FLOWX.AI

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Documents plugin / Using the plugin / Generating docs based on templates

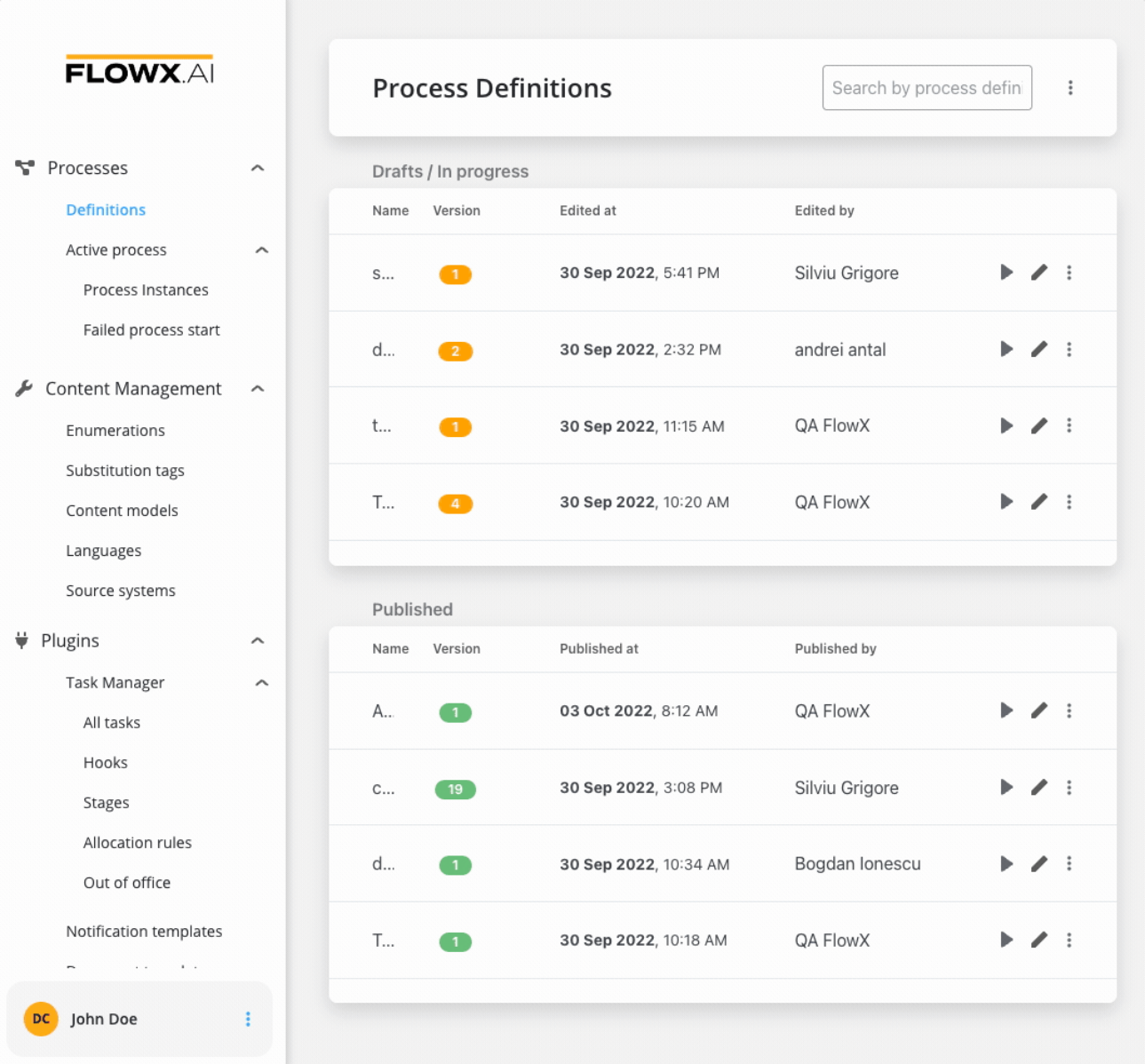
Contents

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Documents plugin / Using the plugin / Generating docs based on templates / Generating from HTML templates

The Document Management Plugin allows you to generate documents based on previously defined document templates. This example specifically covers generating documents using HTML templates.

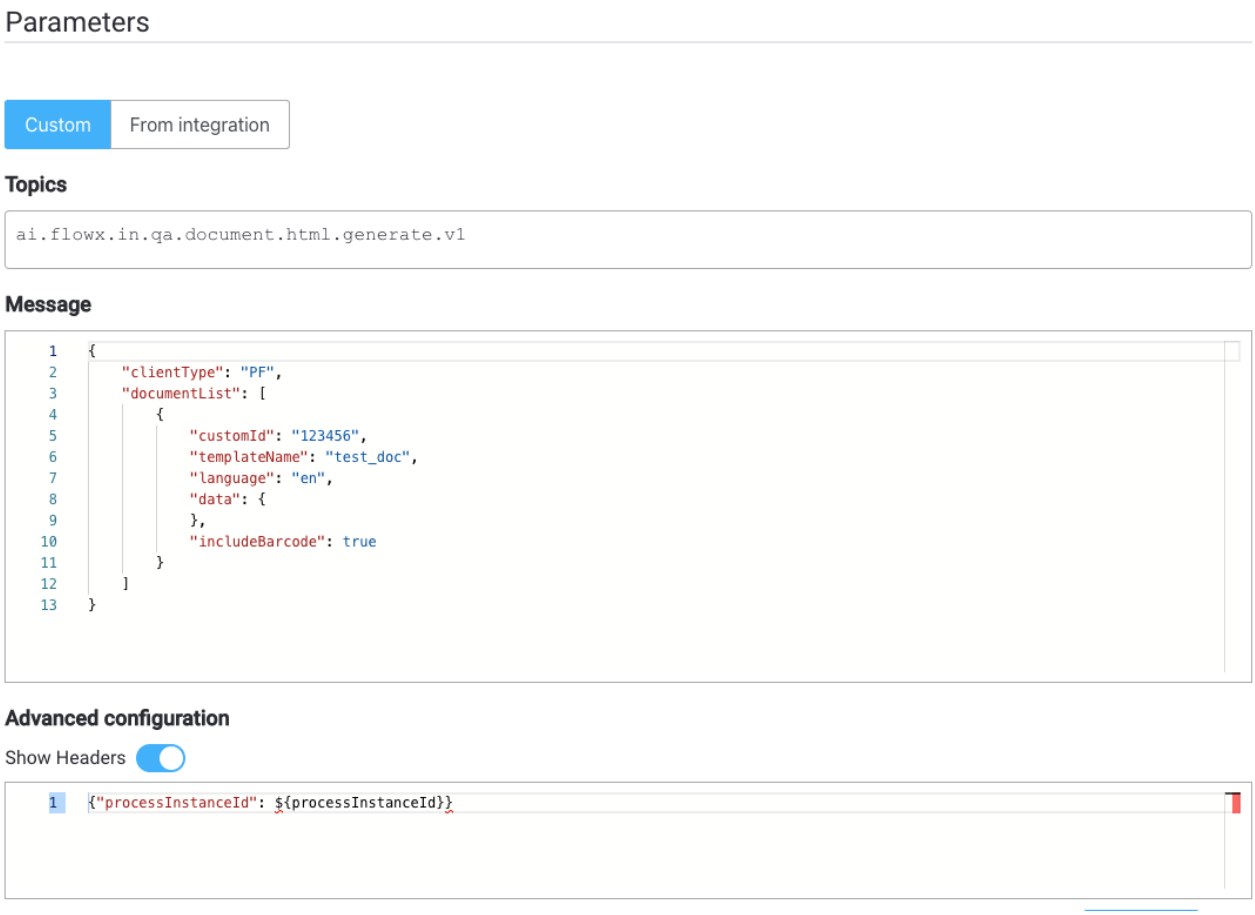
Creating a template

Use the WYSIWYG editor to create a document template.



Sending the request

1. Create a process that includes a **Kafka send event node** and a **Kafka receive event node** (one for sending the request and one for receiving the reply).
2. Configure the first node (Kafka Send Event) by adding a **Kafka send action**.
3. Add the **Kafka topic** to which the request should be sent.
4. Fill in the message with the following expected values in the request body:



- **documentList**: A list of documents to be generated with properties (name and value to be replaced in the document templates)

- **customId**: Client ID
- **templateName**: The name of the template to be used
- **language**
- **includeBarcode**: True/False
- **data**: A map containing the values that should be replaced in the document template. The keys used in the map should match the ones defined in the HTML template.

INFO

Kafka topic names can be set by using (overwriting) the following environment variables in the deployment:

- `KAFKA_TOPIC_DOCUMENT_GENERATE_HTML_IN` - default value: `ai.flowx.in.qa.document.html.generate.v1` - the topic that listens for the request from the engine
- `KAFKA_TOPIC_DOCUMENT_GENERATE_HTML_OUT` - default value: `ai.flowx.updates.qa.document.html.generate.v1` - the topic on which the engine expects the reply

The above examples of topics are extracted from an internal testing environment. When setting topics for other environments, follow the pattern `ai.flowx.updates.{{environment}}.document.generate.v1`.

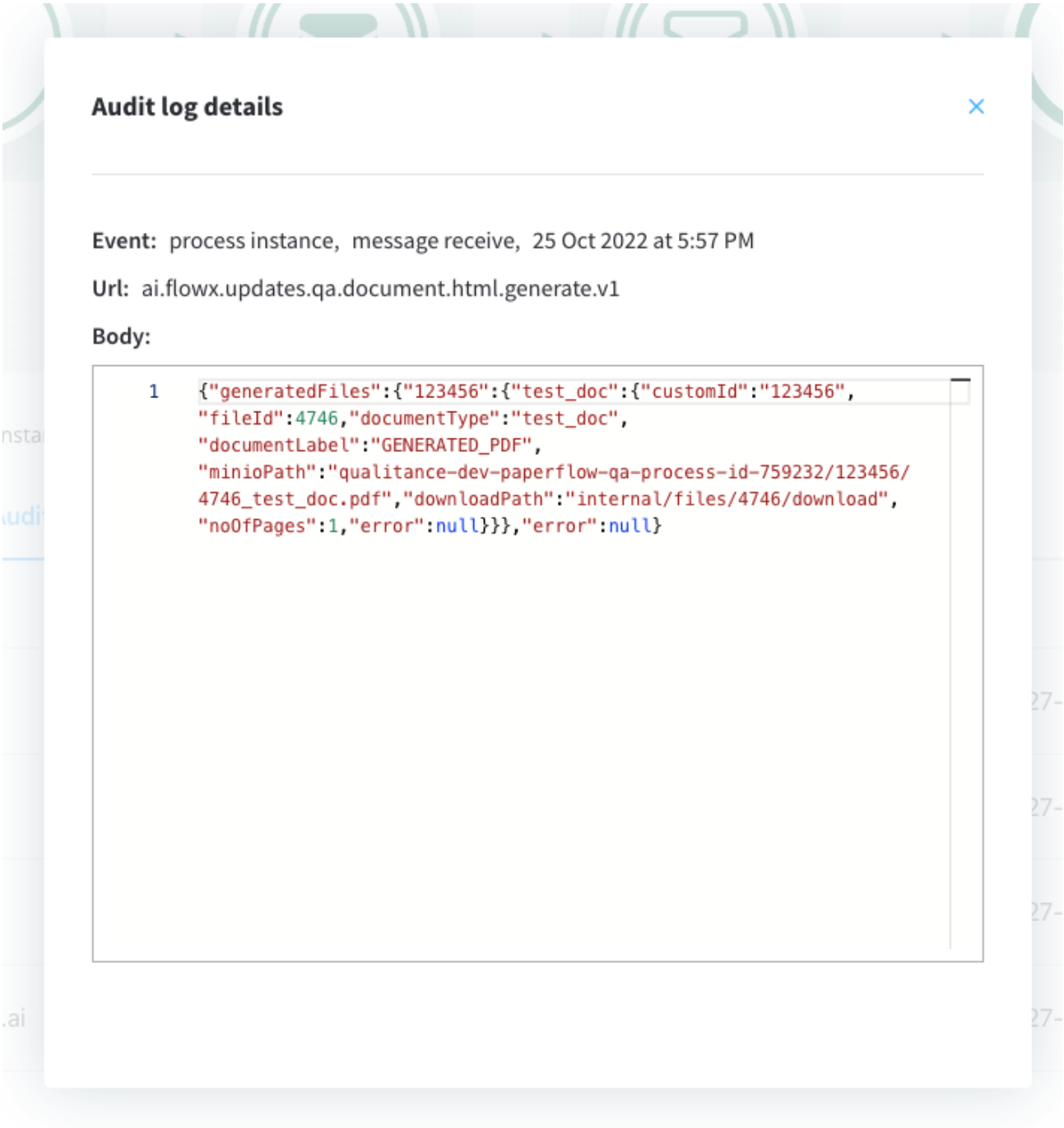
CAUTION

The engine listens for messages on topics with specific naming patterns. Make sure to use an outgoing topic name that matches the pattern configured in the engine.

Reply

> **INFO**
>
> You can view the response by accessing the **Audit log** menu.

The response will be sent on the output Kafka topic defined in the Kafka Receive Event Node. The response will contain the following information:



Values expected in the event body:

- **generatedFiles**: List of generated files.
  - **customId**: Client ID.

- **fileId**: The ID of the generated file.
- **documentType**: The name of the document template.
- **documentLabel**: A label or description for the document.
- **minioPath**: The path where the converted file is saved. It represents the location of the file in the storage system, whether it's a MinIO path or an S3 path, depending on the specific storage solution.
- **downloadPath**: The download path for the converted file. It specifies the location from where the file can be downloaded.
- **noOfPages**: The number of pages in the generated file.
- **error**: If there were any errors encountered during the generation process, they would be specified here. In the provided example, the value is null, indicating no errors.

Example of generated file response received on `KAFKA_TOPIC_DOCUMENT_GENERATE_HTML_IN` topic :

```
{
  "generatedFiles": {
    "123456": {
      "test_doc": {
        "customId": "123456",
        "fileId": 4746,
        "documentType": "test_doc",
        "documentLabel": "GENERATED_PDF",
        "minioPath": "qualitance-dev-paperflow-qa-process-
id-759232/123456/4746_test_doc.pdf", //or S3 path, depending
on your storage solution
        "downloadPath": "internal/files/4746/download",
        "noOfPages": 1,
        "error": null
```

```
        }
      }
    },
    "error": null
}
```


Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Documents plugin / Using the plugin / Generating docs based on templates / Managing HTML templates

In the Document Management Plugin, you have the flexibility to define and manage HTML templates for generating documents. These templates can incorporate various types of parameters to customize the content. Let's explore the different types of parameters and their specifications:

Configuring HTML templates

Text parameters

Text parameters are used to include dynamic text in the template. For example, you can include the company name and registration number in an offer document. Here's an example of HTML template specifications:

Lorem ipsum: Test Company SRL, dolor sit amet **RO1234567**.

```
<p><strong>Lorem ipsum: <span th:text="${companyName}">
</span></strong>, dolor sit amet <strong><span
th:text="${cui}"></span></strong>.</p>
```

Data specifications:

```
{
  "data": {
    "companyName": "Test Company SRL",
    "cui": "RO1234567"
  }
}
```

Dynamic tables - repeatable rows

Dynamic tables are useful when you want to display a table with repeatable rows. Each row can represent a different element from a generated list of objects. Here's an example of HTML template specifications:

Lorem ipsum The greatest offer - deluxe edition dolor sit amet, consectetur adipiscing elit. Nullam ante quam, dictum et accumsan quis, laoreet id lorem. Mauris bibendum consequat viverra. Ut accumsan volutpat augue. Cras id tortor hendrerit, fringilla ligula et, consequat quam. Proin quis dui et nisi ullamcorper pretium nec eu nulla. Sed ut sapien ac arcu accumsan varius. Proin faucibus augue tellus, at ultrices sapien vestibulum non. Nam pellentesque augue eu molestie sagittis.

Name	Value
Price (USD/MWh)*	25
Distribution rate (USD /MWh)**	C1 category: 27, C2 category: 29
Subscription price / day / place of consumption***	C1 category: 1.25, C2 category: 1.32
Period of validity of the price	Validity time fixed price Monday, from the start date of delivery to the date of completion of delivery
Payment term	90 days

```
<table>
    <thead>
        <tr class="headings">
            <th class="column-title">Name</th>
            <th class="column-title">Value</th>
        </tr>
    </thead>
    <tbody>
        <tr class='even pointer' th:each="row:
${offerValuesRows}" id="tablerow">
            <td th:each="header: ${offerValuesHeader}"
th:text="${row.get(header)}">
        </tr>
    </tbody>
</table>
```

Data specifications:

```
"data": {
   "offerValuesHeader": [
     "Name",
     "Value"
   ],
   "offerValuesRows": [
     { "Name": "Price (USD/MWh)", "Value": "25" },
     { "Name": "Distribution rate (USD/MWh)", "Value": "C1
category: 27, C2 category: 29" },
     { "Name": "Subscription price / day / place of
consumption", "Value": "C1 category: 1.25, C2 category:
1.32" },
     { "Name": "Period of validity of the price", "Value":
"Validity time fixed price Monday, from the start date of
delivery to the date of completion of delivery" },
     { "Name": "Payment term", "Value": "90 days" }
   ]
 }
```

Dynamic tables - repeatable table

This type of dynamic table allows you to display a table multiple times based on the elements of a generated list of objects. Here's an example of HTML template specifications:

Oferta Denumire oferta este aplicabila urmatoarelor locuri de consum:

Loc de consum	Distribuitor	Cod CLC	Modalitatate introducere consum	Tip consum	Categorie consum (MWh)	Consum total anual (MWh)
Lorem ipsum	Distribuitor 1	123456	Lorem ipsum kghf	Lorem ipsum	Lorem ipsum	Lorem ipsum

Loc de consum	Distribuitor	Cod CLC	Modalitatate introducere consum	Tip consum	Categorie consum (MWh)	Consum total anual (MWh)
Lorem ipsum	Distribuitor 2	131313	Lorem ipsum tryuty	Lorem ipsum	Lorem ipsum	Lorem ipsum

```
<p>Offer:</p>
<div th:each="type: ${consumptionPoints}">
<table>
    <thead>
       <tr>
          <th> Usage place </th>
          <th> Distributor </th>
          <th> CLC code </th>
          <th> Usage method input </th>
          <th> Usage type </th>
          <th> Usage category \n(MWh) </th>
          <th> Total usage \n(MWh) </th>
       </tr>
    </thead>
    <tbody>
       <tr th:if="${type.consumptionPoint.empty}">
           <td colspan=\"7\"> No information available here!
</td>
        </tr>
        <tr th:each=\"consumptionPoint :
${type.consumptionPoint}\=">
```

```
            <td><span
th:text="${consumptionPoint.consumptionPoint}"> Usage place
</span></td>
            <td><span
th:text="${consumptionPoint.distribuitor}"> Distributor
</span></td>
            <td><span th:text="${consumptionPoint.clcCode}">
Cod CLC </span></td>
            <td><span
th:text="${consumptionPoint.consumerInputMethod}"> Usage
method input </span></td>
            <td><span
th:text="${consumptionPoint.consumerType}"> Usage type
</span></td>
            <td><span
th:text="${consumptionPoint.consumerCategory}"> Usage
category \n(MWh) </span></td>
            <td><span
th:text="${consumptionPoint.totalAnnualConsumption}"> Total
usage \n(MWh) </span></td>
         </tr>
      </tbody>
   </table>
</div>
```

Data specifications:

```
  "data": {
    "consumptionPoints": [
      {
        "consumptionPoint": [
          {
            "consumptionPoint": "Lorem ipsum",
```

```
            "distribuitor": "Distributor 1",
            "clcCode": "123456",
            "consumerInputMethod": "Lorem ipsum",
            "consumerType": "Lorem ipsum",
            "consumerCategory": "Lorem ipsum",
            "totalAnnualConsumption": "Lorem ipsum"
          }
        ]
      },
      {
        "consumptionPoint": [
          {
            "consumptionPoint": "Lorem ipsum",
            "distribuitor": "Distributor 2",
            "clcCode": "131313",
            "consumerInputMethod": "Lorem ipsum ipsum",
            "consumerType": "Lorem ipsum",
            "consumerCategory": "Lorem ipsum",
            "totalAnnualConsumption": "Lorem ipsum"
          }
        ]
      }
    ]
  }
```


Dynamic sections

Dynamic sections allow you to display specific content based on certain conditions. For example, you can display a paragraph only when a certain condition is met. Here's an example of HTML template specifications:

PJ section, visible only if pjCLient = true

```
<span th:if="${pjCLient==true}">
    <p><b>PJ section, visible only if pjCLient = true</b>
</p>
    <p><span th:text="${termTechnicalServices}"></span></p>
</span>
<span th:if="${pjCLient==false}">
    <p><b>PF section, visible only if pjCLient = false</b>
</p>
    <p><span th:text="${termInsuranceServices}"></span></p>
</span>
```

Data specifications:

```
"data": {
   "pjCLient": true
 }
```

Images

You can include images in your final document by referencing them in the template. Here's an example of HTML template specifications:

Thank you,
LOREM IPSUM
John Smith
Administrator

Helen Smith
President

Test Company SRL, RO1234567
Hughes Michelle Sophie
Your function here,

```
<td class='align'><img th:src="*
{''+signature}" alt=\"\"
height='100px'/></td>
```

Data specifications:

```
"data": {
    "signature": "INSERT_BASE64_IMAGE"
  }
```

Barcodes

If you want to include a barcode, you can set the `includeBarcode` parameter to true.

For information on how to use barcodes and OCR, check the following section.

» OCR plugin

Lists

Lists are useful for displaying values from selected items in a checkbox as a bulleted list. Here's an example of HTML template specifications:

Income source:

- Income 1
- Income 2
- Income 3
- Income 4

```
  <div th:if="${incomeSource != null}">
    <h3>Income source:</h3>
    <ul>
       <li th:each="item : ${incomeSource}"
th:text="${item}"></li>
    </ul>
  </div>
```

Data specifications:

```
{
    "data": {
        "incomeSource": [
            "Income 1",
```

```
            "Income 2",
            "Income 3",
            "Income 4"
        ]
    }
}
```

Examples

> **TIP**
>
> Download a PDF sample generated based on the HTML example, **here**.

Was this page helpful?