PLATFORM DEEP DIVE / Core components / Core extensions / generic-parameters

Contents

PLATFORM DEEP DIVE / Core components / Core extensions / Generic parameters

Generic parameters are variables or settings that are used to control the behavior of a software application or system. These parameters are designed to be flexible and adaptable, allowing users to customize the software to their specific needs.

Through the

The fallback content to display on prerendering
, you can create, edit, import, or export these generic parameters. You can also assign the relevant environment(s) to these parameters, ensuring they are applied exactly where they are needed.

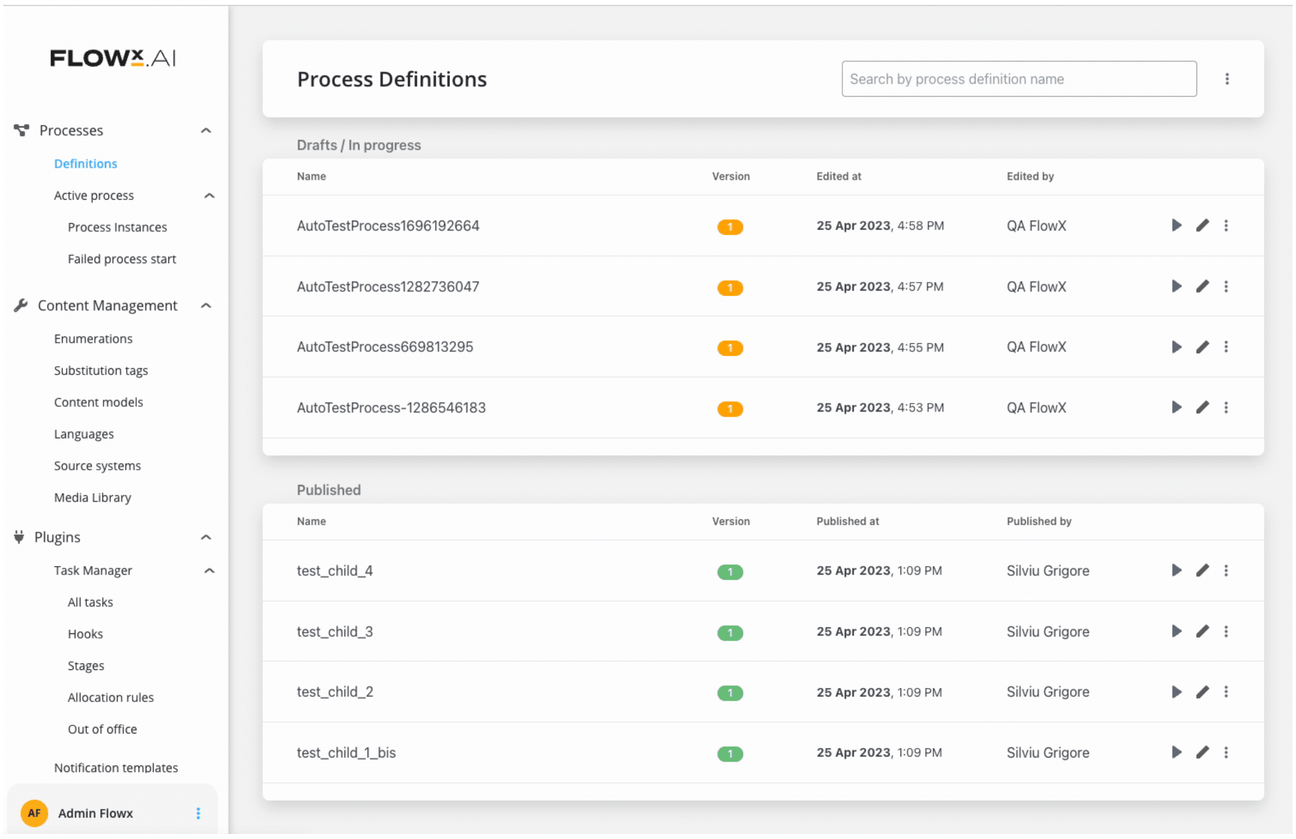
Why do you need generic parameters?

Generic parameters can be defined and used in many scenarios. Here are a few examples of useful generic parameters:

Parameter	Description
baseURL	This parameter can be used to define the base URL of an API or website, which can be utilized across multiple environments
redirectURL	This parameter can be used to define the URL to which a user should be redirected after completing a certain action or process. This can save time and effort by avoiding the need to hardcode multiple redirect URLs.
envFilePath	This parameter can be used to define the path of the environment file that stores a document uploaded

To add a new generic parameter, follow the next steps:

1. Go to **FLOWX Designer** and select the **General Settings** tab.
2. Select **Generic Parameters** from the list.
3. Click **New parameter**.
4. Fill in the details.
5. Click **Save**.



Process Definitions

Drafts / In progress

Name	Version	Edited at	Edited by
AutoTestProcess1696192664	1	25 Apr 2023, 4:58 PM	QA FlowX
AutoTestProcess1282736047	1	25 Apr 2023, 4:57 PM	QA FlowX
AutoTestProcess669813295	1	25 Apr 2023, 4:55 PM	QA FlowX
AutoTestProcess-1286546183	1	25 Apr 2023, 4:53 PM	QA FlowX

Published

Name	Version	Published at	Published by
test_child_4	1	25 Apr 2023, 1:09 PM	Silviu Grigore
test_child_3	1	25 Apr 2023, 1:09 PM	Silviu Grigore
test_child_2	1	25 Apr 2023, 1:09 PM	Silviu Grigore
test_child_1_bis	1	25 Apr 2023, 1:09 PM	Silviu Grigore

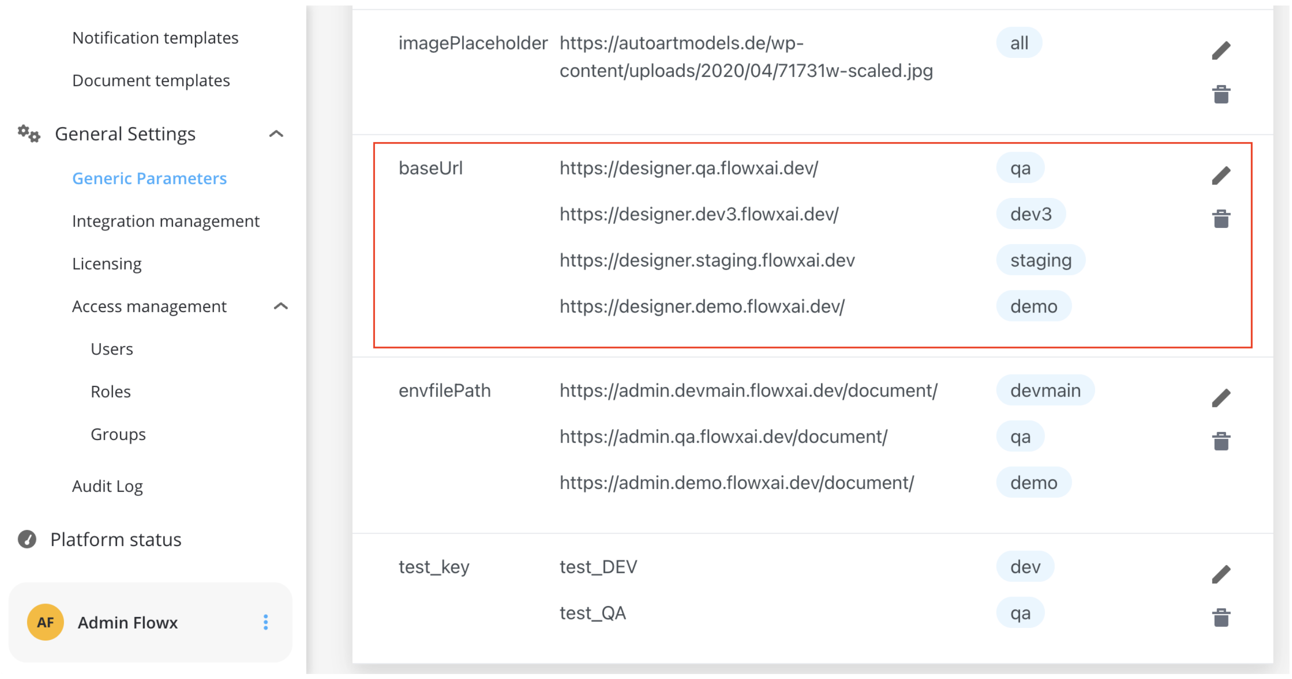
Configuring a generic parameter

To configure a generic parameter you need to fill in the following details:

- **Key** - the key that will be used in a process to call the generic parameter
- **Value** - the value that will replace the key depending on the defined parameters
- **Environment** - set the environment where you want to use the generic parameter (❗ leave empty to apply to all environments)
- **Add a new value** - to add a new value for the same key but for a different environment

INFO

For example, if you want to set a `baseURL` generic parameter (the URL will be different, depending on the environmaent).

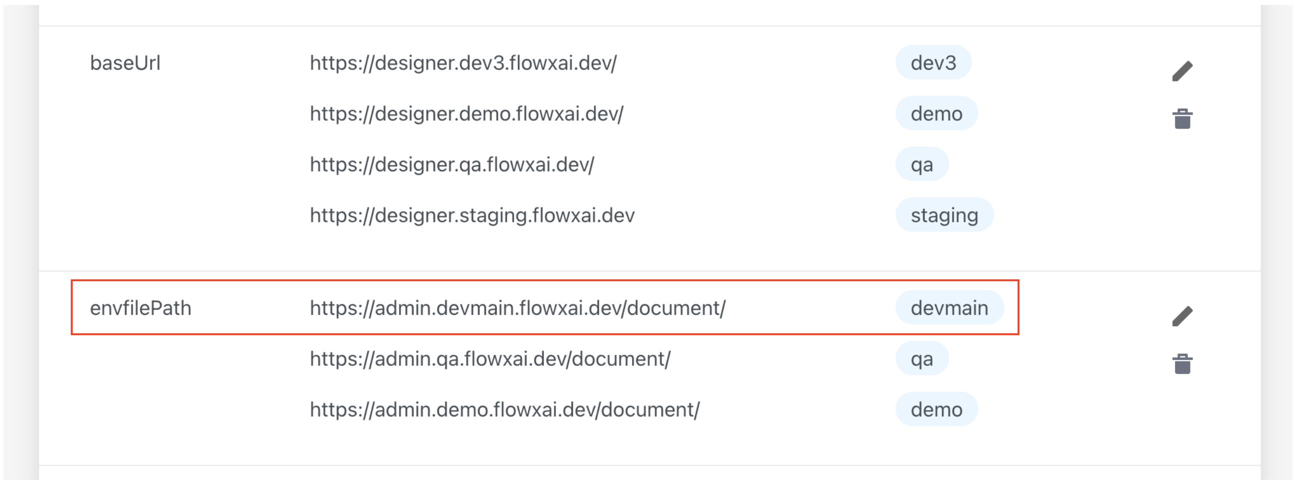


imagePlaceholder	https://autoartmodels.de/wp-content/uploads/2020/04/71731w-scaled.jpg	all
baseUrl	https://designer.qa.flowxai.dev/	qa
	https://designer.dev3.flowxai.dev/	dev3
	https://designer.staging.flowxai.dev	staging
	https://designer.demo.flowxai.dev/	demo
envfilePath	https://admin.devmain.flowxai.dev/document/	devmain
	https://admin.qa.flowxai.dev/document/	qa
	https://admin.demo.flowxai.dev/document/	demo
test_key	test_DEV	dev
	test_QA	qa

Using generic parameters

Use case

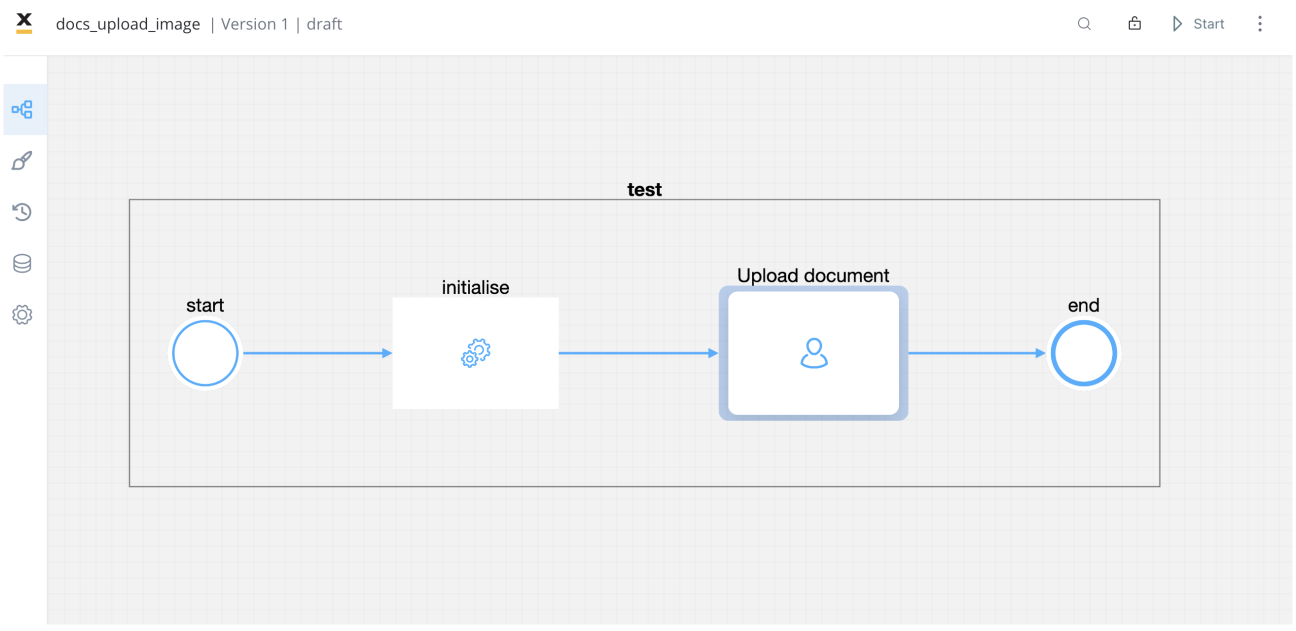
Imagine that you need to create a process in which you need to upload an image or a document. We will define a generic parameter called `envfilePath` that will represent the path where the document/image will be uploaded.



baseUrl	https://designer.dev3.flowxai.dev/	dev3
	https://designer.demo.flowxai.dev/	demo
	https://designer.qa.flowxai.dev/	qa
	https://designer.staging.flowxai.dev	staging
envfilePath	https://admin.devmain.flowxai.dev/document/	devmain
	https://admin.qa.flowxai.dev/document/	qa
	https://admin.demo.flowxai.dev/document/	demo

The minimum requirement to build an upload doc/image process:

- a start node
- a task node
- a user task node
- an end node



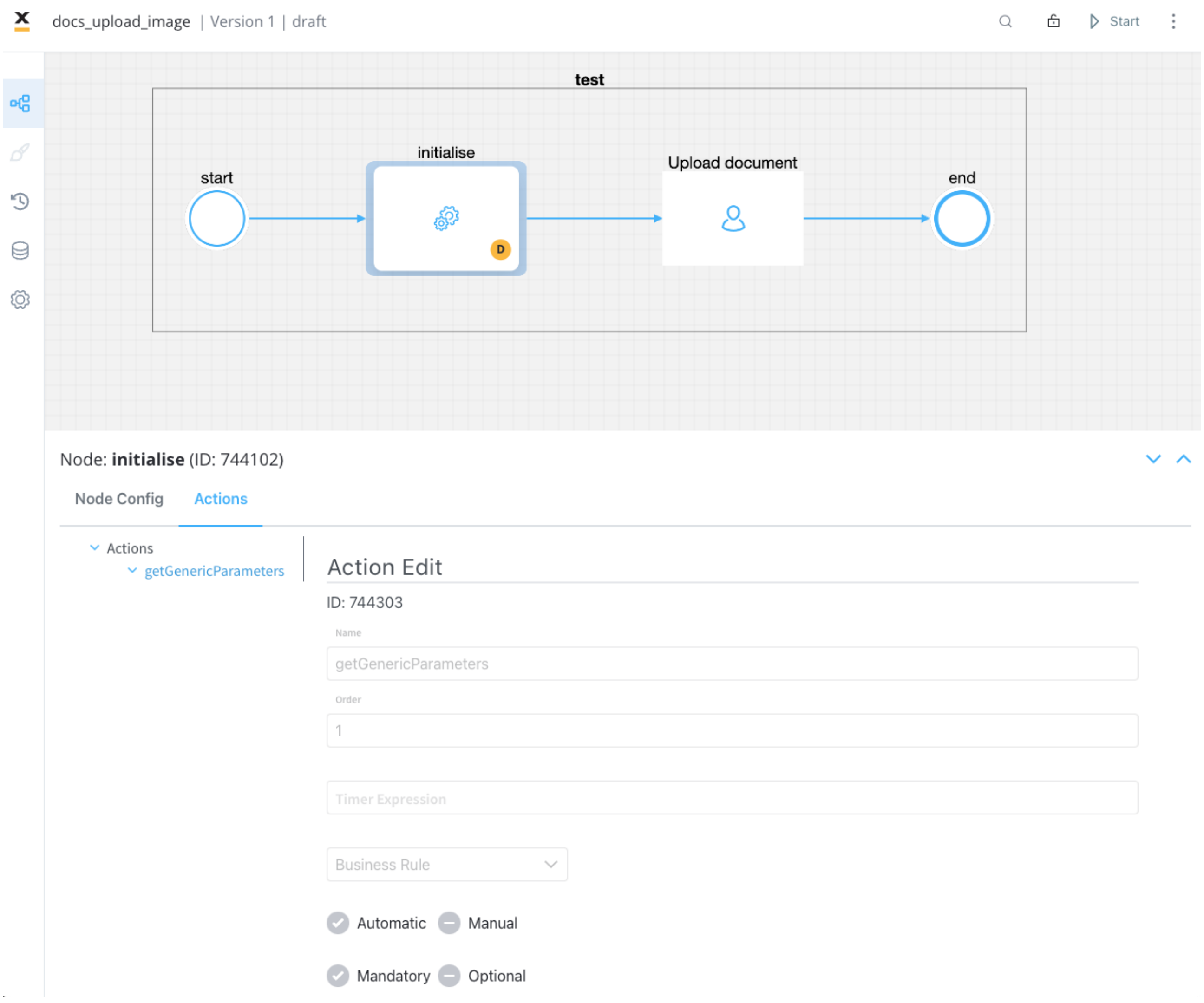
Configuring the task node

Set a

The fallback content to display on prerendering
action on the task node with the following properties:

Action Edit

- **Name** - used internally to make a distinction between different actions on nodes in the process - example *getGenericParameters*
- **Order** - if multiple actions are defined on the same node, the running order should be set using this option
- **Action type** - should be set to **Business Rule**
- **Trigger type** - Automatic - choose if this action should be triggered automatically (when the process flow reaches this step)
- **Required type** - automatic actions can only be defined as mandatory



Action Edit

ID: 744303

Name: getGenericParameters

Order: 1

Timer Expression

Business Rule

Automatic [✓] Manual

Mandatory [✓] Optional

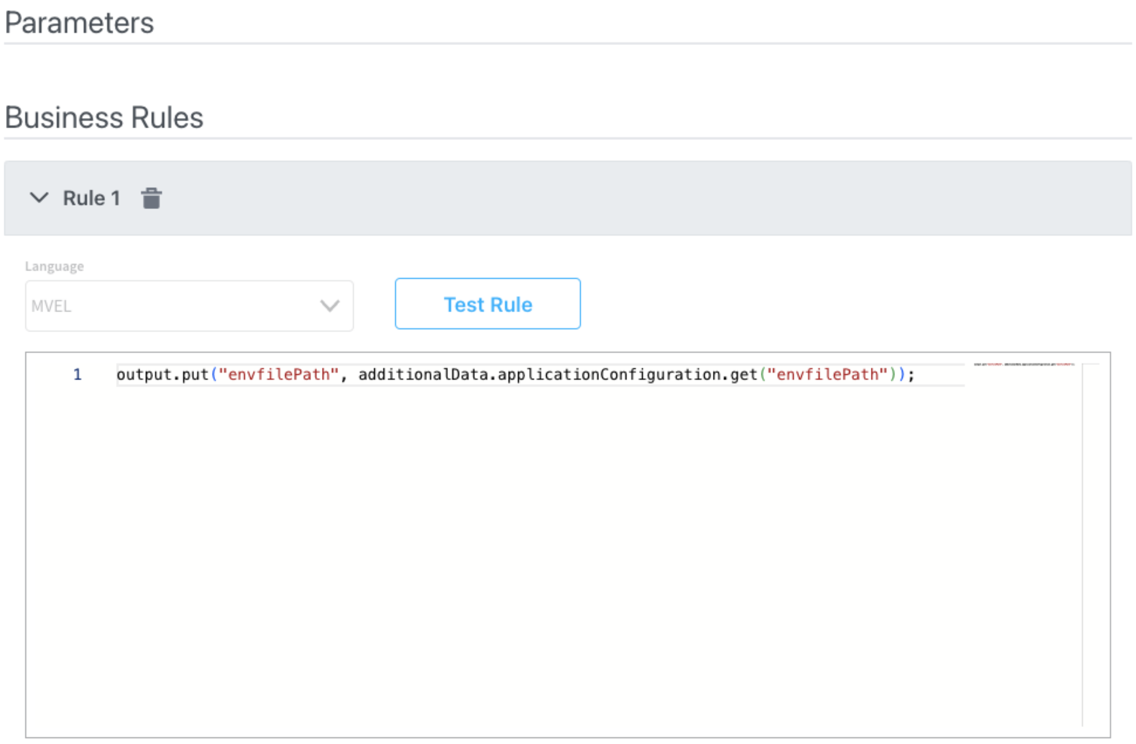
Parameters

- **Language** - we will choose **MVEL** for this business rule example

» MVEL details here

- **Message body** - MVEL expression

```
output.put("envfilePath",
additionalData.applicationConfiguration.get("envfilePath"));
```



```
output.put("envfilePath", additionalData.applicationConfiguration.get("envfilePath"));
```

This MVEL business rule assigns a value to a key, `envFilePath` (our defined generic parameter) in the "output" map object. The value assigned to the key is retrieved from another object, `additionalData.applicationConfiguration`, using the "get" method and passing the key `envFilePath` as the parameter.

In other words, this rule extracts the value of the `envFilePath` generic parameter from the `additionalData.applicationConfiguration` object and assigns it to the `envFilePath` key in the "output" map object.

It is important to note that the `additionalData.applicationConfiguration` object and the "output" map object must be previously defined and accessible in the current context for this rule to work.

Configuring the user task node

On this node we will define the following:

- an Upload File action with two child actions:
  - a Business Rule
  - a Send data to user interface action

INFO

Child actions can be marked as callbacks to be run after a reply from an external system is received. They will need to be set when defining the interaction with the external system (the Kafka send action).

For example, a callback function might be used to handle a user's interaction with a web page, such as upload a file. When the user performs the action, the callback function is executed, allowing the web application to respond appropriately.

Configuring Upload file action

Set an Upload file action on the task node with the following properties:

Action Edit

- **Name** - *uploadDocument*

- **Order** - if multiple actions are defined on the same node, the running order should be set using this option
- **Action type** - should be set to **Upload File**
- **Trigger type** - manually (triggered by the user)
- **Required type** - optional
- Repeatable - yes - should be checked if the action can be triggered multiple times
- **Autorun Children** - when this is switched on, the child actions (the ones defined as mandatory and automatic) will run immediately after the execution of the parent action is finalized

Parameters

- **Address** - the Kafka topic where the file will be posted - `ai.flowx.in.devmain.document.persist.v1`

CAUTION

In this example we used an environment called `devmain`, topic naming convention is different depending on what environment you are working.

- **Document Type** - other metadata that can be set (useful for the document plugin) - example: `BULK`
- **Folder** - allows you to configure a value by which the file will be identified in the future - example: `1234_${processInstanceId}`
- **Advanced configuration (Show headers)** - this represents a JSON value that will be sent on the headers of the Kafka message

```
{"processInstanceId": ${processInstanceId}, "destinationId":
"Upload document", "callbacksForAction": "uploadDocument"}
```

- `callbacksForAction` - the value of this key is a string that specifies a callback action associated with the `Upload document` destination ID (node). This is part of an event-driven system (Kafka send action) where this callback will be called once the `uploadDocument` action is completed.

Configuring Business rule action

```
envfilePath = input.?envfilePath;
uploadedDocument = input.?uploadedDocument;
if(uploadedDocument.?downloadPath != null &&
uploadedDocument.?downloadPath != ""){
    filePath = envfilePath + uploadedDocument.?downloadPath;
    uploadedDocument.filePath = filePath;

    output.put("uploadedDocument", uploadedDocument);
}
```

1. The business rule is expecting two inputs: `envfilePath` and `uploadedDocument`.
2. It is checking if the `downloadPath` property of the `uploadedDocument` input is not null and not an empty string. If it's not, then it proceeds to the next steps.
3. It concatenates the `envfilePath` and the `downloadPath` to form the full file path (filePath) where the uploaded document is expected to be located.
4. It updates the `uploadedDocument` input by adding a new property called `filePath` with the value of `filePath`.

5. It puts the updated `uploadedDocument` object into the output object as a key-value pair, with the key being "uploadedDocument".

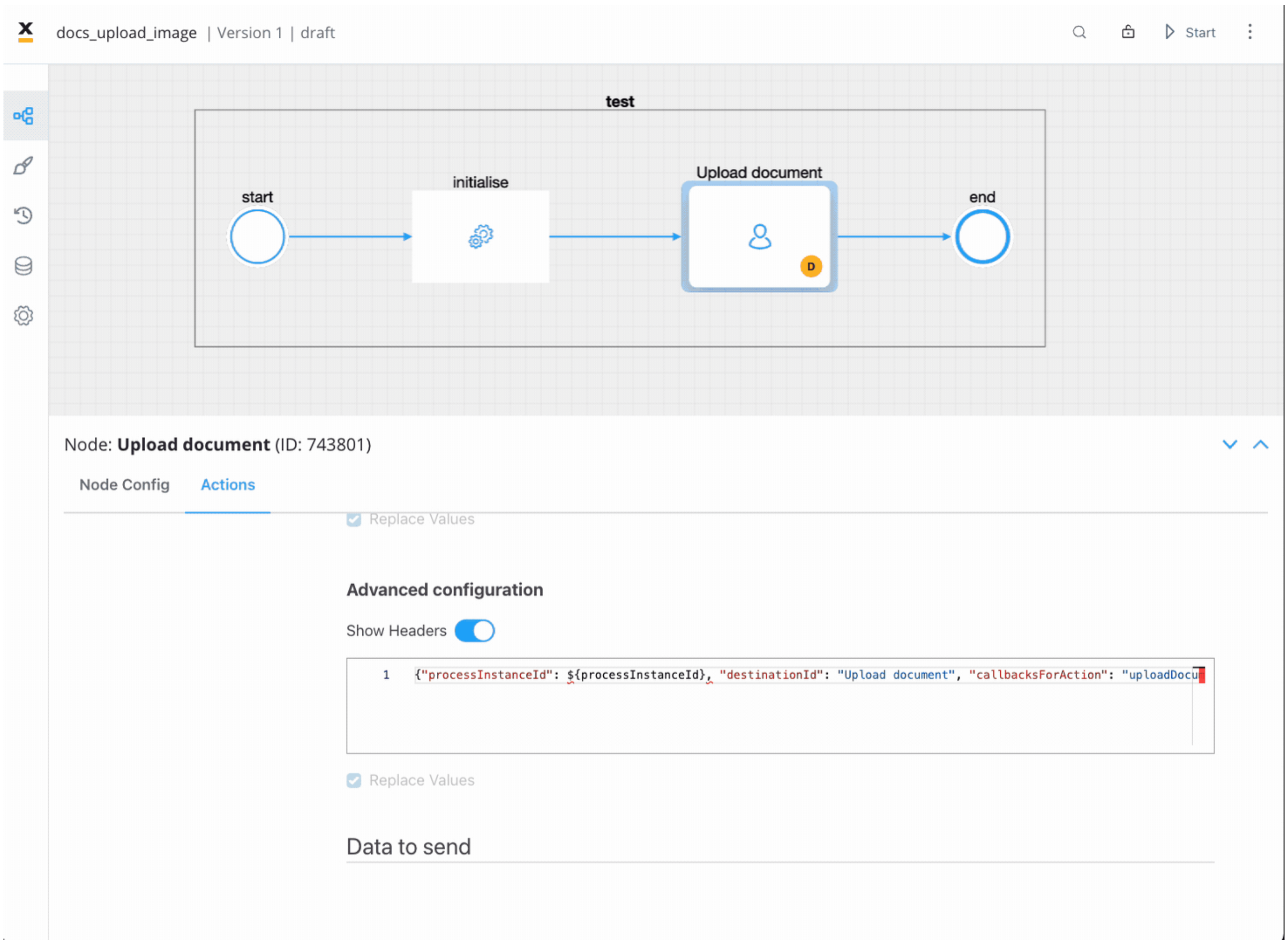
In summary, the code seems to be processing an uploaded document by checking its download path, constructing a full file path, and updating the document object with the new file path. Finally, it outputs the updated document object.

Configuring a Send data to user interface action

```
{"uploadedDocument":
    {
        "filePath": "${uploadedDocument.filePath}"
    }
}
```

"filePath": This is a key in the object which holds the value ${uploadedDocument.filePath}. The syntax ${...} suggests that it's a variable placeholder that will be replaced with the actual value at runtime.

After configuring all the nodes and parameters, run the process:



Node: **Upload document** (ID: 743801)

Node Config | Actions

Replace Values

Advanced configuration

Show Headers

```
{"processInstanceId": ${processInstanceId}, "destinationId": "Upload document", "callbacksForAction": "uploadDocu
```

Replace Values

Data to send

Was this page helpful?