||secondProductHasQuantity
+||$.Account.Order[?
(@.OrderID=="order104")]||jsonata||$length($string(currentElement.Product[1].Quantity)) > 0 ? true
: false||secondProductHasQuantity
```

JSON during runtime

first query:

```
{
  "Account": {
    "Account Name": "Firefly",
    "Order": [...],
  },
  "currentElement": {
    "OrderID": "order103",
    "Product": [
      {
        "Product Name": "Bowler Hat",
        "ProductID": 858383,
        "Quantity": 2,
      },
      {
        "Product Name": "Trilby hat",
        "ProductID": 858236,
      },
    ]
  }
}
```

second query:

```
{
  "Account": {
    "Account Name": "Firefly",
    "Order": [...],
  },
  "currentElement": {
    "OrderID": "order104",
    "Product": [
      {
        "Product Name": "Bowler Hat",
        "ProductID": 858383,
        "Quantity": 4,
      },
      {
        "Product Name": "Cloak",
        "ProductID": 345664,
        "Quantity": 1,
      },
    ]
  }
}
```

Result

```

{
  "Account": {
    "Account Name": "Firefly",
    "Order": [
      {
        "OrderID": "order103",
        "Product": [
          {
            "Product Name": "Bowler Hat",
            "ProductID": 858383,
            "Quantity": 2,
          },
          {
            "Product Name": "Trilby hat",
            "ProductID": 858236,
          },
        ],
        "secondProductHasQuantity": false,
      },
      {
        "OrderID": "order104",
        "Product": [
          {
            "Product Name": "Bowler Hat",
            "ProductID": 858383,
            "Quantity": 4,
          },
          {
            "Product Name": "Cloak",
            "ProductID": 345664,
            "Quantity": 1,
          },
        ],
        "secondProductHasQuantity": true,
      },
    ],
  },
}

```

Error handling

This app uses two types of errors:

- job data/log error: If an action fails (e.g. no element found for JSONPath expression), the job and a log file will be sent to their connections. Additionally the error is logged in the switch messages.
- job fail: if the file format is not supported or for any other reason the job will fail and gets sent to the problem jobs folder. The thrown error gets logged as error and can be looked up in the switch messages.

Private data

The following private data tags will be set if an error occurs:

Tag	Value Type	Description
lastErrorElement	String	The name of the flow element
lastErrorId	jsonWizardError	
lastErrorCode	Number	an error code that defines the type of error that occurred
lastErrorMessage	String	detailed error message

Error Codes:

```
enum ERROR_CODES {
  generalError = 0,
  fileHandlingError = 1,
  fileFormatError = 2,
  conversionError = 3,
  invalidParameterValue = 4,
  parsingError = 5,
}
```