

# 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](#)
    - [Delete](#)
      - [Remove Key from Object](#)
      - [Remove Key from Array](#)
      - [Remove Elements from Array](#)
  - [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.: myOrderID=\$..[0].OrderID

## Working Mode Dependend Properties

### JSON-Dataset

Property	Value	Description
<i>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
<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	<b>Default:</b> 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
- 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 `$.`
- 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
    }
  ]
}
```

---

## 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
    }
  ]
}
```

## 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,  
}
```