Table of contents

JSON-Create

Description

With JSON-Create you can generate JSON files from a multiline field, an external JSON file or from an XML.

The XML conversion comes with several expert options to satisfy all your needs.

The mutliline and the external JSON support using JSONata or JSONPath expressions as value, which will be resolved by the app.

Compatibility

Switch Fall 2022 and higher.

Getting Started

Use one of our sample flows and drop a sample file into the flow.

Output Connections

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

- Log success: carries the created job if `Append Result as Dataset` is set to `false`
- Data error: carries the incoming job if the operation fails at the first attempt.
- Data success: carries the incoming job after the operation succeeds. If there are no data success connections the output is simply suppressed (with logging a warning).

Flow Element

General Properties

Property	Value	Description
Generate JSON From	enum [Multi-line | XML-File | XML-Dataset | External XML-File | External JSON-File]	Defines the source of the input file
Append Result As Dataset	Boolean	Defines a JSON (optionally with Switch variables)
Job Name	String	Defines the name proper of the created JSON
Dataset Name	String	Defines the dataset name of the JSON

Working Mode Dependending Properties

Multi-line

Property	Value	Description
JSON	String[]	Defines a JSON (optionally with Switch variables)
Additional datasets	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.
Additional file	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.
Object name	String	Defines the name of the appended object of the additional file.

External JSON-File

Property	Value	Description
File Path	String	Path to external XML file

Property	Value	Description
Delete after Injection	Boolean	Defines if the external file should be deleted or not after it is injected as Switch job
Additional datasets	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.
Additional file	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.
Object name	String	Defines the name of the appended object of the additional file.

XML

Property	Value	Description
Expert Settigns	Boolean	Enable or disable additional settings for the conversion from XML to JSON - click HERE further details

Defaults:

```
{
      preserveOrder: false,
      attributeNamePrefix: "",
      attributesGroupName: "$",
      textNodeName: "_",
      ignoreAttributes: false,
      removeNSPrefix: false,
      allowBooleanAttributes: false,
      parseTagValue: false,
      parseAttributeValue: false,
      trimValues: true,
      cdataPropName: false,
      stopNodes: [],
      alwaysCreateTextNode: false,
      commentPropName: false,
      unpairedTags: [],
      processEntities: true,
      htmlEntities: false,
      ignoreDeclaration: false,
      ignorePiTags: true
}
```

XML-Dataset

Property	Value	Description
Dataset Name	String	Name of the XML-Dataset that should be converted to JSON

External XML-File

Property	Value	Description
File Path	String	Path to external XML file
Delete after Injection	Boolean	Defines if the external file should be deleted or not after it is injected as Switch job

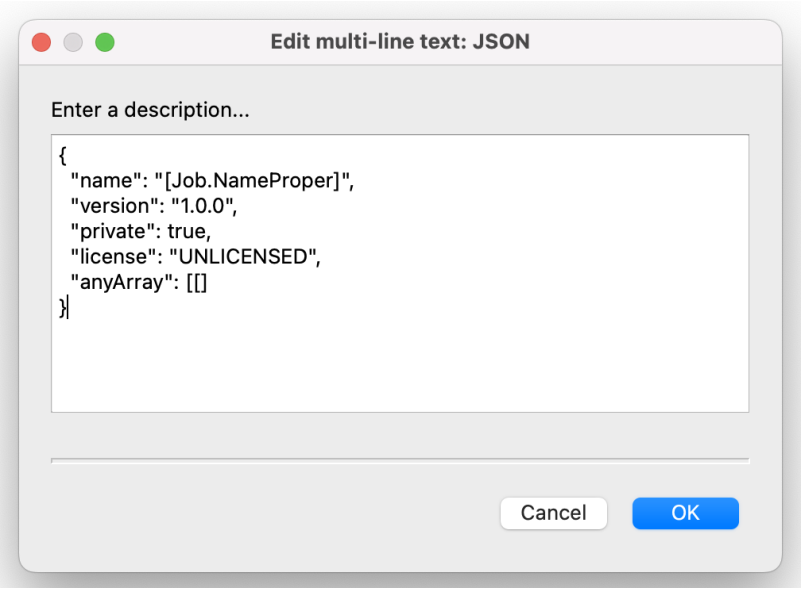
Examples

Multi-line to JSON

Input

Job-Name: `Test.pdf`

> ***NOTE:*** To the array `anyArray` an open bracket `[` must be added in order to resolve the switch variable



Result

The variable `[Job.NameProper]` was successfully resolved to `test`.

```
{
  "name": "test",
  "version": "1.0.0",
  "private": true,
  "license": "UNLICENSED",
  "anyArray": []
}
```

XML to JSON

Input

```xml
<?xml version="1.0" encoding="UTF-8"?>
<orders version="1.0">
  <order>
    <ID>1</ID>
    <name>Order1</name>
  </order>
  <order>
    <ID>2</ID>
    <name>Order1</name>
  </order>
</orders>
```

Result

```json
{
  "orders": {
    "order": [
      {
        "ID": "1",
        "name": "Order1"
      },
      {
        "ID": "2",
        "name": "Order1"
      }
    ],
    "$": {
      "version": "1.0"
    }
  }
}
```

Multi-line to JSON using JSONata and JSONPath

The JSONata expression must have the prefix "jsonata=" and the JSONPath expression must have "jsonpath=" as prefix. Notice that the whole string must be inside quotation marks (like below).

There is also a sample flow 'multiline additional dataset + file and jsonata + jsonpath' that can be used to reproduce this example.

Please notice to use the 'lookupDataset.json' as input file and to select 'lookupExternalFile.json' as external file.

The result is send to the log success connection.

Input

```
{
  "name": "Sample",
  "paper": "A4",
  "product": "Product1",
  "fromAdditionalFile": {
    "paperDetails": "jsonpath=$.lookupExternalFile[?(@.name == @root.paper)]"
  },
  "fromAdditionalDataset": {
    "webshop1": {
      "totalAmount": "jsonata=$sum(lookupDataset[source='webshop1'].amount)"
    },
    "webshop2": {
      "totalAmount": "jsonata=$sum(lookupDataset[source='webshop2'].amount)"
    }
  }
}
```

Dataset

The content of the dataset named 'lookupDataset' will be appended to the input JSON during runtime and is deleted after all queries are done.

Content:

```
[
  {
    "name": "Product2",
    "source": "webshop1",
    "amount": 100
  },
  {
    "name": "Product2",
    "source": "webshop1",
    "amount": 5
  },
  {
    "name": "Product1",
    "source": "webshop1",
    "amount": 1
  },
  {
    "name": "Product1",
    "source": "webshop2",
    "amount": 1
  },
  {
    "name": "Product1",
    "source": "webshop2",
    "amount": 10
  }
]
```


External JSON

The content of the external JSON will be appended to the input JSON during runtime and is deleted after all queries are done. The name of the object is defined by the `Name` property - In this case it is 'lookupExternalFile'

Content:

```
[
  {
    "name": "A4",
    "dimensions": {
      "width": 210,
      "height": 297
    }
  },
  {
    "name": "A3",
    "dimensions": {
      "width": 420,
      "height": 297
    }
  }
]
```

JSON during runtime

```
{
  "name": "Sample",
  "paper": "A4",
  "product": "Product1",
  "fromAdditionalFile": {
    "paperDetails": "jsonpath=$.lookupExternalFile[?(@.name == @root.paper)]"
  },
  "fromAdditionalDataset": {
    "webshop1": {
      "totalAmount": "jsonata=$sum(lookupDataset[source='webshop1'].amount)"
    },
    "webshop2": {
      "totalAmount": "jsonata=$sum(lookupDataset[source='webshop2'].amount)"
    }
  },
  "lookupExternalFile": [...],
  "lookupDataset": [...]
}
```

Result

Resolved JSON.

```
{
  "name": "Sample",
  "paper": "A4",
  "product": "Product1",
  "includes": [
    "additionalDataset",
    "additionalFile",
    "jsonata",
    "jsonpath"
  ],
  "fromAdditionalFile": {
    "paperDetails": {
      "name": "A4",
      "dimensions": {
        "width": 210,
        "height": 297
      }
    }
  },
  "fromAdditionalDataset": {
    "webshop1": {
      "totalAmount": 106
    },
    "webshop2": {
      "totalAmount": 11
    }
  }
}
```

Error handling

This app uses two types of errors:

- job data: if an handled error occures (e.g. wrong file format), the error message is logged in the switch messages.
- job fail: if any other error occurs, 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	jsonCreateError	

Tag	Value | Type	Description
lastErrorCode	Number	an error code that defines the type of error that occured
lastErrorMessage	String	detailed error message

Error Codes:

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