



PLATFORM DEEP DIVE

Contents

- PLATFORM DEEP DIVE / Core components / FLOWX.AI Engine
 - A high-level overview
 - Orchestration
 - REST API
 - Triggering or skipping nodes on a process based on Flow Names
- PLATFORM DEEP DIVE / Core components / Advancing controller
 - Usage
 - Configuration
- PLATFORM DEEP DIVE / Core components / Events gateway
- PLATFORM DEEP DIVE / Core components / Core extensions / Content management / Using the service
 - Define needed Kafka topics
 - Example: Request a label by language or source system code
- PLATFORM DEEP DIVE / Core components / Core extensions / Content management / Enumerations
- PLATFORM DEEP DIVE / Core components / Core extensions / Content management / Substitution tags
- PLATFORM DEEP DIVE / Core components / Core extensions / Content management / Content models
- PLATFORM DEEP DIVE / Core components / Core extensions / Content management / Languages
- PLATFORM DEEP DIVE / Core components / Core extensions / Content management / Source systems
- PLATFORM DEEP DIVE / Core components / Core extensions / Content management / Media library
 - Displaying assets
 - Searching assets

- Replacing assets
- Referencing assets in UI Designer
- Icons
 - Customization
- Export/import media assets
 - Import media assets
 - Export all
- PLATFORM DEEP DIVE / Core components / Core extensions / Generic parameters
 - Configuring a generic parameter
 - Using generic parameters
 - Use case
 - Configuring the task node
 - Configuring the user task node
- PLATFORM DEEP DIVE / Core components / Core extensions / Integration management / Configuring access rights for Integration Management
- PLATFORM DEEP DIVE / Core components / Core extensions / Licensing
- PLATFORM DEEP DIVE / Core components / Core extensions / Audit log
 - Filtering
 - Audit log details
- PLATFORM DEEP DIVE / Core components / Core extensions / Scheduler
 - Overview
 - Using the scheduler
- PLATFORM DEEP DIVE / Core components / Core extensions / Search data service
 - Using search data
- PLATFORM DEEP DIVE / Core components / Renderer SDKs / Using the Angular Renderer
 - Angular project requirements

- Installing the library
- Using the library
 - Authorization
 - Development
 - Running the tests
- PLATFORM DEEP DIVE / Core components / Renderer SDKs / Using the iOS Renderer
 - iOS Project Requirements
 - Installing the library
 - Swift Package Manager
 - Cocoapods
 - Library dependencies
 - Configuring the library
 - FXConfig
 - FXSessionConfig
 - Using the library
 - How to start and end FlowX session
 - How to start a process
 - How to resume a process
 - How to end a process
 - How to run an action from a custom component
 - How to run an upload action from a custom component
 - Getting a substitution tag value by key
 - Getting a media item url by key
 - Handling authorization token changes
 - FXDataSource
- PLATFORM DEEP DIVE / Core components / Renderer SDKs / Using the Android Renderer
 - Android project requirements

- Installing the library
 - Library dependencies
- Accessing the documentation
- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Documents plugin /
Using the plugin / Generating docs based on templates / Generating from
HTML templates
 - Creating a template
 - Sending the request
 - Reply
- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Documents plugin /
Using the plugin / Generating docs based on templates / Managing HTML
templates
 - Configuring HTML templates
 - Text parameters
 - Dynamic tables - repeatable rows
 - Dynamic tables - repeatable table
 - Dynamic sections
 - Images
 - Barcodes
 - Lists
 - Examples
- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Documents plugin /
Using the plugin / Uploading a new document
 - Defining the process
 - Configuring the process definition
 - User task node
 - Milestone nodes
 - Receiving the reply

- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Documents plugin / Using the plugin / Converting documents to different formats
 - Sending the request
 - Receiving the reply
- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Documents plugin / Using the plugin / Splitting a document
 - Sending the request
 - Receiving the reply
- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Documents plugin / Using the plugin / Updating and deleting document files
 - Updating files
 - Sending the request
 - Receiving the reply
 - Deleting files from a document
 - Sending the request
 - Receiving the reply
- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Documents plugin / Using the plugin / Getting URLs for documents
 - Sending the request
 - Receiving the reply
- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Documents plugin / Using the plugin / Listing stored files
 - REST API
 - List buckets
 - List Objects in a Bucket
 - Download File
- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Notifications plugin / Using the plugin / Managing notification templates

- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Notifications plugin / Using the plugin / Sending a notification
 - Configuring the process
 - Define needed Kafka topics
 - Example: send a notification from a business flow
- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Notifications plugin / Using the plugin / Sending an email with attachments
 - Defining process actions
 - Example: send an email notification with attached files from a business flow
- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Notifications plugin / Using the plugin / Forward notifications to an external system
- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Notifications plugin / Using the plugin / OTP flow / Generate OTP
 - Define needed Kafka topics
 - Request to generate an OTP
 - Response from generate OTP
 - Example: generate an OTP from a business flow
- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Notifications plugin / Using the plugin / OTP flow / Validate OTP
 - Define needed Kafka topics
 - Request to validate an OTP
 - Reply from validate OTP
 - Example: validate an OTP from a business flow
- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Task management / Using allocation rules
- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Task management / Using hooks

- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Task management / Using out of office records
- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Task management / Using stages
- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Customer management / Using the customer management plugin
 - Kafka topics for customer management
 - Key examples
 - Keys description
- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Customer management / Customer management plugin example
 - Integrate a customer search in a business flow
- PLATFORM DEEP DIVE / Plugins / Custom Plugins / OCR plugin
 - Using the OCR plugin
 - Use case
 - Scenario for FLOWX.AI generated documents
 - Setup guide
- PLATFORM DEEP DIVE / Plugins / Custom Plugins / Reporting / Authorization & access roles
 - IAM solution
 - Prerequisites
- PLATFORM DEEP DIVE / Plugins / Plugins setup guides / Customer management plugin setup
 - Infrastructure Prerequisites:
 - Elastic Search
 - Postgres database
 - Configuration
 - Authorization configuration
 - Datasource configuration

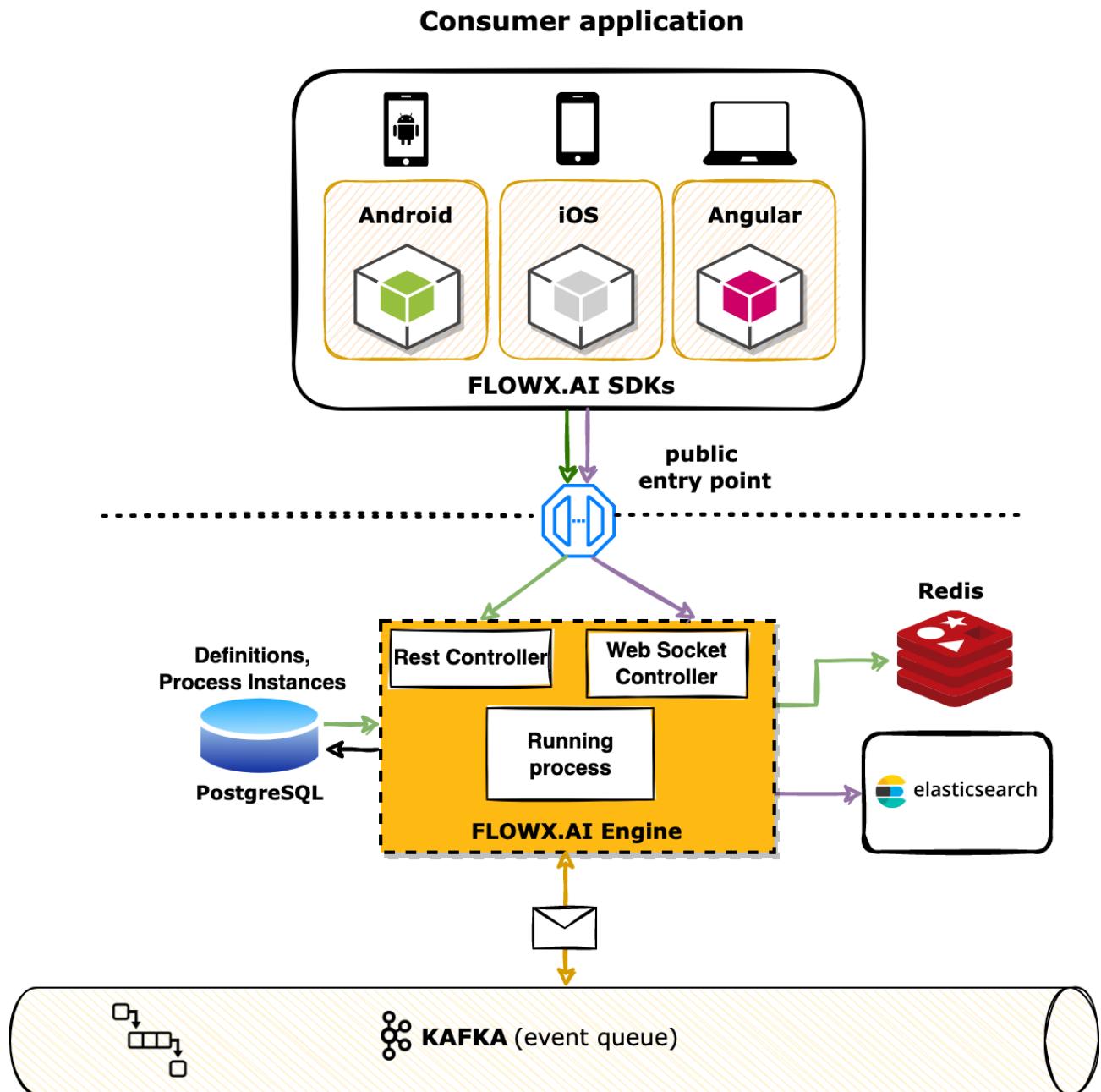
- Elastic search configuration
- Kafka configuration
- Logging
- PLATFORM DEEP DIVE / Plugins / Plugins setup guides / Documents plugin setup / Configuring access rights for Documents
- PLATFORM DEEP DIVE / Plugins / Plugins setup guides / Notification templates plugin setup / Configuring access rights for Notifications
- PLATFORM DEEP DIVE / Plugins / Plugins setup guides / OCR plugin setup
 - Infrastructure Prerequisites:
 - Deployment/Configuration
 - Credentials
 - Kafka configuration
 - Authorization
 - Storage (S3 configuration)
 - Performance
 - Certificates
 - Workers behavior
- PLATFORM DEEP DIVE / Plugins / Plugins setup guides / Reporting setup guide
 - Dependencies
 - Postgres database
 - Reporting plugin helm chart (containing CRON)
 - Superset
 - After installation
 - Datasource configuration
 - Redis configuration
 - Keycloak configuration
 - Extend the Security Manager
 - Configure Superset

- PLATFORM DEEP DIVE / Plugins / Plugins setup guides / Task Manager
 - plugin setup / Configuring access rights for Task management
- PLATFORM DEEP DIVE / Plugins / WYSIWYG editor
- PLATFORM DEEP DIVE / Integrations / Creating a Kafka consumer
 - Required dependencies
 - Configuration
 - Code sample for a Kafka Listener
- PLATFORM DEEP DIVE / Integrations / Creating a Kafka producer
 - Required dependencies
 - Configuration
 - Code sample for a Kafka producer
- PLATFORM DEEP DIVE / Integrations / Jaeger setup for microservices
 - Required dependencies
 - Needed configs
 - Add Kafka interceptors for Tracing
 - Extract Jaeger span context from received Kafka message
 - Send span context with outgoing Kafka messages
- PLATFORM DEEP DIVE / Integrations / Mock integrations
 - Setup
 - Adding a new integration
- PLATFORM DEEP DIVE / Third-party components
- PLATFORM DEEP DIVE / User roles management / Swimlanes
- PLATFORM DEEP DIVE / User roles management / Business filters

PLATFORM DEEP DIVE / Core components / FLOWX.AI Engine

The engine is the core of the platform, it is the service that runs instances of the process definitions, generates UI, communicates with the frontend and also with custom integrations and plugins. It keeps track of all currently running process instances and makes sure the process flows run correctly.

A high-level overview



Orchestration

Creating and interacting with process instances is pretty straightforward, as most of the interaction happens automatically and is handled by the engine.

The only points that need user interaction are starting the process and executing user tasks on it (for example when a user fills in a form on the screen and saves the results).

REST API

The process can be started by making an API call. Certain parameters needed by the process can be sent on the request body.

Some special cases for starting process instances are:

- starting a process instance from another instance and inheriting some data from the first one to the second
- a process can have multiple start nodes, in which case, a start condition must be set when making the start process call

▶ **POST**

ENGINE_URL/api/process/PROCESS_DEFINITION_NAME/start

▶ **POST**

**ENGINE_URL/api/process/PROCESS_DEFINITION_NAME/start/inherit
From/RELATED_PROCESS_INSTANCE_UUID**

The `paramsToInherit` map should hold the needed values on one of the following keys, depending on the desired outcome:

- `paramsToCopy` - this is used to pick only a subset of parameters to be inherited from the parent process; it holds the list of key names that will be

inherited from the parent parameters

- `withoutParams` - this is used in case we need to remove some parameter values from the parent process before inheriting them; it holds the list of key names that will be removed from the parent parameters

If none of these keys have values, all the parameter values from the parent process will be inherited by the new process.

▶ **GET** `ENGINE_URL/api/process/PROCESS_INSTANCE_ID/status`

▶ **POST**
`ENGINE_URL/api/process/PROCESS_INSTANCE_ID/token/TOKEN_IN
STANCE_ID/action/ACTION_NAME/execute`

▶ **POST**
`ENGINE_URL/api/process/PROCESS_INSTANCE_ID/token/TOKEN_IN
STANCE_ID/action/ACTION_NAME/upload`

» [FLOWX.AI Engine setup guide](#)

Triggering or skipping nodes on a process based on Flow Names

There might be cases when you want to include or exclude process nodes based on some information that is available at start. For example, in case of a bank onboarding process, you might want a few extra nodes in the process in case the person trying to onboard is a minor.

For these cases, we have added the possibility of defining flow names. For each process node, we can choose to set if they are only available in certain cases. In the example above, we will need a flow name, let's call it `enroll_minor`. And we'll have to add this to the extra nodes.

When starting a process, in case a value is set on the `flowName` key in the values map, it will be used in order to decide which nodes to be included in a certain process instance. This means that if an adult starts the bank onboarding process, no extra key will be added when starting the process, so the extra nodes that have the `enroll_minor` flow name set will not be included. Those nodes will only be included in case a minor person starts an onboarding process.

INFO

If no `flowName` value is set on a node, this means the node will be included in all possible flows.

A node could also be a part of multiple flow names.

Was this page helpful?

PLATFORM DEEP DIVE / Core components / Advancing controller

The Advancing Controller is a support service for the [Process Engine](#) that enhances the efficiency of advancing operations. It facilitates equal distribution and redistribution of the workload during scale-up and scale-down scenarios.

To achieve its functionality, the Advancing Controller microservice utilizes Postgres triggers in the database configuration.

(!) INFO

A Postgres trigger is a function that is automatically executed whenever specific events, such as inserts, updates, or deletions, occur in the database.

Usage

The Advancing Controller Service is responsible for managing and optimizing the advancement process in the PostgreSQL/OracleDB databases. It ensures efficient workload distribution, performs cleanup tasks, and monitors the status of worker pods. If a worker pod fails, the service reassigns its work to other pods to prevent process instances from getting stuck. It is essential to have the Advancing Controller Service running alongside the Process Engine for uninterrupted instance advancement.

(!) INFO

It is important to ensure that both the Process Engine and the Advancing Controller microservice are up and running concurrently for optimal performance.

Configuration

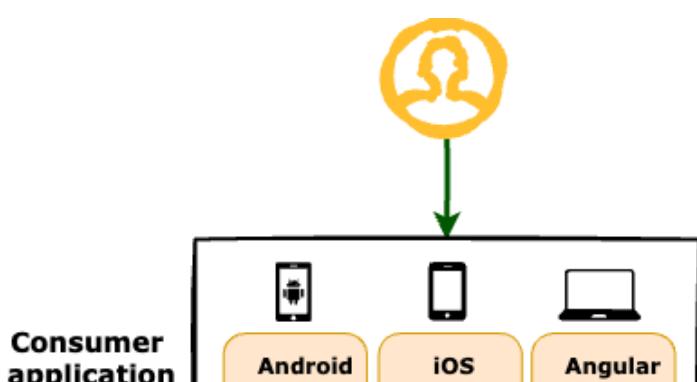
For detailed instructions on how to set up the Advancing Controller microservice, refer to the following guide:

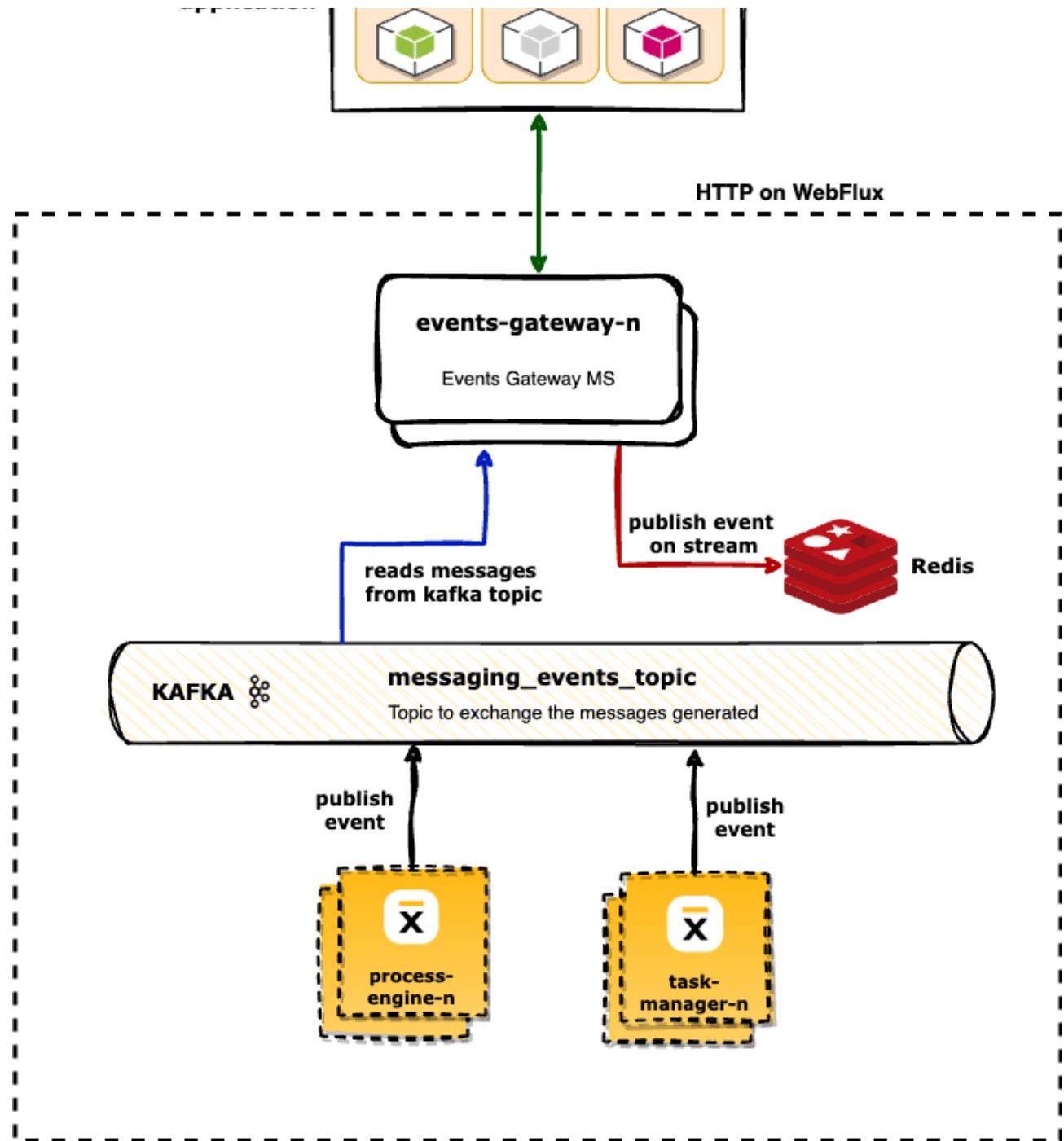
» [Advancing controller setup guide](#)

Was this page helpful?

PLATFORM DEEP DIVE / Core components / Events gateway

Events Gateway a service that centralizes the communication with SSE (Server-sent Events) messages from Backend to Frontend.





It has an important role as a central component for handling and distributing events within the system:

- **Event Processing:** The events-gateway system is responsible for receiving and processing events from various sources, such as the Process Engine and Task Manager. It acts as an intermediary between these systems and the rest of the system components.
- **Message Distribution:** The events-gateway system reads messages from the Kafka topic (`messaging_events_topic`) and distributes them to relevant components within the system. It ensures that the messages are appropriately routed to the intended recipients for further processing.
- **Event Publication:** The events-gateway system plays a crucial role in publishing events to the frontend renderer (FE renderer). It communicates with the frontend renderer using `HTTP` via `WebFlux`. By publishing events, it enables real-time updates and notifications on the user interface, keeping the user informed about the progress and changes in the system.
- **Integration with Redis:** The events-gateway system also interacts with Redis to publish events on a stream. This allows other components in the system to consume the events from the Redis stream and take appropriate actions based on the received events.

In summary, the events-gateway system acts as a hub for event processing, message distribution, and event publication within the system. It ensures efficient communication and coordination between various system components, facilitating real-time updates and maintaining system consistency.

For more details about how to configure events-gateway microservice, check the following section:

» [Events gateway configuration](#)

Was this page helpful?

PLATFORM DEEP DIVE / Core components / Core extensions / Content management / Using the service

After you deployed the CMS service in your infrastructure, you can start defining and using custom content types, such as different lists (which can have different values for the same code depending on the external system that is used), blog posts etc.

You can also set the default application name to be used in your configuration. This is needed when retrieving the contents.

```
application:  
  defaultApplication: DEFAULT_APPLICATION_NAME
```

If this configuration is not set, the service will use `flowx` as the default value.

Define needed Kafka topics

Kafka topic names can be set by using environment variables:

Default parameter (env var)	Default FLOWX.AI val
KAFKA_TOPIC_REQUEST_CONTENT_IN	ai.flowx.dev.plugin.cms.trigger.r
KAFKA_TOPIC_REQUEST_CONTENT_OUT	ai.flowx.dev.engine.receive.plug

⚠ CAUTION

The Engine is listening for messages on topics with names of a certain pattern, make sure to use an outgoing topic name that matches the pattern configured in the Engine.

Example: Request a label by language or source system code

Used to translate custom codes into labels using the specified [language](#) or a certain [source system](#).

Various external systems and integrations might use different labels for the same information. In the processes, it is easier to use the corresponding code and translate this into the needed label when necessary: for example when sending data to other integrations, when generating documents, etc.

You will need to add a [Kafka send event CMS service](#).

The following values are expected in the request body:

- at least one of `language` and `sourceSystem` should be defined (if you only need the `sourceSystem` to be translated, you can leave `language` empty and

vice versa, but they cannot both be empty)

- a list of `entries` to be translated

Example:

```
{  
  "language": "en-US",  
  "sourceSystem": "CS"  
  "entries": [  
    {  
      "codes": [  
        "ROMANIA",  
        "BAHAMAS"  
      ],  
      "contentDescription": {  
        "name": "country",  
        "application": "flowx",  
        "version": 1,  
        "draft": true  
      }  
    }  
  ]  
}
```

If the value for `application` is not sent, the `defaultApplication` value will be used when retrieving the contents from the database.

`version` and `draft` are not mandatory, if they are not specified, the latest published content will be used.

The service will respond with the following message structure:

```
{  
  "entries": [  
    {  
      "code": "ROMANIA",  
      "label": "ROMANIA -en"  
      "translatedCode": "ROMANIA-CS"  
    },  
  
    {  
      "code": "BAHAMAS",  
      "label": "BAHAMAS -en"  
      "translatedCode": "BAHAMAS-CS"  
    }  
  ],  
  "error": null
```

Was this page helpful?

PLATFORM DEEP DIVE / Core components / Core extensions / Content management / Enumerations

A collection of values that can be utilized as content in UI components or templates is managed using

The fallback content to display on prerendering

- . Values can be defined for certain **source systems** or **languages**.

Name	Version	Draft	Last updated
ActivityDomai...	4	<input checked="" type="checkbox"/>	24 Jun 2022, 5:47 PM
ActivityDomai...	3	<input checked="" type="checkbox"/>	23 Jun 2022, 4:21 PM
ActivityDomai...	4	<input checked="" type="checkbox"/>	24 Jun 2022, 5:47 PM
County	5	<input checked="" type="checkbox"/>	24 Jun 2022, 5:47 PM
Enumeration_...	1	<input checked="" type="checkbox"/>	27 Jun 2022, 1:09 PM

On the main screen inside **Enumerations**, you have the following elements:

- **Name** - the name of the enumeration
- **Version** - the version of the enumeration
- **Draft** - switch button used to control the status of an enumeration, could be **Draft or Published**
- **Last Updated** - the last time an enumeration has been updated
- **Open** - button used to access an enumeration to configure it/ add more values, etc.
- **Delete** - button used to delete an enumeration
- **New enumeration** - button used to create a new enumeration
- **Breadcrumbs >** Import/Export**

For each entry (when you hit the **Open** button) inside an enumeration we have to define the following properties:

- **Code** - not displayed in the end-user interface, but used to assure value uniqueness
- **Labels** - strings that are displayed in the end-user interface, according to the language set for the generated solution
- **External source systems codes** - values that are set for each external system that might consume data from the process; these codes are further used by connectors, in order to send to an external system a value that it can validate

Values for ActivityDomain_Companies						New value
Code	ro	en	de	FLEX	FLOWX	
A	Agricultur...	Agricultur...	-	-	-	» ✍ trash
B	Extractiv...	Extractiv...	-	-	-	» ✍ trash
C	Industria ...	Manufact...	-	-	-	» ✍ trash
D	Productio...	Productio...	-	-	-	» ✍ trash

Adding a new enumeration

To add a new enumeration, follow the next steps:

1. Go to **FLOWX Designer** and select the **Content Management** tab.
2. Select **Enumerations** from the list.
3. Add a suggestive name for your enumeration and then click **Add**.

Enumerations

N

The screenshot shows a table of enumerations on the left and a modal dialog on the right.

Name	Version	Draft	Last upc
Activity			24 Jun
Activity			23 Jun
Activity			24 Jun
County			24 Jun
Enumeration		Add	27 Jun
StatusFor_CPA_Test	3		24 Jun
Tara_Test	3		24 Jun
Test_Andrei	3		24 Jun

The modal dialog is titled "Add new enumeration" and contains a single input field labeled "Name".

Configuring an enumeration

After creating an enumeration, you can add values to it.

To configure an enumeration value, follow the next steps:

1. Go to FLOWX.AI Designer and select the **Content Management** tab.
2. Select **Enumerations** from the list and open an enumeration.
3. Click **New value** and fill in the necessary details:

- **Code** - as mentioned above, this is not displayed in the end-user interface but is used to assure value uniqueness
- **Labels** - set the value of the string for each language you would like to use
- **Source Systems** - values that are set for each external system that might consume data from the process

Code	ro	en	de	FLEX	FLOWX
test_enumer...	-	-	-	-	-

Creating a child collection

Enumerations can also be defined as a hierarchy - for each entry, we can define a list of children values (for example, name of the countries defined under the continents' enumeration values); hierarchies further enable cascading values in the end-user interface (for example, after selecting a continent in the first select **UI component**, the second select component will contain only the children of this continent).

Code	ro	en	de	FLEX	FLOWX
test_test	Test	-	-	-	Test

Importing/exporting an enumeration

You can use the import/export feature to import or export enumerations using the following formats:

- JSON
- CSV

The screenshot shows the FLOWX.AI platform interface. On the left is a sidebar with navigation links: Processes, Definitions, Active process; Content Management (with Enumerations selected); Plugins, Task Manager. The main area is titled 'Enumerations' and lists several entries:

Name	Version	Draft	Last upd	Actions
Activity Domain Companies	4	OFF	24 Jun	» 🗑
Test Enumerations	3	OFF	23 Jun	» 🗑
Example	4	OFF	24 Jun	» 🗑
County	5	OFF	24 Jun 2022, 5:47 PM	» 🗑
Test	3	OFF	24 Jun 2022, 5:47 PM	» 🗑
Test Monthly Income	3	OFF	24 Jun 2022, 5:47 PM	» 🗑
Test Activity Domains	3	OFF	24 Jun 2022, 5:47 PM	» 🗑
Monthly Income	3	OFF	24 Jun 2022, 5:47 PM	» 🗑

A context menu is open over the 'Activity Domain Companies' entry, showing options: Import (from JSON, from CSV), Export (to JSON, to CSV). The 'from CSV' and 'to CSV' options are highlighted with red boxes.

Enumerations example

Enumerations, for instance, can be used to build elaborate lists of values (with children). Assuming you wish to add enumerations for **Activity Domain Companies**, you can create children collections by grouping lists and other related domains and activities.

We have the following example for **Activity Domain Companies**:

Activity Domain Companies → Agriculture forestry and fishing:

- **Agriculture, hunting, and related services →**

▶ Cultivation of non-perennial plants:

▶ Cultivation of plants from permanent crops:

▶ Animal husbandry:

- **Forestry and logging →**

▶ Forestry and other forestry activities:

▶ Logging:

▶ Collection of non-wood forest products from spontaneous flora:

- **Fisheries and aquaculture →**

▶ Fishing:

▶ Aquaculture:

This is the output after adding all the lists/collections from above:

Enumerations				New enumeration	⋮
Name	Version	Draft	Last updated		
ActivityDomain_Companies	6	<input checked="" type="checkbox"/>	27 Jun 2022, 6:00 PM	»	trash
ActivityDomain_Companies-A-01-01	3	<input checked="" type="checkbox"/>	23 Jun 2022, 4:21 PM	»	trash
ActivityDomain_Companies-B-05-062	6	<input checked="" type="checkbox"/>	27 Jun 2022, 6:00 PM	»	trash
County	7	<input checked="" type="checkbox"/>	27 Jun 2022, 6:00 PM	»	trash
Enumeration_Tesy	1	<input checked="" type="checkbox"/>	27 Jun 2022, 1:09 PM	»	trash
StatusFATCA_test	5	<input checked="" type="checkbox"/>	27 Jun 2022, 6:00 PM	»	trash
Tara_Test	5	<input checked="" type="checkbox"/>	27 Jun 2022, 6:00 PM	»	trash
Test_Andrei	5	<input checked="" type="checkbox"/>	27 Jun 2022, 6:00 PM	»	trash
VenitLunar_test	5	<input checked="" type="checkbox"/>	27 Jun 2022, 6:00 PM	»	trash

Was this page helpful?

PLATFORM DEEP DIVE / Core components / Core extensions / Content management / Substitution tags

Substitution tags are used to generate dynamic content across the platform. As

The fallback content to display on prerendering

, substitution tags can be defined for each language set for the solution.

Key	ro-RO	en-US	ro	en	fr	de	
emptyState	-	-	Per...	Th...	-	-	
firstName	-	-	Pre...	Fir...	-	-	
openAccount	-	-	De...	Op...	-	-	
openPersonal	-	-	De...	Op...	-	-	
searchAfter	-	-	Ca...	Se...	-	-	
openPf	-	-	De...	Op...	-	-	
res	-	-	Re...	Re...	-	-	

On the main screen inside **Substitution tags**, you have the following elements:

- **Key**
- **Values** - strings that are used in the end-user interface, according to the **language** set for the generated solution
- **Edit** - button used to edit substitution tags
- **Delete** - button used to delete substitution tags
- **New value** - button used to add a new substitution tag
- **Breadcrumbs menu:**
 - **Import**
 - from JSON
 - from CSV
 - **Export**
 - to JSON

- to CSV
- **Search by** - search function used to easily look for a particular substitution tag

Adding new substitution tags

To add a new substitution tag, follow the next steps.

1. Go to
The fallback content to display on prerendering
and select the **Content Management** tab.
2. Select **Substitution tags** from the list.
3. Click **New value**.
4. Fill in the necessary details:
 - Key
 - Languages
5. Click **Add** after you finish.

Sub

Add new substitution tag^x

Key

Key

Languages

ro :

en :

de :

Add

⚠ CAUTION

When working with substitution tags or other elements that imply values from other languages defined in the CMS, when running a

The fallback content to display on prerendering

, the default values extracted will be the ones marked by the default language.

Getting a substitution tag by key

```
public func getTag(forKey key: String) -> String?
```

All substitution tags will be retrieved by the **SDK** before starting the first process and will be stored in memory.

Whenever the container app needs a substitution tag value for populating the UI of the custom components, it can request the substitution tag using the method above, providing the key.

For example, substitution tags can be used to localize the content inside an application.

Example

Localizing the app



You must first check and configure the FLOWX.AI Angular renderer to be able to replicate this example. Click [here](#) for more information.

The `flxLocalize` pipe is found in the `FlxLocalizationModule`.

```
import { FlxLocalizationModule } from 'flowx-process-renderer';
```

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-dummy-component',
  template: `<h3>{{ "stringToLocalize" | flxLocalize}}</h3>`,
})

export class DummyComponent{
  stringToLocalize: string = `@@localizedString`
}
```

Strings that need to be localized must have the ' {@@}' prefix which the **flxLocalize** pipe uses to extract and replace the string with a value found in the substitution tags enumeration.

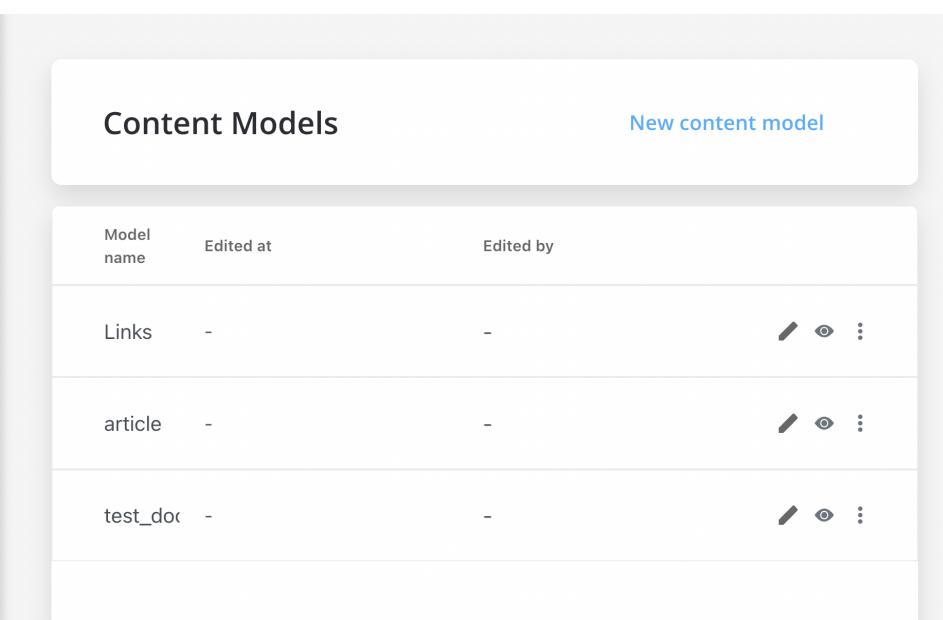
Substitution tags are retrieved when a start process call is first made, and it's cached on subsequent start process calls.

Was this page helpful?

PLATFORM DEEP DIVE / Core components / Core extensions /

Content management / Content models

Content models are used to create complex content collections using customizable pieces of content. For example, you can define content models for an article that will be displayed on a page, creating customizable content.



The screenshot shows the FLOWX.AI platform interface. On the left, there is a sidebar with the following navigation items:

- Processes
 - Definitions
 - Active process
- Content Management
 - Enumerations
 - Substitution tags
 - Content models** (this item is highlighted in blue)
 - Languages
 - Source systems

The main content area is titled "Content Models" and contains a table with three rows of data. The columns are labeled "Model name", "Edited at", and "Edited by". Each row has a set of three icons (pencil, eye, and dots) on the right side.

Model name	Edited at	Edited by
Links	-	-
article	-	-
test_doc	-	-

On the main screen inside **Content models**, you have the following elements:

- **Model name** - the name of the content model
- **Edited at**
- **Edited by**
- **Edit** - button used to edit a content model
- **View process** - click **View process** to open a content model and see its attributes

- **Export/ Delete (Breadcrumbs menu)** - use the breadcrumbs to access the **Export** and **Delete** functions
- **New content model** - button used to add a new content model

For each entry (when you hit the **Open** button) inside an enumeration we have to define the following properties:

- **Language**
- **Name**
- **Attributes (depending on what you add as attributes)**

Configuring attributes

Attributes that can be added to a content model can have the following values:

- **String**
- **Number**
- **Boolean**

Also, attributes can be either **Mandatory** or **Optional**

Adding new content models

To add a new content model, follow the next steps:

1. Go to **FLOWX Designer** and select the **Content Management** tab.
2. Select **Content models** from the list.
3. Click **New content model**.
4. Add a suggestive name for your content model.

5. Fill in the following details:

- **Name** (mandatory)
- **Attributes:**
 - **Name** - add a name for the attribute
 - **Type** - String, Number or Boolean
 - **Mandatory** - mark an Attribute as mandatory
 - **Actions** - add or remove

The screenshot shows a modal window titled "General settings". It contains a "Name" input field and a "Attributes" table. The table has columns: Name, Type, Mandatory, and Actions. A row in the table has a "Property name" input field, a "Type" dropdown set to "String", an empty "Mandatory" checkbox, and a "Actions" column with a checked checkbox and a red "X" button. A context menu is open over the "Type" dropdown, showing options: String (selected), Number, and Boolean. At the bottom left is a "Close" button, and at the bottom right is a "Save" button.

Name	Type	Mandatory	Actions
Property name	String	<input type="checkbox"/>	<input checked="" type="checkbox"/> <input type="button" value="X"/>

Editing content models

To configure an entry inside a content model, follow the next steps:

1. Go to **FLOWX Designer** and select the **Content Management** tab.
2. Select **Content models** from the list and select a content model.

3. Click **View process** and then click **Open**.

4. Click **Edit** to modify the values.

The screenshot shows the FLOWX.AI platform interface. On the left, there is a sidebar with the following navigation items:

- Processes
 - Definitions
 - Active process
- Content Management
 - Enumerations
 - Substitution tags
 - Content models**
- Languages
- Source systems

Below the sidebar, the main content area has a header "Content Models" and a "New content model" button. It displays a table with three rows of content models:

Model name	Edited at	Edited by	Actions
Links	-	-	
article	-	-	
test_docs	-	-	

Was this page helpful?

PLATFORM DEEP DIVE / Core components / Core extensions / Content management / Languages

The FLOWX Headless CMS can store and manage languages. You can add a language and use it in almost any content management configuration.

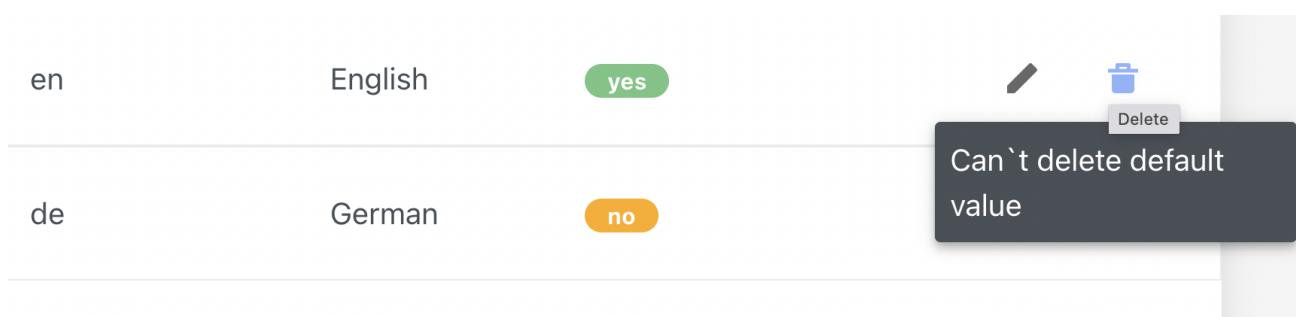
Code	Name	Default
ro	Romanian	no
en	English	yes
de	German	no

On the main screen inside **Languages**, you have the following elements:

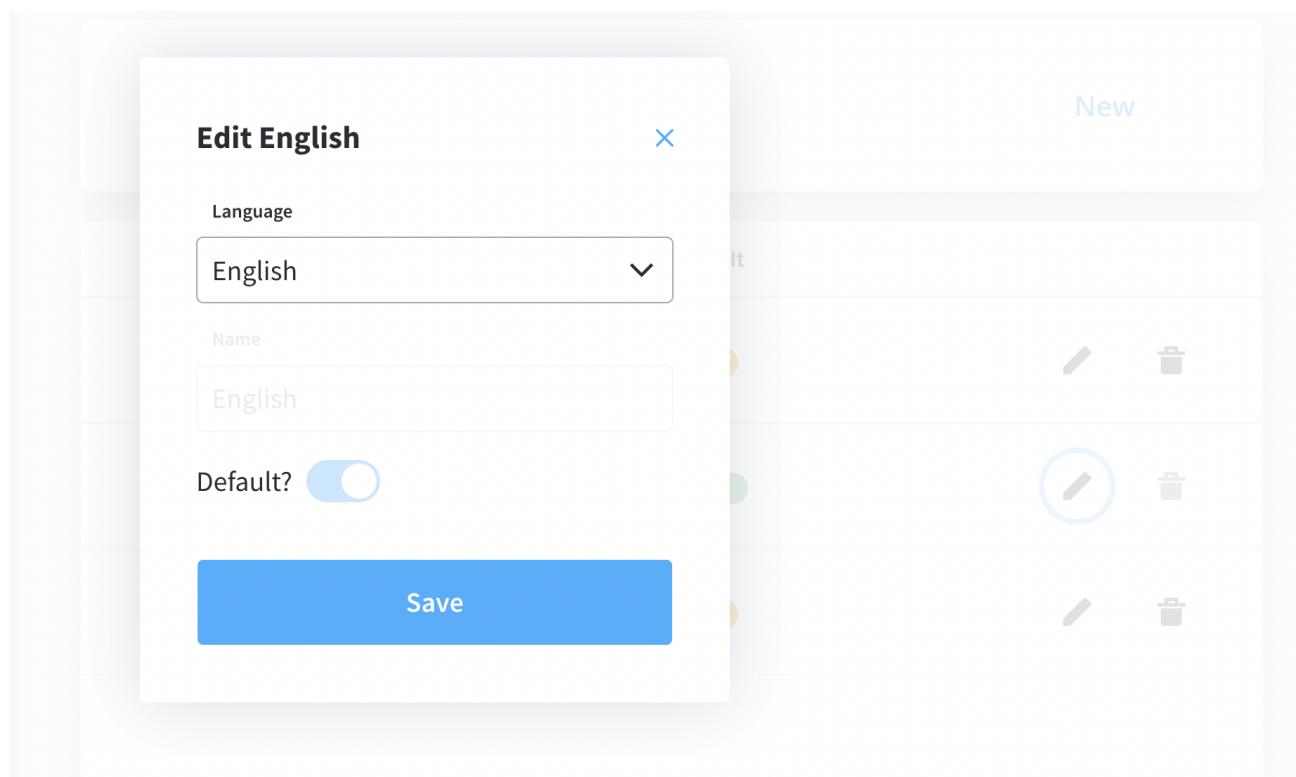
- **Code** - not displayed in the end-user interface, but used to assure value uniqueness
- **Name** - the name of the language
- **Default** - you can set a language as **Default** (default values can't be deleted)

⚠ CAUTION

When working with substitution tags or other elements that imply values from other languages defined in the CMS, when running a process, the default values extracted will be the ones marked by the default language.



- **Edit** - button used to edit a language



- **Delete** - button used to delete a language

🔥 DANGER

Before deleting a language make sure that this is not used in any content management configuration.

- **New** - button used to add a new language

Adding a new language

To add a new language, follow the next steps:

1. Go to
The fallback content to display on prerendering
and select the **Content Management** tab.
2. Select **Languages** from the list.
3. Choose a new **language** from the list.
4. Click **Add** after you finish.

The screenshot shows the FLOWX.AI platform's sidebar navigation on the left, which includes sections for Content Management, Plugins, and General Settings. The Content Management section is expanded, showing options like Enumerations, Substitution tags, Content models, Languages (which is highlighted in blue), and Source systems. The Plugins section is also expanded, showing Task Manager, Hooks, Stages, Notification templates, and Document templates. The General Settings section is collapsed. To the right of the sidebar, a modal window titled "Languages" is open. It has a "New" button in the top right corner. The main area lists three languages: Romanian (Code: ro, Name: Romanian, Default: no), English (Code: en, Name: English, Default: yes), and German (Code: de, Name: German, Default: no). Each language entry has edit and delete icons.

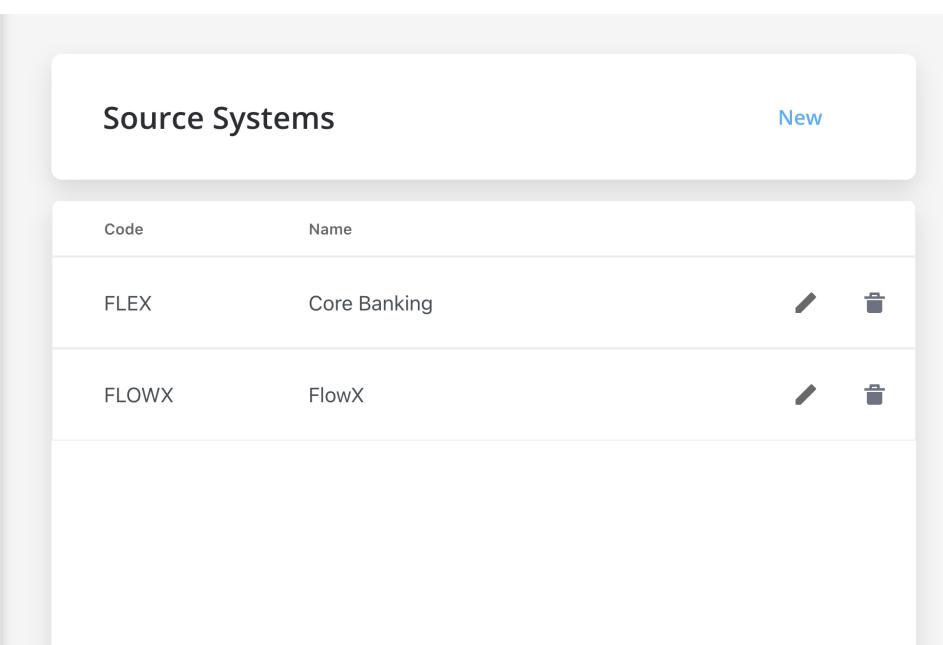
Code	Name	Default	Action
ro	Romanian	no	
en	English	yes	
de	German	no	

Was this page helpful?

PLATFORM DEEP DIVE / Core components / Core extensions / Content management / Source systems

If multiple

The fallback content to display on prerendering values are needed to communicate with other systems, source systems can be used.



The screenshot shows the FLOWX.AI interface with the sidebar collapsed. The main area is titled "Source Systems" with a "New" button in the top right. A table lists two entries:

Code	Name	Actions
FLEX	Core Banking	
FLOWX	FlowX	

The sidebar on the left includes sections for Processes (Definitions, Active process), Content Management (Enumerations, Substitution tags, Content models, Languages), and Source systems, which is highlighted in blue.

On the main screen inside **Source systems**, you have the following elements:

- **Code** - not displayed in the end-user interface, but used to assure value uniqueness

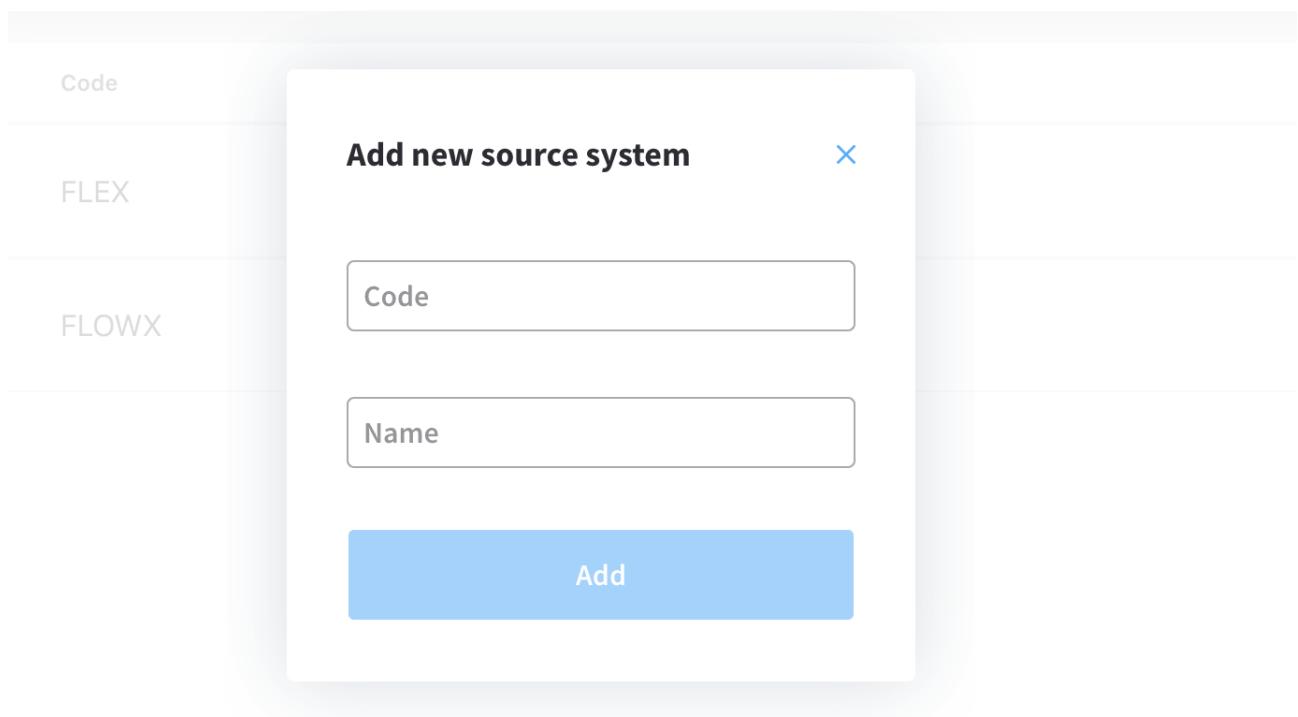
- **Name** - the name of the source system
- **Edit** - button used to edit a source system
- **Delete** - button used to delete a source system
- **New** - button used to add a new source system

Adding new source systems

To add a new source system, follow the next steps.

1. Go to
The fallback content to display on prerendering
and select the **Content Management** tab.
2. Select **Source systems** from the list.
3. Fill in the necessary details:
 - Code
 - Name
4. Click **Add** after you finish.

Source Systems



Was this page helpful?

**PLATFORM DEEP DIVE / Core
components / Core extensions /
Content management / Media library**

The media library serves as a centralized hub for managing and organizing various types of media files, including images, GIFs, and more. It encompasses all the files that have been uploaded to the

The fallback content to display on prerendering , providing a convenient location to view, organize, and upload new media files.

The screenshot shows the FLOWX.AI platform's interface. On the left, there is a sidebar with the following navigation items:

- Processes
- Definitions
- Active process
- Process Instances
- Failed process start
- Content Management
 - Enumerations
 - Substitution tags
 - Content models
 - Languages
 - Source systems
 - Media Library
- Plugins
 - Task Manager
 - All tasks
 - Hooks
 - Stages
 - Allocation rules
 - Out of office

At the bottom of the sidebar, there is a user profile for "John Doe" and a three-dot menu icon.

The main area is titled "Media library" and contains a table of uploaded files:

Preview	Key	Format	Size	Edited at	Edited by	Actions
	silviu	webp	0.51 MB	04 Oct 2022, 11:29 AM	John Doe	
	teamwork	png	0.1 MB	04 Oct 2022, 9:24 AM	John Doe	
	dodge-challenger	jpeg	0.19 MB	04 Oct 2022, 9:24 AM	John Doe	
	viper_2	jpeg	0.82 MB	04 Oct 2022, 9:24 AM	John Doe	
	dodge viper	jpeg	0.05 MB	04 Oct 2022, 9:24 AM	John Doe	
	impala	gif	0.74 MB	04 Oct 2022, 9:24 AM	Jane Doe	
	lamborghini	png	0.68 MB	04 Oct 2022, 9:24 AM	John Doe	

!(INFO)

You can also upload an image directly to the Media Library on the spot when configuring a process using the **UI Designer**. More information [here](#).

Uploading a new asset

To upload an asset to the Media Library, follow the next steps:

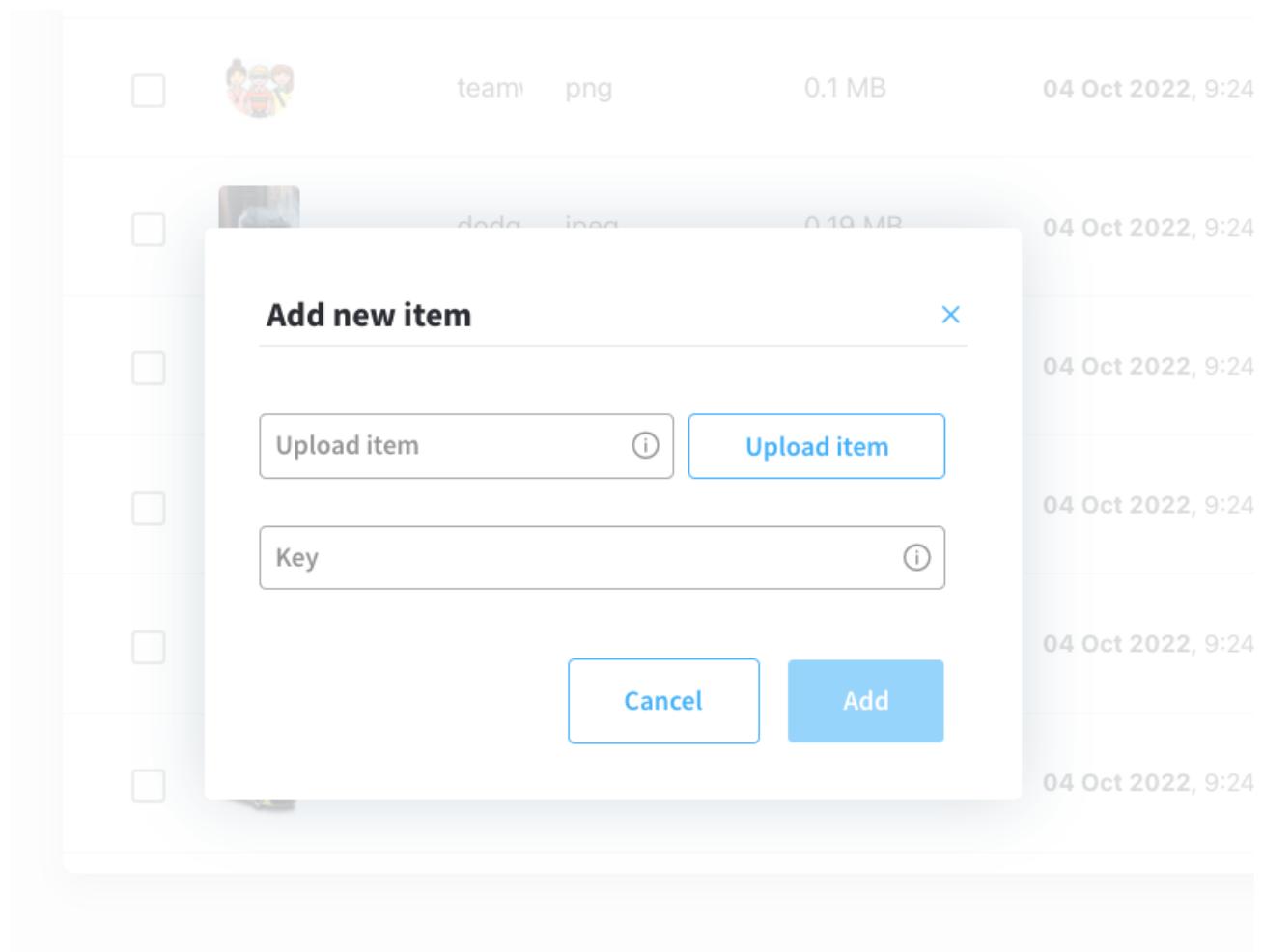
1. Open

The fallback content to display on prerendering

2. Go to **Content Management** tab and select **Media Library**.

3. Click **Add new item**, the following details will be displayed:

- **Upload item** - opens a local file browser
- **Key** - the key must be unique, you cannot change it afterwards



4. Click **Upload item** button and select a file from your local browser.
5. Click **Upload item** button again to upload the asset.

 **CAUTION**

Supported formats: PNG, JPEG, JPG, GIF, SVG or WebP format, 1 MB maximum size.

Displaying assets

Users can preview all the uploaded assets just by accessing the **Media Library**.

You have the following information about assets:

- Preview (thumbnail 48x48)
- Key
- Format ("-" for unknown format)
- Size
- Edited at
- Edited by

Media library

[New item](#) [⋮](#)

Preview	Key	Format	Size	Edited at	Edited by	⋮
<input type="checkbox"/>	 silviu	webp	0.51 MB	04 Oct 2022, 11:29 AM	John Doe	 
<input type="checkbox"/>	 teamwork	png	0.1 MB	04 Oct 2022, 9:24 AM	John Doe	 
<input type="checkbox"/>	 dodge-challenger	jpeg	0.19 MB	04 Oct 2022, 9:24 AM	John Doe	 
<input type="checkbox"/>	 viper_2	jpeg	0.82 MB	04 Oct 2022, 9:24 AM	John Doe	 
<input type="checkbox"/>	 dodge viper	jpeg	0.05 MB	04 Oct 2022, 9:24 AM	John Doe	 
<input type="checkbox"/>	 impala	gif	0.74 MB	04 Oct 2022, 9:24 AM	John Doe	 
<input type="checkbox"/>	 lamborghini	png	0.68 MB	04 Oct 2022, 9:24 AM	John Doe	 

Searching assets

You can search an asset by using its key (full or substring).

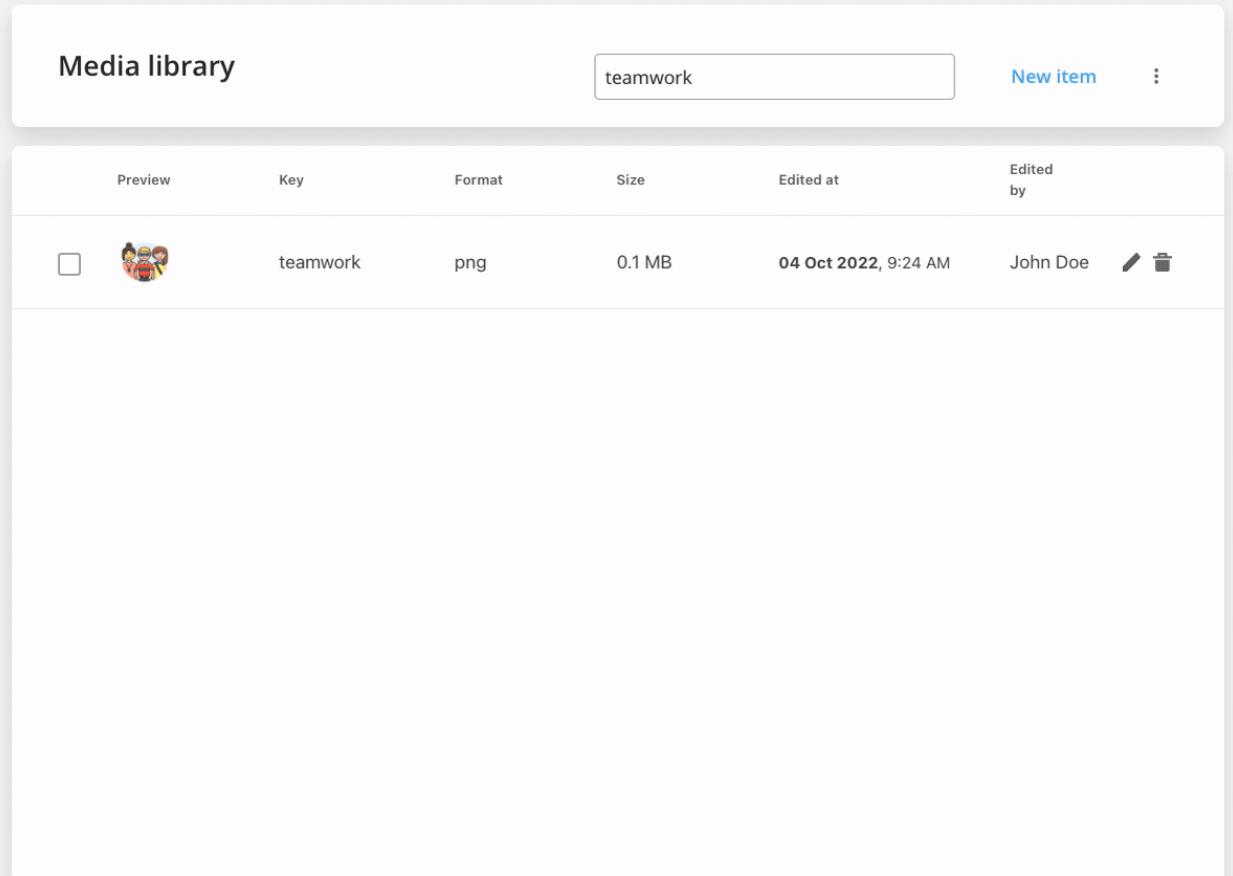
Media library

[New item](#) [⋮](#)

Preview	Key	Format	Size	Edited at	Edited by	⋮
<input type="checkbox"/>	 teamwork	png	0.1 MB	04 Oct 2022, 9:24 AM	John Doe	 

Replacing assets

You can replace an item on a specific key (this will not break references to process definitions).



The screenshot shows a 'Media library' interface. At the top, there is a search bar containing the text 'teamwork' and a 'New item' button. Below the search bar is a table header with columns: Preview, Key, Format, Size, Edited at, and Edited by. A single item is listed in the table:

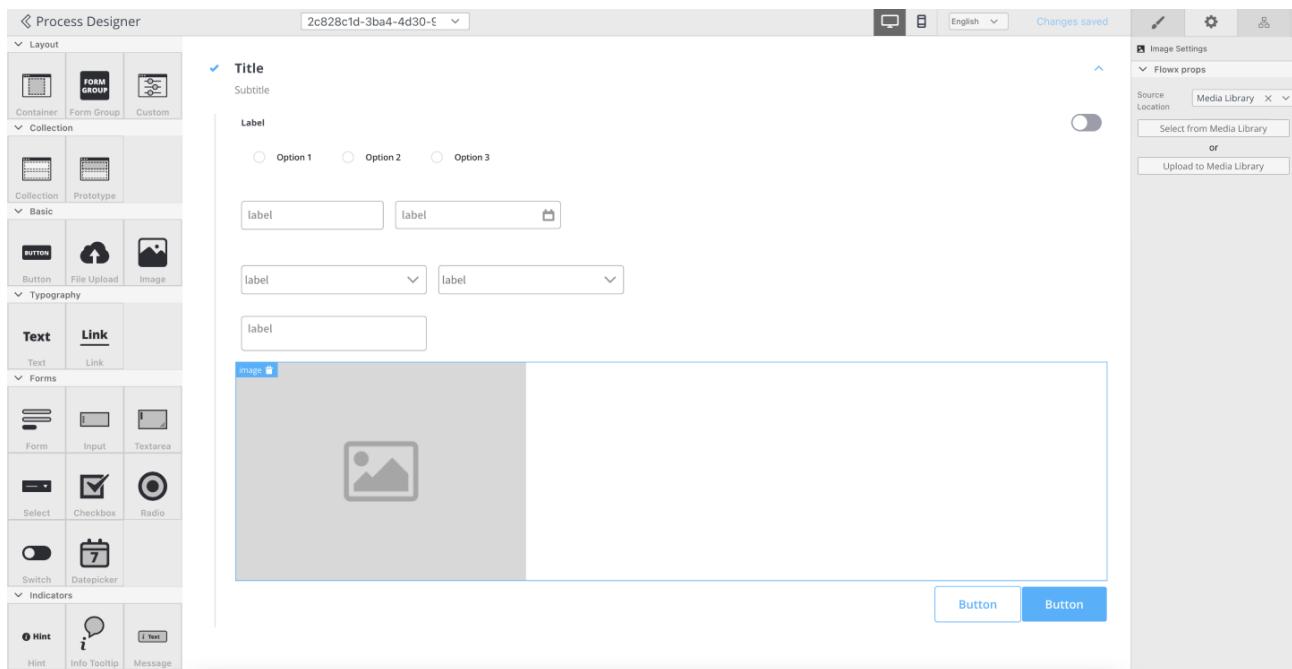
Preview	Key	Format	Size	Edited at	Edited by
<input type="checkbox"/> 	teamwork	png	0.1 MB	04 Oct 2022, 9:24 AM	John Doe  

Referencing assets in UI Designer

You have the following options when configuring image components using **UI Designer**:

- Source Location - here you must select **Media Library** as source location
- Image Key

- **Option 1:** trigger a dropdown with images keys - you can type and filter options or can select from the initial list in dropdown
- **Option 2:** open a popup with images thumbnails and keys then you can type and filter options or can select from the initial list

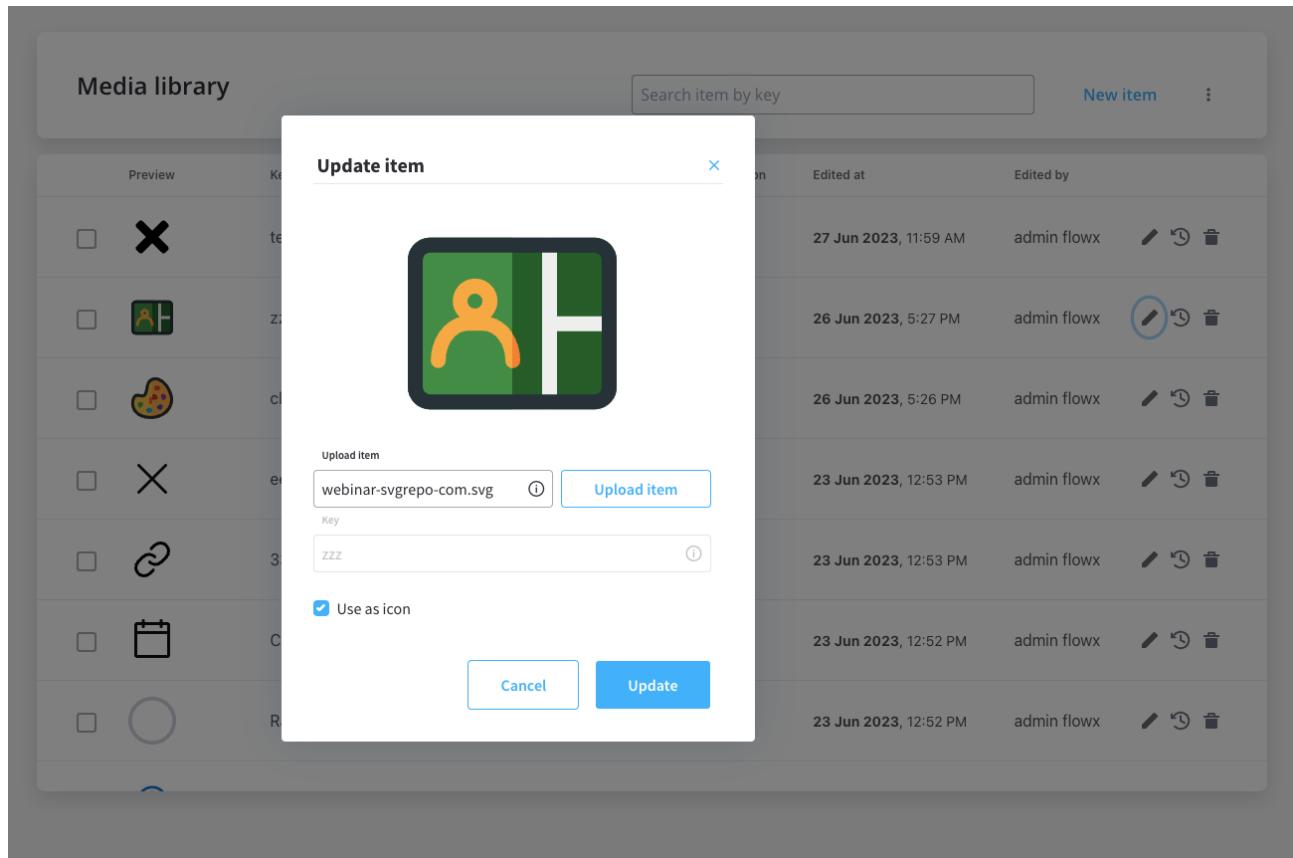


!(INFO)

More details on how to configure an image component using UI Designer - [here](#).

Icons

The Icons feature allows you to personalize the icons used in UI elements. By uploading SVG files through the Media Library and marking them, you can choose icons from the available list in the UI Designer.



!(info)

When selecting icons in the UI Designer, only SVG files marked as icons in the Media Library will be displayed.

!(info)

To ensure optimal visual rendering and alignment within your UI elements, it is recommended to use icons with small sizes such as: 16px, 24px, 32px.

Using icons specifically designed for these sizes helps maintain consistency and ensures a visually pleasing user interface. It is advisable to select icons

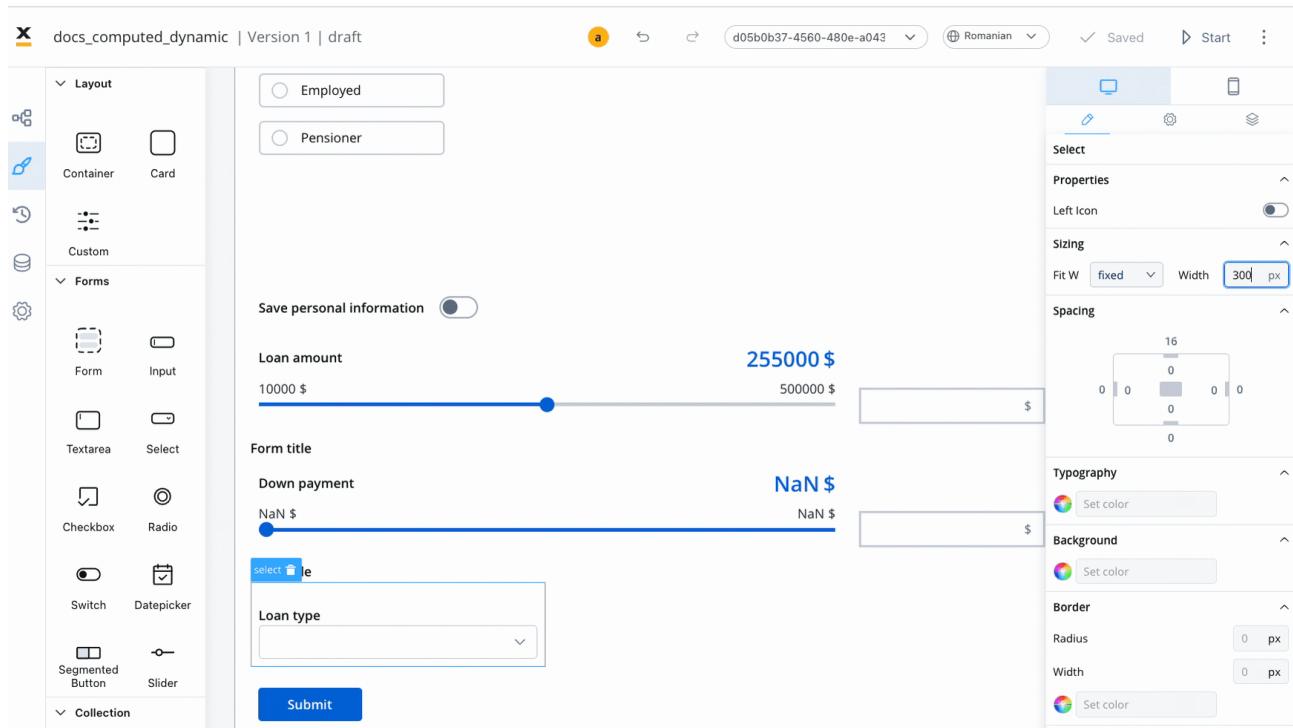
from icon sets that provide these size options or to resize icons proportionally to fit within these dimensions.

⚠ CAUTION

Icons are displayed or rendered at their original, inherent size.

Customization

Content-specific icons pertain to the content of UI elements, such as icons for **input fields** or **send message buttons**. These icons are readily accessible in the **UI Designer**.



More details on how to add icons on each element, check the sections below:

» Input element

» Select element

» Buttons

Export/import media assets

The import/export feature allows you to import or export media assets, enabling easy transfer and management of supported types of media files.

	Preview	Key	Format	Size	Last modified
<input type="checkbox"/>		Fil	svg	0.78 KB	31 May 2023, 11:38 AM
<input type="checkbox"/>		Fil	svg	0.57 KB	31 May 2023, 11:38 AM
<input type="checkbox"/>		Ar	svg	0.89 KB	31 May 2023, 11:37 AM
<input type="checkbox"/>		Ar	svg	0.97 KB	31 May 2023, 11:37 AM
<input type="checkbox"/>		H_	svg	0.41 KB	30 May 2023, 3:56 PM
<input type="checkbox"/>		H_	svg	0.52 KB	30 May 2023, 3:56 PM
<input type="checkbox"/>		Ct	svg	0.2 KB	30 May 2023, 3:56 PM

Import media assets

Use this function to import media assets of various supported types. It provides a convenient way to bring in images, videos, or other media resources.

Export all

Use this function to export all media assets stored in your application or system. The exported data will be in JSON format, allowing for easy sharing, backup, or migration of the media assets.

The exported JSON structure will resemble the following example:

```
{  
  "images": [  
    {  
      "key": "cart",  
      "application": "flowx",  
      "filename": "maxresdefault.jpg",  
      "format": "jpeg",  
      "contentType": "image/jpeg",  
      "size": 39593,  
      "storagePath":  
        "https://d22tnnndi9lo60.cloudfront.net/devmain/flowx/cart/1681951553544_maxresdefault.jpg",  
      "thumbnailStoragePath":  
        "https://d22tnnndi9lo60.cloudfront.net/devmain/flowx/cart/1681951553544_maxresdefault.jpg",  
    },  
    {  
      "key": "pizza",  
      "application": "flowx",  
      "filename": "pizza.jpeg",  
      "format": "jpeg",  
      "contentType": "image/jpeg",  
      "size": 22845,  
      "storagePath":  
        "https://d22tnnndi9lo60.cloudfront.net/devmain/flowx/pizza/1681951553544_pizza.jpeg",  
      "thumbnailStoragePath":  
        "https://d22tnnndi9lo60.cloudfront.net/devmain/flowx/pizza/1681951553544_pizza.jpeg",  
    }  
  ],  
  "exportVersion": 1  
}
```

- `images` - is an array that contains multiple objects, each representing an image
- `exportVersion` - represents the version number of the exported data, it holds the image-related information
- `key` - represents a unique identifier or name for the image, it helps identify and differentiate images within the context of the application
- `application` - specifies the name or identifier of the application associated with the image, it indicates which application or system the image is related to
- `filename` - the name of the file for the image, it represents the original filename of the image file
- `format` - a string property that specifies the format or file extension of the image
- `contentType` - the MIME type or content type of the image, it specifies the type of data contained within the image file
- `size` - represents the size of the image file in bytes, it indicates the file's storage size on a disk or in a data storage system
- `storagePath` - the URL or path to the location where the original image file is stored, it points to the location from where the image can be accessed or retrieved
- `thumbnailStoragePath` - the URL or path to the location where a thumbnail version of the image is stored, it points to the location from where the thumbnail image can be accessed or retrieved

Was this page helpful?

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

Parameter	Description
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**.

The screenshot shows the FLOWX.AI platform interface. On the left is a sidebar with navigation links: Processes (Definitions, Active process, Process Instances, Failed process start), Content Management (Enumerations, Substitution tags, Content models, Languages, Source systems, Media Library), Plugins (Task Manager, All tasks, Hooks, Stages, Allocation rules, Out of office, Notification templates), and a user icon (Admin Flowx). The main area is titled "Process Definitions" and contains two sections: "Drafts / In progress" and "Published". The "Drafts / In progress" section lists four entries:

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

The "Published" section lists four entries:

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 environment).

The screenshot shows the Flowx AI platform's configuration interface. On the left, there is a sidebar with various settings like General Settings (expanded, showing Generic Parameters), Integration management, Licensing, Access management, Users, Roles, Groups, Audit Log, and Platform status. At the bottom left is an 'Admin Flowx' button. The main area displays a list of generic parameters:

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

Using generic parameters

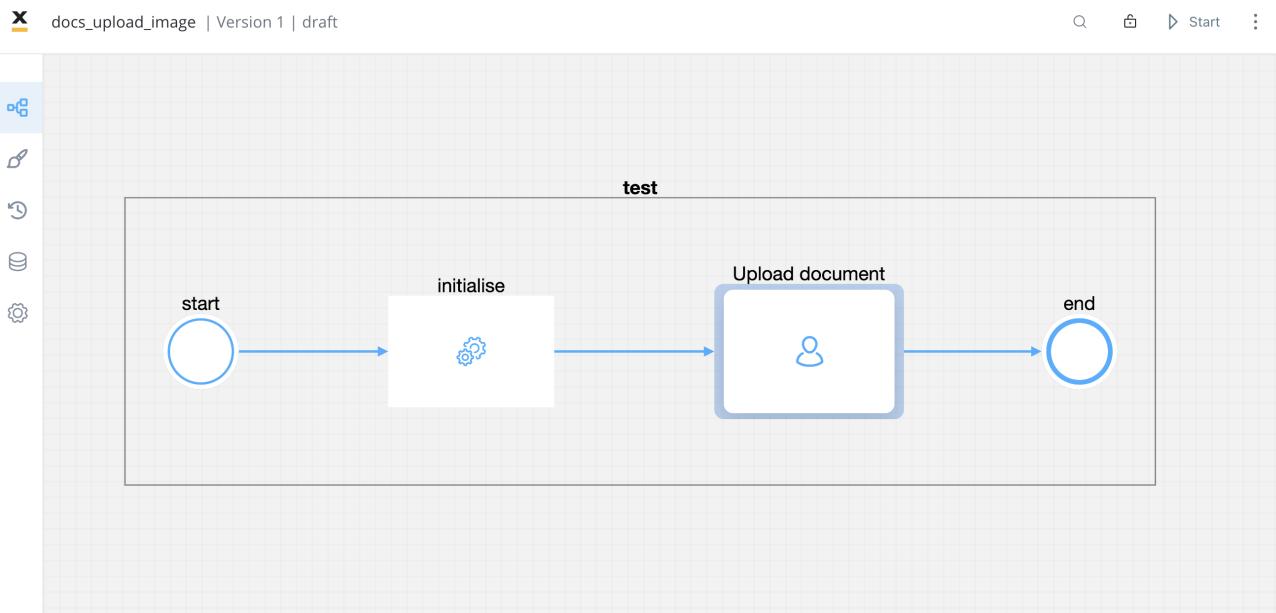
Use case

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



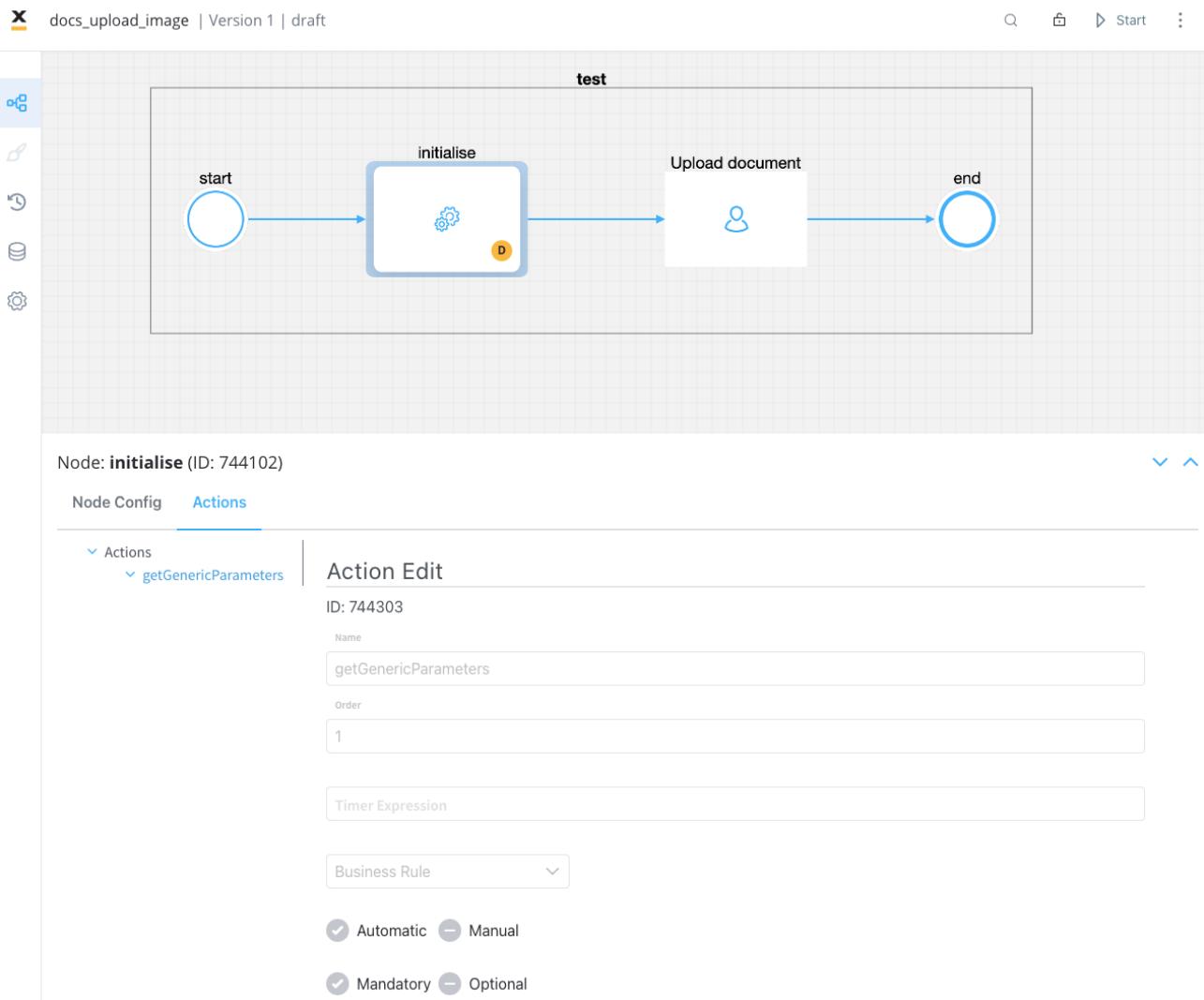
Configuring the task node

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



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"));
```

Parameters

Business Rules

▼ Rule 1 

Language

MVEL

[Test Rule](#)

```
1 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".

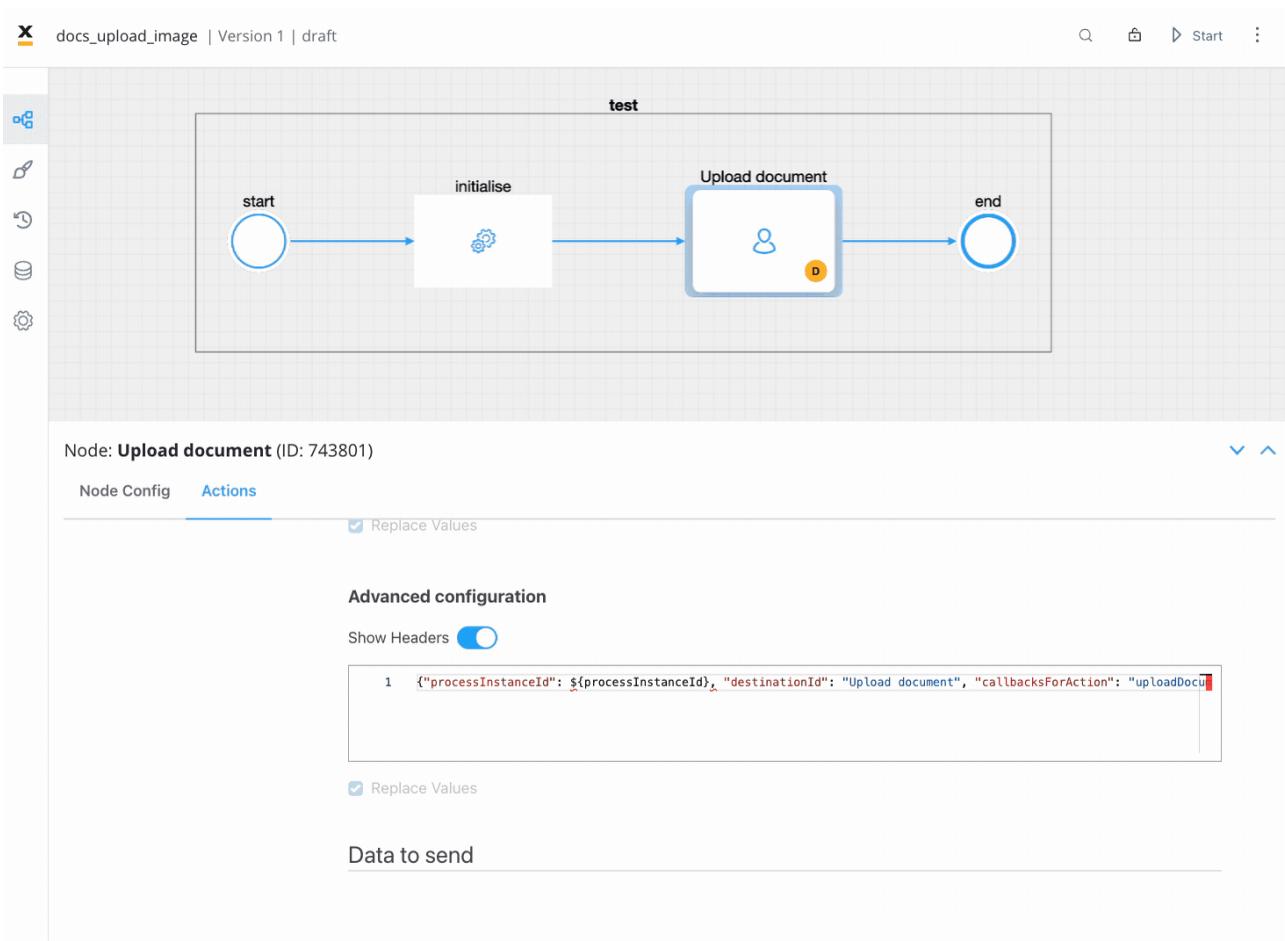
In summary, the code seems to be processing an uploaded document by checking its download path, constructing a full file path, and updating the document object with the new file path. Finally, it outputs the updated document object.

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:



Was this page helpful?

PLATFORM DEEP DIVE / Core components / Core extensions / Integration management / Configuring access rights for Integration Management

Granular access rights can be configured for restricting access to the Integration Management plugin component. These access rights must be configured in the Designer (admin) deployment.

The following access authorizations are provided, with the specified access scopes:

1. **Manage-integrations** - for configuring access for managing integration management

Available scopes:

- import - users can import integrations
- read - users can view integrations
- edit - users can edit integrations
- admin - users can delete integrations

Integration management is preconfigured with the following default users roles for each of the access scopes mentioned above:

- manage-integrations
 - import:
 - ROLE_ADMIN_MANAGE_INTEGRATIONS_IMPORT
 - ROLE_ADMIN_MANAGE_INTEGRATIONS_EDIT
 - ROLE_ADMIN_MANAGE_INTEGRATIONS_ADMIN
 - read:
 - ROLE_ADMIN_MANAGE_INTEGRATIONS_READ
 - ROLE_ADMIN_MANAGE_INTEGRATIONS_IMPORT
 - ROLE_ADMIN_MANAGE_INTEGRATIONS_EDIT

- ROLE_ADMIN_MANAGE_INTEGRATIONS_ADMIN
- edit:
 - ROLE_ADMIN_MANAGE_INTEGRATIONS_EDIT
 - ROLE_ADMIN_MANAGE_INTEGRATIONS_ADMIN
- admin:
 - ROLE_ADMIN_MANAGE_INTEGRATIONS_ADMIN

DANGER

These roles need to be defined in the chosen identity provider solution. It can be either kycloak, RH-SSO, or another identity provider solution. For more details on how to define service accounts, check the Access rights section.

In case other custom roles are needed, you can configure them using environment variables. More than one role can be set for each access scope.

To configure access for each of the roles above, adapt the following input:

```
SECURITY_ACCESSAUTHORIZATIONS_AUTHORIZATIONNAME_SCOPES_SCOPENAME_ROLESALLOWED: NEEDED_ROLE_NAMES
```

Possible values for AUTHORIZATIONNAME: MANAGEDOCUMENTTEMPLATES.

Possible values for SCOPENAME: import, read, edit, admin.

For example, if you need to configure role access for read, insert this:

```
SECURITY_ACCESSAUTHORIZATIONS_MANAGEINTEGRATIONS_SCOPES_READ_ROLE_NAME_TEST
```

Was this page helpful?

PLATFORM DEEP DIVE / Core components / Core extensions / Licensing

The License Engine is part of the core components of the

The fallback content to display on prerendering

. It is used for displaying reports regarding the usage of the platform in the

The fallback content to display on prerendering

It can be quickly deployed on the chosen infrastructure and then connected to the

The fallback content to display on prerendering

through

The fallback content to display on prerendering events.

Let's go through the steps needed in order to deploy and set up the service:

» [License engine setup guide](#)

Multiple roles are available in the license engine, here are the steps for configuring them:

» [Configuring access roles \(old\)](#)

Was this page helpful?

PLATFORM DEEP DIVE / Core components / Core extensions / Audit log

The Audit Log service provides a centralized location for all audit events. The following details are available for each event:

- **Timestamp** - the date and time the event occurred, the timestamp is displayed in a reversed chronologically order
- **User** - the entity who initiated the event, could be a username or a system
- **Subject** - the area or component of the system affected by the event

► Possible values

- **Event** - the specific action that occurred

► Possible values

- **Subject identifier** - the name related to the subject, there are different types of identifiers based on the selected subject
- **Version** - the version of the process definition at the time of the event
- **Status** - the outcome of the event (e.g. success or failure)

Audit Logs							
Timestamp	User	Section	Subject	Event	Subject Identifier	Status	Actions
20 Jan 2023, 12:07 PM	john.doe@email.com	Process Instance	Process Instance	View	b1a7359c-0d45-4c3e-8bd1-ede16ed5c8f5	success	
20 Jan 2023, 12:07 PM	john.doe@email.com	Process Instance	Process Instance	Start	2e3ba843-3e8a-43f9-a4cf-e599bd7d19bf	success	
20 Jan 2023, 11:39 AM	john.doe@email.com	Process versioning	Process Definition	Created	name_of_the_process	success	
20 Jan 2023, 11:37 AM	john.doe@email.com	BPMN Diagram	Node	Update	node_name	success	
20 Jan 2023, 11:37 AM	john.doe@email.com	BPMN Diagram	Node	Update	test_node	success	
20 Jan 2023, 11:35 AM	jane.doe@email.com	BPMN Diagram	Action	Created	name_of_the_action	success	
20 Jan 2023, 11:35 AM	jane.doe@email.com	BPMN Diagram	Connector	Delete	8db94d59-837f-4c42-a60c-3d38b110c19.	success	
20 Jan 2023, 11:35 AM	jane.doe@email.com	BPMN Diagram	Connector	Created	8db94d59-837f-4c42-a60c-3d38b110c19.	success	

Filtering

Users can filter audit records by event date and by selecting specific options for User, Subject, and Subject Identifier.

- Filter by event date

Audit Logs										
Timestamp	User	Section	Subject	Event	Subject Identifier	Status				
Filter by event date										
20 Jan 2023		Process Instance	Process Instance	View	b1a7359c-0d45-4c3e-8bd1-ed16ed5c8f5	success				
20 Jan 2023	< January 2023 >	Process Instance	Process Instance	Start	2e3ba843-3e8a-43f9-a4cf-e599bd7d19bf	success				
20 Jan 2023	Su Mo Tu We Th Fr Sa	Process Definition	Process Definition	Created	name_of_the_process	success				
20 Jan 2023	1 2 3 4 5 6 7	Program	Node	Update	node_name	success				
20 Jan 2023	8 9 10 11 12 13 14	Program	Node	Update	test_node	success				
20 Jan 2023	15 16 17 18 19 20 21	Program	Action	Created	name_of_the_action	success				
20 Jan 2023	22 23 24 25 26 27 28	Program	Connector	Delete	8db94d59-837f-4c42-a60c-3d38b110c19.	success				
20 Jan 2023	29 30 31 1 2 3 4	Program	Connector	Created	8db94d59-837f-4c42-a60c-3d38b110c19.	success				
	^ ^ ^ 12 : 41 PM v v v									
	Today									

- User - single selection, type at least 4 characters
- Subject - single selection
- Subject identifier - exact match

Audit log details

To view additional details for a specific event, users can click the eye icon on the right of the event in the list. Additional information available in the audit log details window includes Here you have the following information:

- Event - the specific action that occurred
- URL - the URL associated with the event
- Body - any additional data or information related to the event

Audit Logs Filter by event date Search by user Subject See

Audit log details X

Event: ui designer, created, 06 Oct 2022 at 6:01 PM

Url: <https://admin.qa.flowxai.dev/api/nodes/716516/templates>

Body:

```
1  {"uiTemplateParentId":6292714,  
 "componentIdentifier":"FORM_GROUP","type":"FLOWX","order":0,  
 "key":"","platformsDisplayOptions":[{"id":null,"flowxProps":  
 {},"style":null,"flexLayout":{"fxLayout":"column",  
 "fxLayoutAlign":"start stretch","fxLayoutGap":"0px"},  
 "className":null,"platform":"DEFAULT","templateConfigId":null,  
 "formFieldId":null}],"templateConfig":[]}
```

Was this page helpful?

PLATFORM DEEP DIVE / Core components / Core extensions / Scheduler

Overview

The Scheduler is part of the core components of the

The fallback content to display on prerendering

. It can be easily added to your custom FLOWX deployment to **enhance the core platform capabilities with functionality specific to scheduling messages.**

The service offers the possibility to schedule a message that you only need to process after a configured time period.

It can be quickly deployed on the chosen infrastructure and then connected to the

The fallback content to display on prerendering through Kafka events.

Let's go through the steps needed in order to deploy and set up the service:

» [Scheduler setup guide](#)

We've prepared some examples of various use cases where this service is useful:

Using the scheduler

After deploying the scheduler service in your infrastructure, you can start using it to schedule messages that you need to process at a later time.

One such example would be to use the scheduler service to expire processes that were started but haven't been finished.



First you need to check the configured topics match the ones configured in the engine.

For example the engine topics

KAFKA_TOPIC_PROCESS_SCHEDULE_OUT_SET and

KAFKA_TOPIC_PROCESS_SCHEDULE_OUT_STOP **should be the same with the ones configured in the scheduler** (KAFKA_TOPIC_SCHEDULE_IN_SET and KAFKA_TOPIC_SCHEDULE_IN_STOP)

When a process is scheduled to expire, the engine sends the following message to the scheduler service (on the topic KAFKA_TOPIC_SCHEDULE_IN_SET):

```
{  
  "applicationName": "onboarding",  
  "applicationId": "04f82408-ee66-4c68-8162-b693b06bba00",  
  "payload": {  
    "scheduledEventType": "EXPIRE_PROCESS",  
    "processInstanceId": "04f82408-ee66-4c68-8162-  
b693b06bba00"  
  },  
  "scheduledTime": 1621412209.353327,
```

```
"responseTopicName": "ai.flowx.process.expire.staging"  
}
```

The scheduled time should be defined as `java.time.Instant`.

At the scheduled time, the payload will be sent back to the response topic defined in the message, like so:

```
{  
  "scheduledEventType": "EXPIRE_PROCESS",  
  "processInstanceId": "04f82408-ee66-4c68-8162-  
b693b06bba00"  
}
```

If you don't need the scheduled message anymore, you can discard it by sending the following message (on the topic `KAFKA_TOPIC_SCHEDULE_IN_STOP`)

```
{  
  "applicationName": "onboarding",  
  "applicationId": "04f82408-ee66-4c68-8162-b693b06bba00"  
}
```

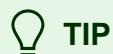
These fields, `applicationName` and `applicationId` are used to uniquely identify a scheduled message.

Was this page helpful?

PLATFORM DEEP DIVE / Core components / Core extensions / Search data service

Search data is a microservice that searches for data in another process.

The new search data microservice enables you to create a process that can perform a search/look for data (using **Kafka send** / **Kafka receive** actions) in other processes.



TIP

Using elastic search, the new search microservice will be able to search for keys that are indexed in ES, via existing mechanics.



CAUTION

Elastic search indexing must be switched on the FLOWX.AI Engine configuration. You can find more details in the **Search data service setup guide**.

Using search data

Use case:

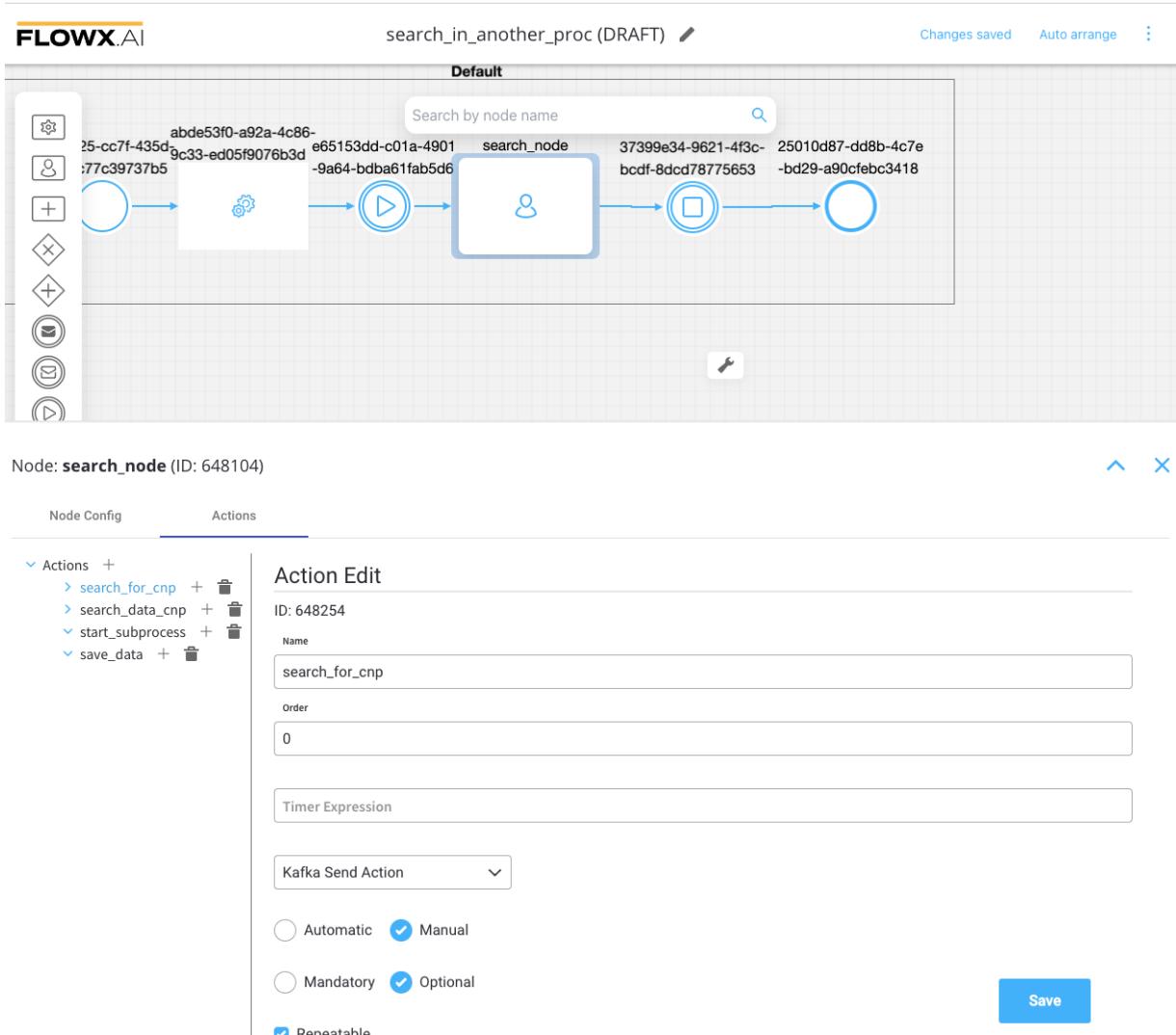
- search for data in other processes
- display results about other processes where the search key was found

1. Create a process using

The fallback content to display on prerendering

2. From the newly created process where you want to perform the search, add a **Task node**.

3. Configure a send event via a **Kafka send action**.



4. Configure the following items:

- **Topic name** - the Kafka topic on which the search service listens for requests; ! respect the **naming pattern**
- **Data to send** - (key) - used when data is sent from the frontend via an action to validate the data (you can find more information in the User Task configuration section)
- **Headers** - required
- **Body message:**
 - `searchKey` - it will hold the result received from the elastic search
 - `value` - value of the key
 - `processDefinitionNames` - the process definition names where to perform the search
 - `processStartDateAfter` - the service will look into process definitions created after the defined date

```
{  
  "searchKey": "application.client.name",  
  "value": "12344",  
  "processStartDateAfter": "formatDeDataStandard", (opt)  
  "processStartDateBefore": "formatDeDataStandard", (opt)  
  "processDefinitionNames": [ "processDef1", "processDef2"  
,  
    "status": [ "ANY", ... ]  
}
```

- Example (dummy values extracted from a process):

Topics

```
ai.flowx.in.qa.data.search.v1
```

Message

```
1  {
2    "searchKey": "application.client.identificationData.lastName",
3    "value": "${searchValue2}",
4    "processDefinitionNames": ["silviu_add_data_process"],
5    "processStartDateAfter": "2022-08-24T13:31:47.912524Z"
6 }
```

Advanced configuration

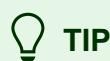
Show Headers

```
1  {"processInstanceId": ${processInstanceId}, "destinationId": "search_node", "callbacksForAction": "search_for_cnp"}]
```

Data to send

Add KeySave

5. A custom microservice (a core extension) will receive this event and will search the value of the process in the elastic search.
6. It will respond to the engine via a Kafka topic.



TIP

The topic must be defined in the **Node config** of the **User task** where you previously added the Kafka Send Action.

The **body message** of the response will look like this:

! If there is no result:

```
{  
  "searchKey": "application.client.name",  
  "result": [],  
  "processStartDate": date,  
  "tooManyResults": true|false  
}
```

- Example (dummy values extracted from a process):



To access the view of your process variables, tokens and subprocesses go to **FLOWX.AI Designer > Active process > Process Instances**. Here you will find the response.

Process definition: **search_in_another_proc** Active process instance: **a4001bf0...**

^ -

Variables Tokens Subprocesses

```
processInstanceId: 665918
* searchResult:
  searchKey: "application.client.identificationData.personalIdentificationNumber"
  result: null
  tooManyResults: false
  resultsNumber: 0
  tokenId: 665970
  searchValue2: "test test"
  tokenUuid: "b0238259-5784-47bc-ad26-086736079d67"
  webSocketPath: "/ws/updates/process"
  processInstanceId: "a4001bf0-6c87-4197-be13-8111bce14850"
  webSocketAddress: "wss://public.qa.flowxai.dev/a4001bf0-6c87-4197-be13-8111bce14850"
```

! If there is a list of results:

```
"searchKey": "application.client.name"
"result": [
  {
    "processInstanceUUID": "UUID",
    "status": "CREATED",
    "processStartDate": date,
    "data" : {"all data in elastic for that
process"}
  ],
  "tooManyResults": true|false
}
```

NOTE: You will receive up to 50 results - if `tooManyResults` is true.

- Example (dummy values extracted from a process):

Process definition: **search_in_another_proc** Active process instance: **5929721e...** □

Variables Tokens Subprocesses

```
processInstanceId: 665917
searchResult:
  searchKey: "application.client.identificationData.personalIdentificationNumber"
  result:
    0: Object {"processInstanceUUID": "16ce7721-97a0-40be-b88b-61d6e906d208", "state": "FINISHED", "processStartDate": "2022-09-13T13:59:56.131Z", "data": {"application": {"client": "1871201460000", "identificationData": {"personalIdentificationNumber": "11f0503b-fd25-4e38-8744-42cef8838fbc"}, "identificationType": "PERSONAL_ID"}, "process": {"name": "Search in another proc", "version": "1.0", "status": "FINISHED", "start": "2022-09-13T13:59:56.131Z", "end": "2022-09-13T13:59:56.131Z", "duration": "PT0S", "variables": {}, "tokens": [{"id": 665969}], "subprocesses": []}, "id": "5929721e-1a5a-4ebd-87d5-72aab51928f"}, "1: Object {"processInstanceUUID": "c8716ea6-0b3b-448d-a814-c84db6347ff7", "state": "FINISHED", "processStartDate": "2022-09-13T14:28:48.988Z", "data": {"application": {"client": "11f0503b-fd25-4e38-8744-42cef8838fbc", "identificationData": {"personalIdentificationNumber": "11f0503b-fd25-4e38-8744-42cef8838fbc"}, "identificationType": "PERSONAL_ID"}, "process": {"name": "Search in another proc", "version": "1.0", "status": "FINISHED", "start": "2022-09-13T14:28:48.988Z", "end": "2022-09-13T14:28:48.988Z", "duration": "PT0S", "variables": {}, "tokens": [{"id": 665969}], "subprocesses": []}, "id": "5929721e-1a5a-4ebd-87d5-72aab51928f"}, "2: Object {"processInstanceUUID": "11f0503b-fd25-4e38-8744-42cef8838fbc", "state": "FINISHED", "processStartDate": "2022-09-13T14:44:59.061Z", "data": {"application": {"client": "11f0503b-fd25-4e38-8744-42cef8838fbc", "identificationData": {"personalIdentificationNumber": "11f0503b-fd25-4e38-8744-42cef8838fbc"}, "identificationType": "PERSONAL_ID"}, "process": {"name": "Search in another proc", "version": "1.0", "status": "FINISHED", "start": "2022-09-13T14:44:59.061Z", "end": "2022-09-13T14:44:59.061Z", "duration": "PT0S", "variables": {}, "tokens": [{"id": 665969}], "subprocesses": []}, "id": "5929721e-1a5a-4ebd-87d5-72aab51928f"}]
  tooManyResults: false
resultsNumber: 3
redirectUrl: "https://designer.flowxai.dev/processes/instance?processUuid=11f0503b-fd25-4e38-8744-42cef8838fbc"
tokenId: 665969
searchValue2: "1871201460000"
tokenUuid: "a057acf5-36ea-4d10-abbd-35c062398ebc"
webSocketPath: "/ws/updates/process"
processInstanceId: "5929721e-1a5a-4ebd-87d5-72aab51928f"
webSocketAddress: "wss://public.qa.flowxai.dev/5929721e-1a5a-4ebd-87d5-72aab51928f"
```

Let's go now through the steps needed to deploy and set up the service:

» [Search data service setup guide](#)

Was this page helpful?

PLATFORM DEEP DIVE / Core components / Renderer SDKs / Using the Angular Renderer

FlowxProcessRenderer is a low code library designed to render UI configured via the Flowx Process Editor.

Angular project requirements

Your app MUST be created using the NG app from the `@angular/cli~15` package. It also MUST use SCSS for styling.

```
npm install -g @angular/cli@15.0
ng new my-flowx-app
```

INFO

To install the npm libraries provided by FLOWX you will need to obtain access to the private FLOWX Nexus registry. Please consult with your project DevOps.

CAUTION

The library uses Angular version `@angular~15`, `npm v8.1.2` and `node v16.13.2`.

CAUTION

If you are using an older version of Angular (for example, v14), please consult the following link for update instructions:

[Update Angular from v14.0 to v15.0](#)

Installing the library

Use the following command to install the `renderer` library and its required dependencies:

```
npm install @flowx/ui-sdk@3.21.0
@flowx/ui-toolkit@3.21.0
@flowx/ui-theme@3.21.0
paperflow-web-components
vanillajs-datepicker@1.3.1
moment@^2.27.0
@angular/flex-layout@15.0.0-beta.42
@angular/material@15.2.0
@angular/material-moment-adapter@15.2.0
@angular/cdk@15.2.0
ng2-pdfjs-viewer@15.0.0
event-source-polyfill@1.0.31
```

Also, in order to successfully link the pdf viewer, add the following declaration in the assets property of you project's angular.json:

```
{
  "glob": "**/*",
  "input": "node_modules/ng2-pdfjs-viewer/pdfjs",
  "output": "/assets/pdfjs"
}
```

Using the library

Once installed, FlxProcessModule will be imported in the `AppModule`
`FlxProcessModule.forRoot({})`.

You MUST also import the dependencies of `FlxProcessModule`:
`HttpClientModule` from `@angular/common/http` and `IconModule` from
`@flowxai/ui-toolkit`.

Using Paperflow web components

Add path to component styles to stylePreprocessesOptions object in **angular.json file**

```
"stylePreprocessorOptions": {  
  "includePaths": [  
    "./node_modules/paperflow-web-components/src/assets/scss",  
    "./node_modules/flowx-process-  
    renderer/src/assets/scss/style.scss",  
    "src/styles"]  
}
```

(!) INFO

Because the datepicker module is build on top of angular material datepicker module, using it requires importing one predefined material theme in your **angular.json** configuration.

```
"styles": [". . .,  
"./node_modules/@angular/material/prebuilt-themes/indigo-  
pink.css"],
```

Theming

Component theming is done through two json files (`theme_tokens.json`, `theme_components.json`) that need to be added in the assets folder of your project. The file paths need to be passed to the `FlxProcessModule.forRoot()` method through the `themePaths` object.

```
themePaths: {  
    components: 'assets/theme/theme_components.json',  
    tokens: 'assets/theme/theme_tokens.json',  
},
```

The **assets/theme/theme_tokens.json** - should hold the design tokens (e.g. colors, fonts) used in the theme. An example can be found [here](#).

The **assets/theme/theme_components.json** - holds metadata used to describe component styles. An example can be found [here](#).

For **Task Management** theming is done through the ppf-theme mixin that accepts as an argument a list of colors grouped under **primary**, **status** and **background**

```
@use 'ppf-theme';  
  
@include ppf-theme.ppf-theme()  
'primary': (  
    'color1': vars.$primary,  
    'color2': vars.$secondary,  
    'color3': vars.$text-color,  
)  
'status': (  
    'success': vars.$success,  
    'warning': vars.$warning,  
    'error': vars.$error,  
)  
'background': (  
    'background1': vars.$background1,  
    'background2': vars.$background2,  
    'background3': vars.$background3,
```

```
  ),  
));
```

Authorization

ⓘ INFO

Every request from the **FLOWX** renderer SDK will be made using the **HttpClientModule** of the client app, which means those requests will go through every interceptor you define here. This is most important to know when building the auth method as it will be the job of the client app to intercept and decorate the requests with the necessary auth info (eg. `Authorization: Bearer ...`).

ⓘ NOTE

It's the responsibility of the client app to implement the authorization flow (using the **OpenID Connect** standard). The renderer SDK will expect to find the **JWT** saved in the browser **localStorage** object at the key named `access_token`.

```
import {BrowserModule} from '@angular/platform-browser';  
import {NgModule} from '@angular/core';  
import { HttpClientModule, HTTP_INTERCEPTORS } from  
'@angular/common/http';  
import {FlxProcessModule} from 'flowx-process-renderer';  
import {IconModule} from 'paperflow-web-components';  
  
import {AppRoutingModule} from './app-routing.module';  
import {AppComponent} from './app.component';
```

```
@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    // will be used by the renderer SDK to make requests
    HttpClientModule,
    // needed by the renderer SDK
    IonicModule.forRoot(),
    FlxProcessModule.forRoot({
      components: {},
      services: {},
      themePaths: {
        components: 'assets/theme/theme_components.json',
        tokens: 'assets/theme/theme_tokens.json',
      },
    }),
  ],
  // this interceptor will decorate the requests with the
  Authorization header
  providers: [
    { provide: HTTP_INTERCEPTORS, useClass: AuthInterceptor,
    multi: true },
  ],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

The `forRoot()` call is required in the application module where the process will be rendered. The `forRoot()` method accepts a config argument where you can

pass extra config info, register a **custom component**, **service**, or **custom validators**.

Custom components will be referenced by name when creating the template config for a user task.

Custom validators will be referenced by name (`currentOrLastYear`) in the template config panel in the validators section of each generated form field.

```
// example
FlxProcessModule.forRoot({
  components: {
    YourCustomComponentIdentifier: CustomComponentInstance,
  },
  services: {
    NomenclatorService,
    LocalDataStoreService,
  },
  validators: {currentOrLastYear },
})
```

```
# example with custom component and custom validator
FlxProcessModule.forRoot({
  components: {
    YourCustomComponentIdentifier: CustomComponentInstance,
  },
  services: {
    NomenclatorService,
    LocalDataStoreService,
  },
  validators: {currentOrLastYear },
})
```

```
// example of a custom validator that restricts data  
selection to  
// the current or the previous year  
  
currentOrLastYear: function currentOrLastYear(AC:  
AbstractControl): { [key: string]: any } {  
  if (!AC) {  
    return null;  
  }  
  
  const yearDate = moment(AC.value, YEAR_FORMAT, true);  
  const currentDateYear = moment(new  
Date()).startOf('year');  
  const lastYear = moment(new Date()).subtract(1,  
'year').startOf('year');  
  
  if (!yearDate.isSame(currentDateYear) &&  
!yearDate.isSame(lastYear)) {  
    return { currentOrLastYear: true };  
  }  
  
  return null;  
}
```

⚠ CAUTION

The error that the validator returns **MUST** match the validator name.

The component is the main container of the UI, which will build and render the components configured via the **FlowX Designer**. It accepts the following inputs:

```
<flx-process-renderer  
[apiUrl]="baseUrl"  
[processApiPath]="processApiPath"  
[processName]="processName"  
[processStartData]="processStartData"  
[debugLogs]="debugLogs"  
[keepState]="keepState"  
[language]="language"  
></flx-process-renderer>
```

Parameters:

Name	Description	Type	Mandatory	Default value	
baseUrl	Your base url	string	true	-	https
processApiPath	Engine API prefix	string	true	-	/onb
processName	Identifies a process	string	true	-	client
processStartData	Data required to start the process	json	true	-	{ "first": "last"

Name	Description	Type	Mandatory	Default value	
debugLogs	When set to true this will print WS messages in the console	boolean	false	false	-
language	Language used to localize the application.	string	false	ro-RO	-

Name	Description	Type	Mandatory	Default value	
keepState	<p>By default all process data is reset when the process renderer component gets destroyed.</p> <p>Setting this to true will keep process data even if the viewport gets destroyed</p>	boolean	false	false	-

Name	Description	Type	Mandatory	Default value	
isDraft	<p>When true allows starting a process in draft state.</p> <p>*Note that isDraft = true requires that processName be the id (number) of the process and NOT the name.</p>	boolean	false	false	-

Data and actions

Custom components will be hydrated with data through the \$data input observable which must be defined in the custom component class.

```
@Component({
  selector: 'my-custom-component',
  templateUrl: './custom-component.component.html',
  styleUrls: ['./custom-component.component.scss'],
})
```

```
export class CustomComponentComponent {
  @Input() data$: Observable<any>;
}
```

Component actions are always found under `data -> actionsFn` key.

Action names are configurable via the process editor.

```
# data object example
data: {
  actionsFn: {
    action_one: () => void;
    action_two: () => void; }
}
```

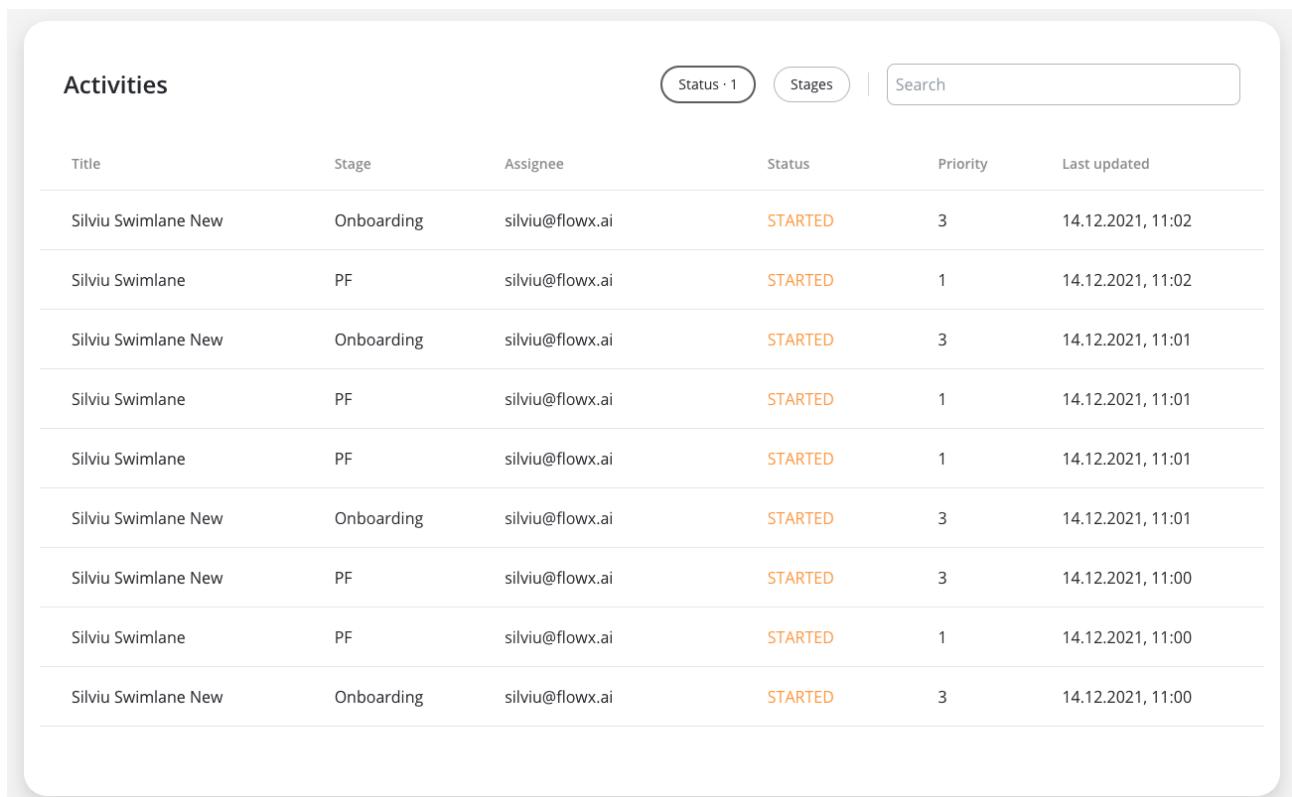
Interacting with the process

Data from the process is communicated via **SSE** protocol under the following keys:

Name	Description	Example
Data	data updates for process model bound to default/custom components	
ProcessMetadata	updates about process metadata, ex: progress update, data about how to render components	

Name	Description	Example	
RunAction	instructs the UI to perform the given action		

Task management component



The screenshot shows a user interface for managing tasks. At the top, there is a header with the title "Activities". Below the header, there are three buttons: "Status · 1", "Stages", and a search bar. The main area contains a table with the following columns: Title, Stage, Assignee, Status, Priority, and Last updated. There are ten rows of data in the table, each representing a task. The tasks are mostly titled "Silviu Swimlane New" or "Silviu Swimlane" and are assigned to "silviu@flowx.ai". Most tasks are in the "Onboarding" stage and have a priority of 3. The last task in the list is in the "PF" stage.

Title	Stage	Assignee	Status	Priority	Last updated
Silviu Swimlane New	Onboarding	silviu@flowx.ai	STARTED	3	14.12.2021, 11:02
Silviu Swimlane	PF	silviu@flowx.ai	STARTED	1	14.12.2021, 11:02
Silviu Swimlane New	Onboarding	silviu@flowx.ai	STARTED	3	14.12.2021, 11:01
Silviu Swimlane	PF	silviu@flowx.ai	STARTED	1	14.12.2021, 11:01
Silviu Swimlane	PF	silviu@flowx.ai	STARTED	1	14.12.2021, 11:01
Silviu Swimlane New	Onboarding	silviu@flowx.ai	STARTED	3	14.12.2021, 11:01
Silviu Swimlane New	PF	silviu@flowx.ai	STARTED	3	14.12.2021, 11:00
Silviu Swimlane	PF	silviu@flowx.ai	STARTED	1	14.12.2021, 11:00
Silviu Swimlane New	Onboarding	silviu@flowx.ai	STARTED	3	14.12.2021, 11:00

The `flx-task-management` component is found in the `FlxTaskManagementModule`. In order to have access to it, import the module where needed:

```
import {FlxTaskManagementModule} from 'flowx-process-renderer';
@NgModule({
  declarations: [
```

```
    ...,
],
imports: [
    ...,
    FlxTaskManagementModule
],
}

export class MyModule {
```

Then in the template:

```
<flx-task-management [baseUrl]="baseUrl" [title]="'Tasks'">
</flx-task-management>
```

Parameters:

Name	Description	Type	Default	Mandatory	
apiUrl	Endpoint where the tasks are available	string	-	true	https://yc
title	Table header value	string	Activities	false	Tasks

Name	Description	Type	Default	Mandatory	
pollingInterval	Interval for polling task updates	number	5000 ms	false	10000

Development

When modifying the library source code and testing it inside the designer app use the following command which rebuilds the flx-process-renderer library, recreates the link between the library and the designer app and recompiles the designer app:

```
npm run build && cd dist/flowx-process-renderer/ && npm link  
&& cd ../../ && npm link flowx-process-renderer && npm run  
start:designer
```

or alternatively run

```
./start_with_build_lib.sh
```

If you want to start the designer app and the flx-process-renderer library in development mode (no need to recompile the lib for every change) run the following command:

```
npm run start:designer-dev
```



CAUTION

Remember to test the final version of the code by building and bundling the renderer library to check that everything works e2e

Trying to use this lib with npm link from another app will most probably fail. If (when) that happens, there are two alternatives that you can use:

1. Use the build-and-sync.sh script, that builds the lib, removes the current build from the client app **node_modules** and copies the newly build lib to the **node_modules** dir of the client app:

```
./build-and-sync.sh ${path to the client app root}

# example (the client app is demo-web):
./build-and-sync.sh ../../demo-web
```

NOTE: This method uses under the hood the build-and-sync.sh script from the first version and the chokidar-cli library to detect file changes.

2. Use the build-and-sync:watch npm script, that builds the library and copies it to the client app's **node_module** directory every time a file changes:

```
npm run build-and-sync:watch --target-path=${path to the
client app root}

# example (the client app is demo-web):
npm run build-and-sync:watch --target-path=../../demo-web
```

Running the tests

```
ng test
```

Coding style tests

Always follow the Angular official [coding styles](#).

Below you will find a Storybook which will demonstrate how components behave under different states, props, and conditions, it allows you to preview and interact with individual UI components in isolation, without the need for a full-fledged application:

» [Storybook](#)

Was this page helpful?

PLATFORM DEEP DIVE / Core components / Renderer SDKs / Using the iOS Renderer

iOS Project Requirements

The minimum requirements are:

- iOS 14
- Swift 5.0

Installing the library

The iOS Renderer is available through Cocoapods and Swift Package Manager.

Swift Package Manager

In Xcode, click `File` → `Add Packages...`, enter FlowX repo's URL `https://github.com/flowx-ai/flowx-ios-sdk`. Set the dependency rule to `Up To Next Major` and add package.

If you are developing a framework and use FlowX as a dependency, add to your `Package.swift` file:

```
dependencies: [
    .package(url: "https://github.com/flowx-ai/flowx-ios-
    sdk", .upToNextMajor(from: "0.96.0"))
]
```

Cocoapods

Prerequisites

- Cocoapods gem installed

Cocoapods private trunk setup

Add the private trunk repo to your local Cocoapods installation with the command:

```
pod repo add flowx-specs git@github.com:flowx-ai/flowx-ios-
specs.git
```

Adding the dependency

Add the source of the private repository in the Podfile

```
source 'git@github.com:flowx-ios-specs.git'
```

Add the pod and then run `pod install`

```
pod 'FlowX'
```

Library dependencies

The iOS Renderer library depends on the following libraries:

- Socket.IO-Client-Swift
- Alamofire
- SVProgressHUD
- SDWebImageSwiftUI

Configuring the library

The SDK has 2 configurations, available through shared instances.

It is recommended to call the configuration methods at app launch.

Otherwise, make sure you do it before the start of any FlowX process.

FXConfig

This config is used for general purpose properties.

Properties

Name	Description	Type	Requirement
baseURL	The base URL used for REST networking	String	Mandatory
imageBaseUrl	The base URL used for media library images	String	Mandatory
language	The language used for retrieving enumerations and substitution tags	String	Mandatory. Defaults to "en"

Name	Description	Type	Requirement
stepViewType	The type of the custom step view class	FXStepViewProtocol.Type	Optional
logEnabled	Value indicating whether console logging is enabled. Default is false	Bool	Optional

Sample

```
FXConfig.sharedInstance.configure { (config) in
    config.baseURL = myDataURL
    config.imageDataURL = myImageDataURL
    config.language = "en"
    config.logEnabled = true
    config.stepViewType = CustomStepView.self
}
```

FXSessionConfig

This config is used for providing networking or auth session-specific properties.

The library expects a session instance managed by the container app. Request adapting and retrying are handled by the container app.

Properties

Name	Description	Type
sessionManager	Alamofire session instance used for REST networking	Session
token	JWT authentication access token	String

Sample

```
FXSessionConfig.sharedInstance.configure { config in
    config.sessionManager = mySessionManager
    config.token = myAccessToken
}
```

Using the library

The library's public APIs are called using the shared instance of FlowX, `FlowX.sharedInstance`.

How to start and end FlowX session

After all the configurations are set, you can start a FlowX session by calling the `startSession()` method.

This is optional, as the session starts lazily when the first process is started.

`FlowX.sharedInstance.startSession()`

When you want to end a FlowX session, you can call the `endSession()` method. This also does a complete clean-up of the started processes.

You might want to use this method in a variety of scenarios, for instance when the user logs out.

`FlowX.sharedInstance.endSession()`

How to start a process

You can start a process by calling the method below.

The container app is responsible with presenting the navigation controller holding the process navigation.

```
public func startProcess(navigationController:  
    UINavigationController,  
        name: String,  
        params: [String: Any]?,  
        isModal: Bool = false,  
        showLoader: Bool = false)
```

`navigationController` - the instance of `UINavigationController` which will hold the process navigation stack

`name` - the name of the process

`params` - the start parameters, if any

`isModal` - a boolean indicating whether the process navigation is modally displayed. When the process navigation is displayed modally, a close bar button item is displayed on each screen displayed throughout the process navigation.

`showLoader` - a boolean indicating whether the loader should be displayed when starting the process.

Sample

```
FlowX.sharedInstance.startProcess(navigationController:  
processNavigationController,  
                                  name: processName,  
                                  params: startParams,  
                                  isModal: true  
                                  showLoader: true)  
  
self.present(processNavigationController, animated: true,  
completion: nil)
```

How to resume a process

You can resume a process by calling the method below.

```
public func continueExistingProcess(uuid: String,  
                                    name: String,  
                                    navigationController:  
UINavigationController,  
                                    isModal: Bool = false) {
```

`uuid` - the UUID string of the process

`name` - the name of the process

`navigationController` - the instance of `UINavigationController` which will hold the process navigation stack

`isModal` - a boolean indicating whether the process navigation is modally displayed. When the process navigation is displayed modally, a close bar button item is displayed on each screen displayed throughout the process navigation.

How to end a process

You can manually end a process by calling the `stopProcess(name: String)` method.

This is useful when you want to explicitly ask the FlowX shared instance to clean up the instance of the process sent as parameter.

For example, it could be used for modally displayed processes that are dismissed by the user, in which case the `dismissRequested(forProcess process: String, navigationController: UINavigationController)` method of the FXDataSource will be called.

Sample

```
FlowX.sharedInstance.stopProcess(name: processName)
```

How to run an action from a custom component

The custom components which the container app provides will contain FlowX actions to be executed. In order to run an action you need to call the following method:

```
public func runAction(action: ProcessActionModel,  
                      params: [String: Any]? = nil)
```

`action` - the `ProcessActionModel` action object

`params` - the parameters for the action

How to run an upload action from a custom component

```
public func runUploadAction(action: ProcessActionModel,  
                           image: UIImage)
```

`action` - the `ProcessActionModel` action object

`image` - the image to upload

```
public func runUploadAction(action: ProcessActionModel,  
                           fileURL: URL)
```

`action` - the `ProcessActionModel` action object

`fileURL` - the local URL of the image

Getting a substitution tag value by key

```
public func getTag(forKey key: String) -> String?
```

All substitution tags will be retrieved by the SDK before starting the first process and will be stored in memory.

Whenever the container app needs a substitution tag value for populating the UI of the custom components, it can request the substitution tag using the method above, providing the key.

Getting a media item url by key

```
public func getMediaItemURL(forKey key: String) -> String?
```

All media items will be retrieved by the SDK before starting the first process and will be stored in memory.

Whenever the container app needs a media item url for populating the UI of the custom components, it can request the url using the method above, providing the key.

```
public func getTag(forKey key: String) -> String?
```

All substitution tags will be retrieved by the SDK before starting the first process and will be stored in memory.

Whenever the container app needs a substitution tag value for populating the UI of the custom components, it can request the substitution tag using the method

above, providing the key.

Handling authorization token changes

When the access token of the auth session changes, you can update it in the renderer using the `func updateAuthorization(token: String)` method.

FXDataSource

The library offers a way of communication with the container app through the `FXDataSource` protocol.

The data source is a public property of FlowX shared instance.

```
public weak var dataSource: FXDataSource?
```

```
public protocol FXDataSource: AnyObject {
    func controllerFor(componentIdentifier: String) -> FXController?

    func viewFor(componentIdentifier: String) -> FXView?

    func viewFor(componentIdentifier: String,
    customComponentViewModel: FXCustomComponentViewModel) ->
    AnyView?

    func navigationController() -> UINavigationController?

    func errorReceivedForAction(name: String?)

    func validate\ValidatorName: String, value: String) ->
    Bool
```

```
        func dismissRequested(forProcess process: String,  
navigationController: UINavigationController)  
  
    }
```

- `func controllerFor(componentIdentifier: String) -> FXController?`

This method is used for providing a custom component UIKit view controller, identified by the componentIdentifier argument.

- `func viewFor(componentIdentifier: String) -> FXView?`

This method is used for providing a custom UIKit view, identified by the componentIdentifier argument.

- `func viewFor(componentIdentifier: String,
customComponentViewModel: FXCustomComponentViewModel) ->
AnyView?`

This method is used for providing a custom SwiftUI view, identified by the componentIdentifier argument. A view model is provided as an ObservableObject to be added as @ObservedObject inside the SwiftUI view.

- `func navigationController() -> UINavigationController?`

This method is used for providing a navigation controller. It can be either a custom `UINavigationController` class, or just a regular `UINavigationController` instance themed by the container app.

- `func errorReceivedForAction(name: String?)`

This method is called when an error occurs after an action is executed.

- `func validate(validatorName: String, value: String) -> Bool`

This method is used for custom validators. It provides the name of the validator and the value to be validated. The method returns a boolean indicating whether the value is valid or not.

- `func dismissRequested(forProcess process: String, navigationController: UINavigationController)`

This method is called, on a modally displayed process navigation, when the user attempts to dismiss the modal navigation. Typically it is used when you want to present a confirmation pop-up.

The container app is responsible with dismissing the UI and calling the stop process APIs.

FXController

FXController is an open class, which helps the container app provide UIKit custom component screens to the renderer. It needs to be subclassed for each custom screen.

```
open class FXController: UIViewController {

    internal(set) public var data: [String: Any]?
    internal(set) public var actions: [ProcessActionModel]?
```

```
open func titleForScreen() -> String? {
    return nil
}

open func populateUI(data: [String: Any]) {

}

open func updateUI(data: [String: Any]) {

}

}
```

- `internal(set) public var data: [String: Any]?`

`data` is a dictionary property, containing the data model for the custom component.

- `internal(set) public var actions: [ProcessActionModel]?`

`actions` is the array of actions provided to the custom component.

- `func titleForScreen() -> String?`

This method is used for setting the screen title. It is called by the renderer when the view controller is displayed.

- `func populateUI(data: [String: Any])`

This method is called by the renderer, after the controller has been presented, when the data is available.

This will happen asynchronously. It is the container app's responsibility to make sure that the initial state of the view controller does not have default/residual values displayed.

- `func updateUI(data: [String: Any])`

This method is called by the renderer when an already displayed view controller needs to update the data shown.

FXView

FXView is a protocol that helps the container app provide custom UIKit subviews of a generated screen to the renderer. It needs to be implemented by `UIView` instances. Similar to `FXController` it has data and actions properties and a populate method.

```
public protocol FXView: UIView {
    var data: [String: Any]? { get set }
    var actions: [ProcessActionModel]? { get set }

    func populateUI(data: [String: Any]?)
```

- `var data: [String: Any]?`

`data` is a dictionary property containing the data model needed by the custom view.

- `var actions: [ProcessActionModel]?`

`actions` is the array of actions provided to the custom view.

- `func populateUI(data: [String: Any]?)`

This method is called by the renderer after the screen containing the view has been displayed.

It is the container app's responsibility to make sure that the initial state of the view does not have default/residual values displayed.

NOTE: It is mandatory for views implementing the FXView protocol to provide the intrinsic content size. Sample:

```
override var intrinsicContentSize: CGSize {  
    return CGSize(width: UIScreen.main.bounds.width, height:  
100)  
}
```

FXCustomComponentViewModel

`FXCustomComponentViewModel` is a class implementing the `ObservableObject` protocol. It is used for managing the state of custom SwiftUI views. It has two published properties, for data and actions.

```
@Published public var data: [String: Any] = [:]  
@Published public var actions: [ProcessActionModel] = []
```

Example

```
struct SampleView: View {  
  
    @ObservedObject var viewModel:
```

```
FXCustomComponentViewModel
```

```
var body: some View {  
    Text("Lorem")  
}  
}
```

Was this page helpful?

PLATFORM DEEP DIVE / Core components / Renderer SDKs / Using the Android Renderer

Android project requirements

To use the Android Renderer library, ensure that your Android project meets the following minimum requirements:

- minSdk 26

Installing the library

1. Add the following code to your Android project's `settings.gradle` file::

```
dependencyResolutionManagement {  
    ...
```

```
repositories {  
    ...  
    maven {  
        credentials {  
            username "YOUR_USERNAME_HERE"  
            password "YOUR_PASSWORD_HERE"  
        }  
        url 'https://nexus-  
jx.dev.rd.flowx.ai/repository/flowx-maven-releases/'  
    }  
}
```

2. Add the following code to your `app/build.gradle` file:

```
dependencies {  
    ...  
    implementation "ai.flowx.android:android-sdk:2.0.1"  
    ...  
}
```

Library dependencies

The Android Renderer library depends on the following libraries:

- Koin
- Retrofit
- Coil

Accessing the documentation

To access the Android Renderer library's documentation, follow these steps:

1. Download the **javadoc.jar** file from the same repository as the library.
2. Extract the **javadoc.jar** file.
3. Open the **index.html** file in your browser.
4. Navigate to `ai.flowx.android.sdk.FlowxSdkApi`.

Was this page helpful?

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.

The screenshot shows the FLOWX.AI platform interface. On the left, there is a sidebar with the following navigation items:

- Processes**
 - Definitions
 - Active process
 - Process Instances
 - Failed process start
- Content Management**
 - Enumerations
 - Substitution tags
 - Content models
 - Languages
 - Source systems
- Plugins**
 - Task Manager
 - All tasks
 - Hooks
 - Stages
 - Allocation rules
 - Out of office
- Notification templates

At the bottom of the sidebar, there is a user profile icon for "John Doe" and a three-dot menu icon.

The main content area is titled "Process Definitions". It features a search bar and a three-dot menu icon. Below the title, there are two sections: "Drafts / In progress" and "Published".

Drafts / In progress

Name	Version	Edited at	Edited by	Actions
s...	1	30 Sep 2022, 5:41 PM	Silviu Grigore	
d...	2	30 Sep 2022, 2:32 PM	andrei antal	
t...	1	30 Sep 2022, 11:15 AM	QA FlowX	
T...	4	30 Sep 2022, 10:20 AM	QA FlowX	

Published

Name	Version	Published at	Published by	Actions
A...	1	03 Oct 2022, 8:12 AM	QA FlowX	
C...	19	30 Sep 2022, 3:08 PM	Silviu Grigore	
d...	1	30 Sep 2022, 10:34 AM	Bogdan Ionescu	
T...	1	30 Sep 2022, 10:18 AM	QA FlowX	

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:

Parameters

Custom From integration

Topics

```
ai.flowx.in.qa.document.html.generate.v1
```

Message

```
1  {
2    "clientType": "PF",
3    "documentList": [
4      {
5        "customId": "123456",
6        "templateName": "test_doc",
7        "language": "en",
8        "data": {
9          },
10       "includeBarcode": true
11     }
12   ]
13 }
```

Advanced configuration

Show Headers

```
1  {"processInstanceId": ${processInstanceId}}
```

- **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:

A screenshot of a software interface titled "Audit log details". The interface displays audit log entries. One entry is expanded, showing detailed information about a process instance message receive event. The expanded entry includes fields for "Event", "Url", and "Body", followed by a JSON representation of the event data.

Audit log details

Event: process instance, message receive, 25 Oct 2022 at 5:57 PM

Url: ai.flowx.updates.qa.document.html.generate.v1

Body:

```
1 {"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","downloadPath":"internal/files/4746/download",  
"noOfPages":1,"error":null}}}, "error":null}
```

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": "R01234567"
  }
}
```

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	Modalitate 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	Modalitate 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'></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"
        ]
    }
}
```

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

Examples



TIP

Download a PDF sample generated based on the HTML example, [here](#).

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Documents plugin / Using the plugin / Uploading a new document

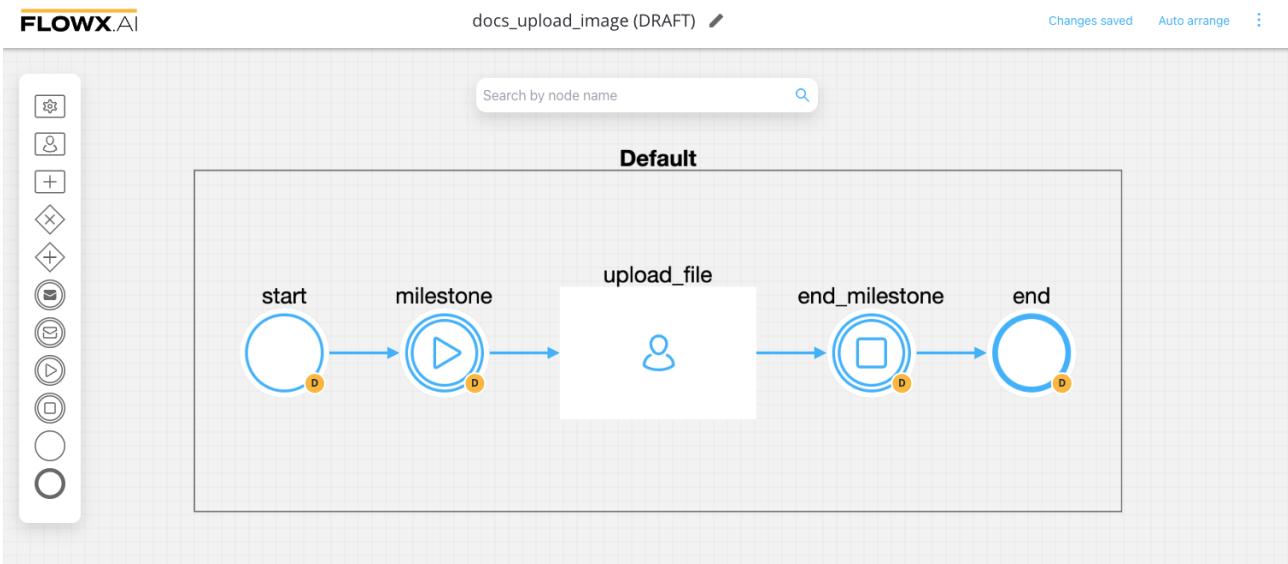
You can integrate document upload into a

The fallback content to display on prerendering by adding a user task node with an **Upload action**. This allows users to interact with the process and choose which file to upload.

(!) INFO

User task

The fallback content to display on prerendering enable you to define and configure UI templates and actions for specific template config nodes, such as an upload file button.



To upload a document using a process, follow the next steps.

Defining the process

1. Create a process definition.
2. Add the necessary nodes, including **start/end nodes**, **start/end milestone nodes**, and a **user task node**.
3. Configure the user task node:
 - Configure the node settings.
 - Configure the upload action, including topics, document type, and folder.

- (UI) Configure the upload button.

Configuring the process definition

User task node

Node Config

- **Swimlane:** Choose a swimlane (if there are multiple swimlanes in the process) to restrict access to specific user roles. If there's only one swimlane, the value is "Default".
- **Stage:** Assign a stage to the node.
- **Topic Name:** Specify the topic name where the process engine listens for the response. This topic should be added to the platform and match the topic naming rule for the engine to listen to it. The default value is `ai.flowx.updates.qa.persist.files.v1`, extracted from `KAFKA_TOPIC_DOCUMENT_PERSIST_IN`.

⚠ CAUTION

A naming pattern must be defined in the

The fallback content to display on prerendering configuration to use the specified topics. It's important to ensure that all events starting with the configured pattern are consumed by the Engine. For example, the `KAFKA_TOPIC_PATTERN` is the topic name pattern where the Engine listens for incoming

The fallback content to display on prerendering events.

- **Key Name:** This key will hold the result received from the external system. If the key already exists in the process values, it will be overwritten.

Node: **upload_file** (ID: 727115)

Node Config	Actions
General Config Node name <input type="text" value="upload_file"/> Can go back? <input checked="" type="checkbox"/>	
Flow Names <input type="text" value="Leave empty if this node is to be included in all flows"/>	
Swimlane <input type="text" value="Default"/>	
<input type="text" value="Stage"/>	
Response Timeout <input type="text" value="Response Timeout (PT30S)"/>	
Data stream topics <input checked="" type="radio"/> Custom <input type="radio"/> From integration <input type="text" value="ai.flowx.updates.qa.persist.files.v1"/> <input type="text" value="files"/> <input type="button" value=""/>	

[Add stream](#)

Task Management

Update task management? 

 Force Task Management Plugin to update information about this process after this node.

Actions

Actions edit

- **Action Type:** Set it to Upload File.
- **Trigger Type:** Choose Manual to allow user-triggered action.
- **Required Type:** Set it as Optional.
- **Reputable:** Check this option if the action can be triggered multiple times.
- **Autorun Children:** When enabled, the child actions defined as mandatory and automatic will run immediately after the parent action is finalized.

Action Edit

ID: 725773

Name

upload_file

Order

1

Timer Expression

Upload File ▼

Automatic Manual

Mandatory Optional

Repeatable

Autorun Children?

Allow BACK on this action?

Parameters

- **Topics:** Set it to `ai.flowx.in.document.persist.v1`, extracted from `KAFKA_TOPIC_DOCUMENT_PERSIST_IN`.
- **Document Type:** Set it to BULK.
- **Folder:** Allows you to configure a value by which the file will be identified in the future.
- **Advanced Configuration (Show Headers):** Represents a JSON value that will be sent in the headers of the Kafka message.

! INFO

Kafka topic names can be customized by overwriting the following environment variables during deployment:

- `KAFKA_TOPIC_DOCUMENT_PERSIST_IN` - default value:
`ai.flowx.in.qa.document.persist.v1`
- `KAFKA_TOPIC_DOCUMENT_PERSIST_OUT` - default value:
`ai.flowx.updates.qa.document.persist.v1`

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

Parameters

Topics

```
ai.flowx.in.qa.document.persist.v1
```

 Replace Values

Document Type

```
BULK
```

 Replace Values

Folder

```
1234_${processInstanceId}
```

 Replace Values

Advanced configuration

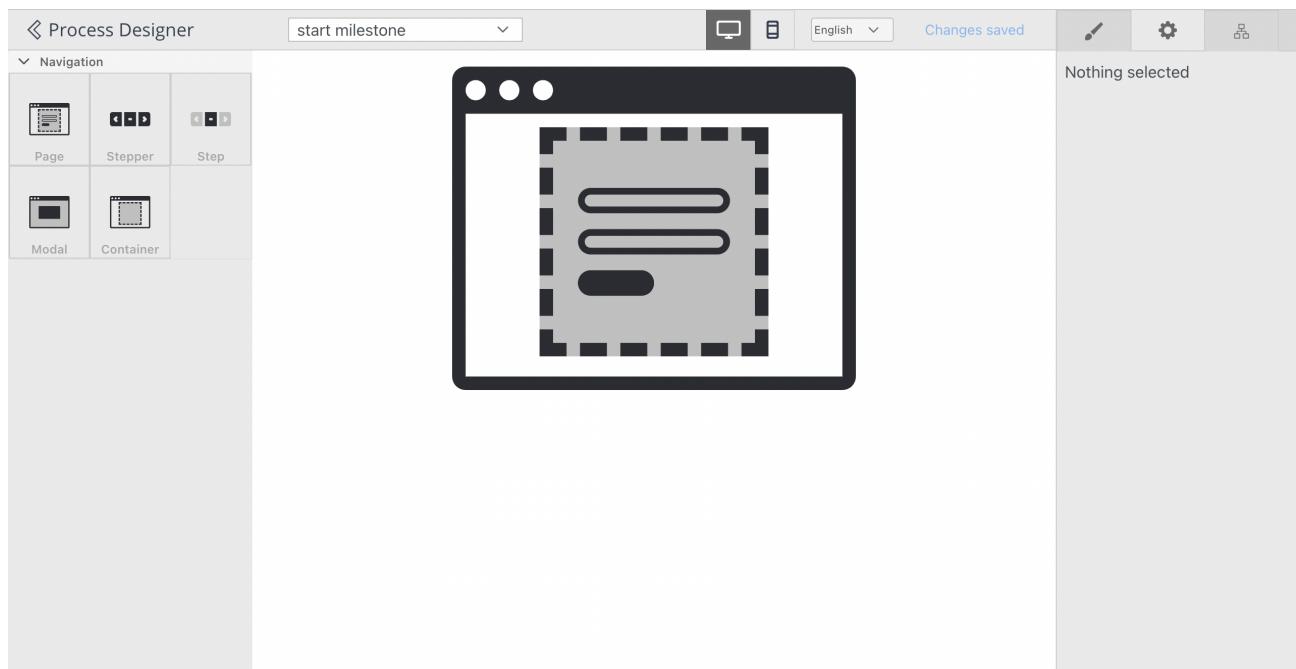
Show Headers

Data to send

Milestone nodes

You can configure start and end milestone nodes before and after the user task. Additionally, you can add a modal template (e.g., a **Page**) to the start milestone

node to display a modal screen, as shown in the example above.



Receiving the reply

The reply body is expected to contain the following values:

- **customId**: The client ID.
- **fileId**: The ID of the file.
- **documentType**: The document type.
- **minioPath**: The path where the uploaded 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 uploaded file. It specifies the location from where the file can be downloaded.
- **noOfPages**: The number of pages in the document.

! INFO

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

Audit log details

Event: process instance, message receive, 13 Oct 2022 at 2:46 PM

Url: ai.flowx.updates.qa.document.persist.v1

Body:

```
1  {"customId": "1234_727605", "fileId": 4718, "documentType": "BULK", "documentLabel": null,
  "minioPath": "bucket-path-qa-process-id-727605/1234_727605/4718_BULK.png",
  "downloadPath": "internal/files/4718/download", "noOfPages": null, "error": null}
```

```
{  
  "customId" : "1234_727605",  
  "fileId" : 4718,  
  "documentType" : "BULK",  
  "documentLabel" : null,  
  "minioPath" : "bucket-path-qa-process-id-
```

```
727605/1234_726254/4718_BULK.png",
  "downloadPath" : "internal/files/4714/download",
  "noOfPages" : null,
  "error" : null
}
```

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Documents plugin / Using the plugin / Converting documents to different formats

⚠ CAUTION

Currently, the supported conversion method is from **PDF** to **JPEG**.

Sending the request

To create a process that converts a document from PDF to JPEG format, follow these steps:

1. Create a process that includes a **Kafka send event** node and a **Kafka receive event** node. The **send node** is used to send the conversion request, and the **receive node** is used to receive the reply.

2. Configure the first node (**Kafka send event**) by adding a **Kafka send action**.

Here is an example:

The screenshot shows the FLOWX.AI platform interface. At the top, it displays the project name "pdf_to_jpeg (DRAFT)" and status indicators for "Changes saved" and "Auto arrange". Below the header is a toolbar with various icons for managing nodes. The main workspace shows a workflow diagram titled "Default" with four states: "start", "convert_request", "convert_reply", and "end". Transitions between these states are labeled "convert_request" and "convert_reply". The "convert_request" state contains a blue icon with a white envelope and a yellow circle containing a lowercase 'd'. The "convert_reply" state also contains a blue icon with a white envelope and a yellow circle containing a lowercase 'd'. The "Actions" tab is selected in the node configuration dialog for the "convert_request" node. The dialog shows the node ID (725652), name (3261a153-b547-49c8-b21d-5cc7e00070d0), order (1), and a dropdown menu set to "Kafka Send Action". It also includes checkboxes for "Automatic" (checked) and "Mandatory" (checked).

Node: **convert_request** (ID: 72562)

Node Config Actions

Actions

Action Edit

ID: 725652

Name
3261a153-b547-49c8-b21d-5cc7e00070d0

Order
1

Timer Expression

Kafka Send Action

Automatic Manual

Mandatory Optional

Save

3. Specify the **Kafka topic** where you want to send the conversion request:

Parameters

Custom

From integration

Topics

```
ai.flowx.in.qa.document.convert.v1
```

4. Fill in the body of the message request:

Message

```
1 { "fileId": 4152, "to": "image/jpeg" }
```

- `fileId`: The file ID that will be converted
- `to`: The file extension to convert to (in this case, "jpeg").

INFO

You can set the Kafka topic names by overwriting the following environment variables during deployment:

- `KAFKA_TOPIC_FILE_CONVERT_IN` - default value:
`ai.flowx.in.qa.document.convert.v1` - the topic that listens for conversion requests from the engine

- `KAFKA_TOPIC_FILE_CONVERT_OUT` - default value:
`ai.flowx.updates.qa.document.convert.v1` - the topic on which the engine expects the reply

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

CAUTION

Make sure to use an outgoing topic name for the reply that matches the pattern configured in the

The fallback content to display on prerendering , as it listens for messages on topics with specific names.

Receiving the reply

INFO

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

The response will be sent to the outgoing Kafka topic (defined on the Kafka receive event node) and can be accessed as follows:

Node: **convert_reply** (ID: 725260)

Node Config

General Config

Node name

convert_reply

Can go back?

Swimlane

Default



Stage



Response Timeout

Response Timeout (PT30S)

Data stream topics

Custom

From integration

Topic Name

ai.flowx.updates.qa.document.convert.v1

Key Name

jpegFiles



Add stream

Values expected in the reply body:

- **customId**: The client ID.
- **fileId**: The file ID.
- **documentType**: The document type.
- **documentLabel**: The document label (if available).
- **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.
- **noOfPages**: The number of pages in the converted file (if available).
- **error**: Any error message in case of an error during the conversion process.

Audit log details

Event: process instance, message receive, 13 Oct 2022 at 4:15 PM

Url: ai.flowx.updates.qa.document.convert.v1

Body:

```
1  {"customId":"1234_727705","fileId":4152,  
 "documentType":"BULK","documentLabel":null,  
 "minioPath":"qualitance-dev-paperflow-qa-process-id-727705  
 /1234_727705/4722_BULK.jpg","downloadPath":"internal/  
 files/4152/download","noOfPages":null,"error":null}
```

cess ir
s

b40-8

b40-8

b40-8

flowx.ai

Process instance

Start

T35btaaz-ab67-4e7a-db40-8

Response:

```
{  
 "customId": "1234_727705",
```

```
"fileId": 4152,  
"documentType": "BULK",  
"documentLabel": null,  
"minioPath": "qualitance-dev-paperflow-qa-process-id-  
727705/1234_727705/4152_BULK.jpg",  
"downloadPath": "internal/files/4152/download",  
"noOfPages": null,  
"error": null  
}
```

Please note that the actual values in the response will depend on the specific conversion request and the document being converted.

[Was this page helpful?](#)

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Documents plugin / Using the plugin / Splitting a document

You can split a document into multiple parts using the Documents Plugin. This feature is useful, for example, when a user uploads a bulk scanned file that needs to be separated into separate files.

Sending the request

To split a document, follow these steps:

1. Create a process and add a **Kafka send event node** and a **Kafka receive event node**. These nodes are used to send the request and receive the reply.
2. Configure the first node, Kafka send event node by adding a **Kafka send action**.

The screenshot shows the FLOWX.AI platform interface. At the top, there's a navigation bar with the FLOWX.AI logo, the project name "split_docs (DRAFT)", and save/arrange buttons. On the left is a sidebar with icons for settings, users, plus, minus, X, plus, minus, and mail. The main area displays a process flow titled "Default" with nodes: start, split_request (highlighted in yellow), split_reply, and end. Below this, a modal window titled "Action Edit" is open for the "split_request" node. The modal has tabs for "Node Config" and "Actions", with "Actions" selected. It shows the node ID (727240) and a collapsible "Actions" section containing "split_request". The "Action Edit" form includes fields for "Name" (split_request), "Order" (1), and a "Timer Expression" field. A dropdown menu shows "Kafka Send Action" is selected. Below the dropdown are two radio buttons: "Automatic" (checked) and "Manual", and two checkboxes: "Mandatory" (checked) and "Optional". A "Save" button is at the bottom right of the modal.

Node: **split_request** (ID: 727240)

Node Config Actions

Actions +
split_request +

Action Edit
ID: 728253
Name: split_request
Order: 1
Timer Expression
Kafka Send Action
Automatic
Mandatory

Save

3. Specify the **Kafka topic** to which you want to send the request.

Parameters

Custom

From integration

Topics

```
ai.flowx.in.document.split.v1
```

4. Fill in the body message request:

Message

```
1  {
2      "fileId":4742,
3      "parts": [
4          {
5              "documentType": "BULK",
6              "customId": "1234_759769",
7              "pagesNo": [1,2]
8          }
9      ]
10 }
```

Advanced configuration

Show Headers

```
1  {"processInstanceId": ${processInstanceId}}
```

- **fileId**: The ID of the file to be split.
- **parts**: A list containing information about the expected document parts.
 - **documentType**: The document type.
 - **customId**: The client ID.

- **shouldOverride**: A boolean value (true or false) indicating whether to override an existing document if one with the same name already exists.
- **pagesNo**: The pages that you want to separate from the document.

INFO

You can customize the Kafka topic names by overwriting the following environment variables during deployment:

`KAFKA_TOPIC_DOCUMENT_SPLIT_IN` - default value:

`ai.flowx.in.qa.document.split.v1` - this is the topic that listens for the request from the engine

`KAFKA_TOPIC_DOCUMENT_SPLIT_OUT` - default value:

`ai.flowx.updates.qa.document.split.v1` - this is 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 this pattern:

`ai.flowx.updates.{{environment}}.document.split.v1`.

CAUTION

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

Receiving the reply

You can view the response by accessing the Audit log menu. The reply will be sent to the Kafka topic specified in the Kafka receive event node.

Node: **split_reply** (ID: 728352)

Node Config

General Config

Node name

Can go back?

Swimlane



Response Timeout

Data stream topics

Topic Name

Key Name



The response body will contain the following values:

- **docs**: A list of documents.
 - **customId**: The client ID.
 - **fileId**: The ID of the file.
 - **documentType**: The document type.
 - **minioPath**: The storage path for the document.
 - **downloadPath**: The download path for the document.
 - **noOfPages**: The number of pages in the document.

The screenshot shows a modal window titled "Audit log details". Inside, the event is described as "process instance, message receive, 25 Oct 2022 at 2:36 PM" with the URL "ai.flowx.updates.qa.document.split.v1". The "Body:" section contains a JSON snippet:

```
1 {"docs": [{"customId": "1234_759769", "fileId": 4743, "documentType": "BULK", "documentLabel": null, "minioPath": "qualitance-dev-paperflow-qa-process-id-759770/1234_759769/4743_BULK.pdf", "downloadPath": "internal/files/4743/download", "noOfPages": 2, "error": null}], "error": null}
```

Here's an example of the response JSON:

```
{  
  "docs": [  
    {
```

```
    "customId": "1234_759769",
    "fileId": 4743,
    "documentType": "BULK",
    "documentLabel": null,
    "minioPath": "qualitance-dev-paperflow-qa-process-id-759770/1234_759769/4743_BULK.pdf",
    "downloadPath": "internal/files/4743/download",
    "noOfPages": 2,
    "error": null
  }
],
"error": null
}
```

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Documents plugin / Using the plugin / Updating and deleting document files

The documents plugin provides functionality for updating and deleting files associated with documents. You can update existing files or remove them from a document.

Updating files

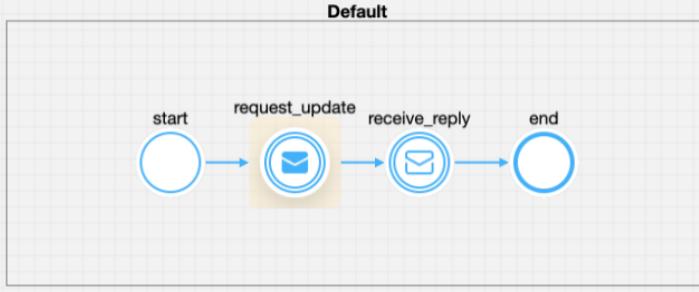
Sending the request

To update files, follow these steps:

1. Create a process and add 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**.

doc_update (DRAFT) 

Changes saved 



Default

start → request_update → receive_reply → end

Node: **request_update** (ID: 763601)

Node Config Actions

Actions +  update_document + 

Action Edit

ID: 762019

Name: update_document

Order: 1

Timer Expression:

Kafka Send Action 

Automatic Manual

Mandatory Optional

Repeatable

3.Specify the **Kafka topic** to send the request to.

Parameters

Custom From integration

Topics

```
ai.flowx.in.qa.document.update.file.v1
```

4. Fill in the body of the request message:

Message

```
1  {
2    "fileId": 4749,
3    "customId": "test_763879"
4 }
```

Advanced configuration

Show Headers

```
1  {"processInstanceId": ${processInstanceId}}
```

- **fileId**: The ID of the file.
- **customId**: The client ID.
- **documentType**: The document type.



INFO

Kafka topic names can be customized by overwriting the following environment variables during deployment:

- `KAFKA_TOPIC_FILE_UPDATE_IN` - default value:
`ai.flowx.in.qa.document.update.file.v1`
- `KAFKA_TOPIC_FILE_UPDATE_OUT` - default value:
`ai.flowx.updates.qa.document.update.file.v1`

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

CAUTION

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

Receiving the reply

Audit log details

Event: process instance, message receive, 26 Oct 2022 at 4:22 PM

Url: ai.flowx.updates.qa.document.update.file.v1

Body:

```
1  {"customId":"test_763879","fileId":4749,  
 "documentType":"BULK","documentLabel":null,  
 "minioPath":"qualitance-dev-paperflow-qa-process-id-763879  
 /test_763879/4749_BULK.pdf","downloadPath":"internal/  
 files/4749/download","noOfPages":null,"error":null}
```

Values expected in the reply body:

- customId = client ID
- fileId = file ID
- documentType = document type
- documentLabel = document label
- minioPath = minio path for the updated file
- downloadPath = download path for the updated file
- error = error description

Example:

```
{  
  "customId": "test_763879",  
  "fileId": 4749,  
  "documentType": "BULK",  
  "documentLabel": null,  
  "minioPath": "qualitance-dev-paperflow-qa-process-id-  
763879/test_763879/4749_BULK.pdf",  
  "downloadPath": "internal/files/4749/download",  
  "noOfPages": null,  
  "error": null  
}
```

Deleting files from a document

Used to delete files after bulk upload.

Sending the request

1. Create a process in which you add a **Kafka send event node** and a **Kafka receive event node** (one to send the request, one to receive the reply).
2. Configure the first node (Kafka send event) - add a **Kafka send action**.

FLOWX.AI delete_doc (DRAFT) Changes saved

```
graph LR; start((start)) --> deleteRequest((delete_request)); deleteRequest --> receiveReply((receive_reply)); receiveReply --> end((end))
```

Default

Node: **delete_request** (ID: 766633)

Node Config	Actions
<input type="button" value="Actions +"/> <input type="button" value="deleteDocument"/> <input type="button" value=""/>	<h3>Action Edit</h3> <p>ID: 765241</p> <p>Name deleteDocument</p> <p>Order 1</p> <p>Timer Expression</p> <p>Kafka Send Action</p> <p><input checked="" type="radio"/> Automatic <input type="radio"/> Manual</p> <p><input checked="" type="checkbox"/> Mandatory <input type="radio"/> Optional</p> <p><input type="checkbox"/> Repeatable</p>

3. Add the **Kafka topic** where to send the request:

Parameters

Custom From integration

Topics

```
ai.flowx.in.qa.document.delete.file.v1
```

4. Fill in the body message request:

Message

```
1  {
2    "customId": "1234_763417",
3    "fileId": 4747,
4    "documentType": "BULK"
5 }
```

Advanced configuration

Show Headers

```
1 {"processInstanceId": ${processInstanceId}}
```

- fileId - the id of the file
- customId - the client ID
- documentType - document type

! INFO

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

`KAFKA_TOPIC_FILE_DELETE_IN` - default value:

`ai.flowx.in.qa.document.delete.file.v1`

`KAFKA_TOPIC_FILE_DELETE_OUT` - default value:

`ai.flowx.updates.document.delete.file.v1`

The Engine is listening for messages on topics with names of a certain pattern, make sure to use an outgoing topic name that matches the pattern configured in the Engine. ...

Receiving the reply

Audit log details

Event: process instance, message receive, 26 Oct 2022 at 12:35 PM

Url: ai.flowx.updates.qa.document.delete.file.v1

Body:

```
1  {"customId":"1234_763417","fileId":4747,"documentType":"BULK",
  "error":null}
```

Values expected in the reply body:

- customId = client ID
- fileId = file ID
- documentType = document type
- error = error description

Example:

```
{  
  "customId": "1234_763417",  
  "fileId": 4747,  
  "documentType": "BULK",  
  "error": null  
}
```

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Documents plugin / Using the plugin / Getting URLs for documents

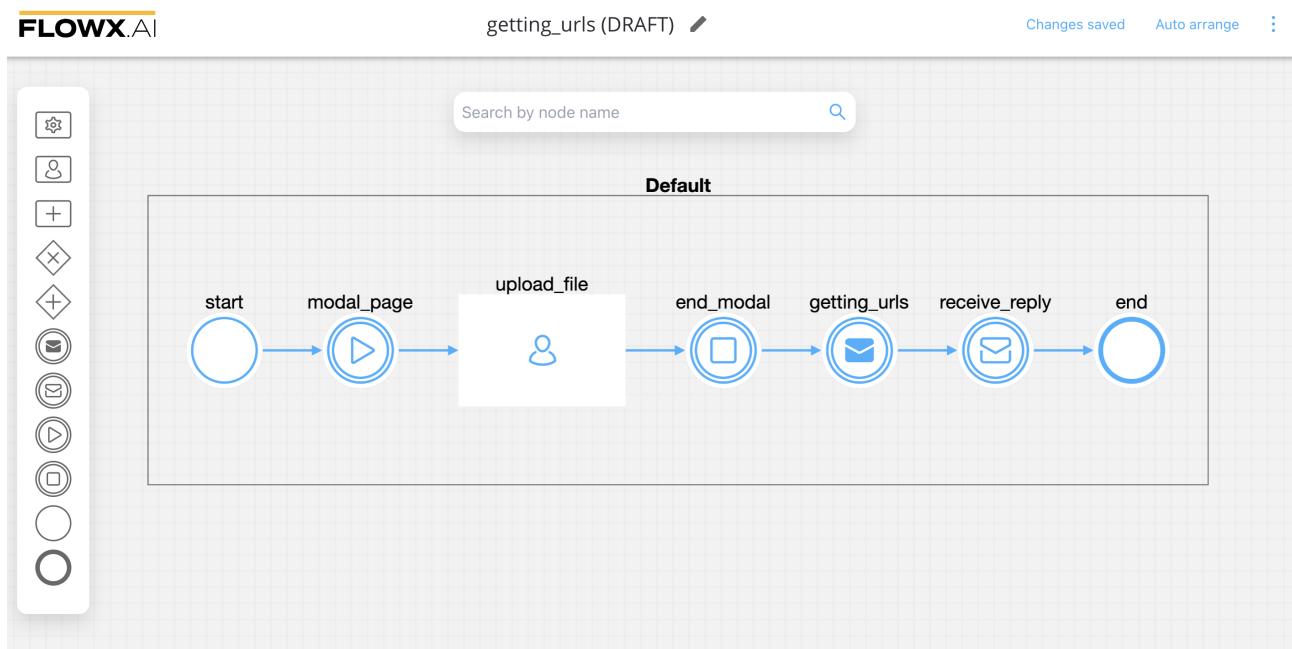
In certain scenarios, you may need to obtain URLs that point to uploaded documents to be used by other integrations. This requires adding a custom action to your process that requests the URLs from the Documents Plugin.

Sending the request

To retrieve document URLs and use them, for example, in the Notification Plugin to attach them to emails, follow the next steps:

1. Create a process and include the following nodes:

- a **Kafka Send Event Node**,
- a **Kafka Receive Event Node**
- a **User Task Node**
- **Start / End <ilestone Nodes** to **create a modal**



2. Configure the **User Task Node** and add an **Upload Action** to it.

Action Edit

ID: 769898

Name

upload_file_action

Order

1

Timer Expression

Upload File



Automatic Manual

Mandatory Optional

Repeatable

Autorun Children?

Allow BACK on this action?

Save

3. Configure the parameters for the **Upload Action**:

Parameters

Topics

```
ai.flowx.in.qa.document.persist.v1
```

 Replace Values

Document Type

```
#{processInstanceId}
```

 Replace Values

Folder

```
1234_#{processInstanceId}
```

 Replace Values

Advanced configuration

Show Headers

```
1 {"processInstanceId": ${processInstanceId}}
```

INFO

For more details on uploading a document and configuring an upload action, refer to the following sections:

Upload document

Upload action

4. Configure the Kafka Send Event Node by adding a **Kafka Send Action** and specifying the **Kafka topic** to send the request to:

Parameters

Custom From integration

Topics

ai.flowx.in.qa.document.urls.v1

5. Fill in the body of the request message for the action:

Message

```
1 ↴ {  
2 ↴   "types": [  
3   |     "${processInstanceId}", "${processInstanceId}"  
4   ]  
5 }
```

Advanced configuration

Show Headers

```
1 {"processInstanceId": ${processInstanceId}}
```

Save

- **types** - a list of document types

6. Configure the **Kafka Receive Event Node** by adding the kafka topic on which the response will be sent.

FLOWX.AI getting_urls (DRAFT) Changes saved Auto arrange ⋮

Default

```
graph LR; end_modal((end_modal)) --> getting_urls((getting_urls)); getting_urls --> receive_reply((receive_reply)); receive_reply --> end((end))
```

Search by node name 🔍

Node: **receive_reply** (ID: 766487) ^ ×

Node Config

Swimlane Default ▼

Stage ▼

Response Timeout

Response Timeout (PT30S)

Data stream topics

Custom From integration

Topic Name	Key Name
ai.flowx.updates.qa.document	receiveReply
Save	

! INFO

Kafka topic names can be set by using environment variables:

- `KAFKA_TOPIC_DOCUMENT_GET_URLS_IN` -
`ai.flowx.in.qa.document.urls.v1` - the topic that listens for the request from the engine
- `KAFKA_TOPIC_DOCUMENT_GET_URLS_OUT` -
`ai.flowx.updates.qa.document.urls.v1` - the topic on which the engine will expect the reply

The example topic names above are from an internal testing environment.

When setting topics for other environments, follow this pattern:

`ai.flowx.updates.{{environment}}.document.urls.v1`.

⚠ CAUTION

The Engine listens for messages on topics with specific naming patterns.

Ensure that your outgoing topic name matches the pattern configured in the Engine.

Receiving the reply

Audit log details

Event: process instance, message receive, 27 Oct 2022 at 12:26 PM

Url: ai.flowx.updates.qa.document.urls.v1

Body:

```
1  [{"success":true,"fullName":"1234_771853/4752_771853.pdf",
  "fileName":"1234_771853/4752_771853","fileExtension":"pdf",
  "url":"http://minio:9000/
  qualitance-dev-paperflow-qa-process-id-771853/1234_771853/
  4752_771853.pdf?X-Amz-Algorithm=AWS4-HMAC-SHA256&
  X-Amz-Credential=minio%2F20221027%2Fus-east-1%2Fs3%2Faws4_r
  equest&X-Amz-Date=20221027T092616Z&X-Amz-Expires=604800&
  X-Amz-SignedHeaders=host&
  X-Amz-Signature=76885166e179263cfabaf00d6cd57ca38d08d31f8f0
  502b3d89d160183c92b56"}]
```

The response body is expected to contain the following values:

```
[  
 {  
   "success": true,
```

```
        "fullName": "1234_771853/4752_771853.pdf",
        "fileName": "1234_771853",
        "fileExtension": "pdf",
        "url": "
<http://SOME_URL/1234_771853/4752_771853.pdf?X-Amz-
Algorithm=SOME_ALGORITHM&X-Amz-Credential=SOME_CREDENTIAL&X-
Amz-Date=20210223T113621Z&X-Amz-Expires=604800&X-Amz-
SignedHeaders=host&X-Amz-Signature=>"  

    }  

]
```

- **success**: A boolean indicating whether the document exists and the URL was generated successfully.
- **fullName**: The full name of the document file, including the directory path.
- **fileName**: The name of the document file without the extension.
- **fileExtension**: The extension of the document file.
- **url**: The full download URL for the document.

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Documents plugin / Using the plugin / Listing stored files

If you are using an S3-compatible cloud storage solution such as [MinIO](#), the stored files are organized into buckets. A bucket serves as a container for objects stored

in Amazon S3. The Documents Plugin provides a REST API that allows you to easily view the files stored in the buckets.

To determine the partitioning strategy used for storing generated documents, you can access the following key in the configuration:

```
application.file-storage.partition-strategy
```

```
application:  
  defaultLocale: en  
  supportedLocales: en, ro  
  jaeger.prefix: document  
  #fileStorageType is the configuration that activates one  
  FileContentService implementation. Valid values: minio /  
  fileSystem  
  file-storage:  
    type: s3  
    disk-directory: MS_SVC_DOCUMENT  
    partition-strategy: NONE
```

The `partition-strategy` property can have two possible values:

- **NONE**: In this case, documents are saved in separate buckets for each process instance, following the previous method. **PROCESS_DATE**: Documents are saved in a single bucket with a subfolder structure based on the process date. For example: `bucket/2022/2022-07-04/process-id-xxxx/customer-id/file.pdf`.

REST API

The Documents Plugin provides the following REST API endpoints for interacting with the stored files:

List buckets

GET `documentURL/internal/storage/buckets`

This endpoint returns a list of available buckets.

List Objects in a Bucket

GET `documentURL/internal/storage/buckets/BUCKET_NAME`

This endpoint retrieves a list of objects stored within a specific bucket. Replace `BUCKET_NAME` with the name of the desired bucket.

Download File

GET `documentURL/internal/storage/download`

This endpoint allows you to download a file by specifying its path or key.

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Notifications plugin

/ Using the plugin / Managing notification templates

You can create and manage notification templates using

The fallback content to display on prerendering web app, by accessing the dedicated section.

Name	Version	Edited at	Edited by
Stunning Template	2	18 Oct 2022, 3:26 PM	John Doe
Amazing Template	1	18 Oct 2022, 3:21 PM	Jane Doe
Exquisite Template	1	18 Oct 2022, 3:18 PM	Bess Twishes

Name	Version	Published at	Published by
Cool Template	21	20 Sep 2022, 4:30 PM	John Doe
Awesome Template	1	03 Aug 2022, 11:48 AM	Jane Doe
Creative Template	1	08 Jun 2022, 11:11 AM	Bess Twishes

Configuring a template

To configure a document template, first, you need to select some information stored in the **Body**:

1. **Type** - could be either MAIL or SMS notifications
2. **Forward on Kafka** - if this checkbox is ticked, the notification is not being sent directly by the plugin to the destination, but forwarded to another adapter (this is mandatory for SMS notifications templates, as they require an external adapter)
3. **Language** - choose the language for your notification template
4. **Subject** - enter a subject

The screenshot shows the FLOWX.AI platform interface for creating a notification template. The left sidebar contains navigation links for Processes, Content Management (with sub-links for Enumerations, Substitution tags, Content models, Languages, and Source systems), Plugins (Task Manager, Hooks, Stages), Notification templates (selected), Document templates, General Settings (Generic Parameters), and Platform status. The top right shows the title "Notifications Template - ExampleTemplate". The main area is divided into "Body" and "Data model" tabs. Under "Body", the "Type" is set to "MAIL", and the "Forward on Kafka" checkbox is unchecked. The "Language" is set to "Romanian (Romania)-ro-RO". The "Subject" field contains the placeholder "Contract [(\${firstInput})]". Below these settings is a rich text editor toolbar with various formatting options like bold, italic, underline, etc. The main content area displays an email template with a header containing the FLOWX.AI logo, a "Salut" greeting, a message body with two input placeholders "#firstInput" and "#secondInput", and a footer with the text "Găsești toate detailele în brosura atașată acestui email." and three small yellow rectangular buttons. At the bottom right of the main area are "Save" and "Publish" buttons.

Editing the content

You can edit the content of a notification template by using the **WYSIWYG** editor embedded in the body of the notification templates body.

Configuring the data model

Using the

The fallback content to display on prerendering

, you can define key pair values (parameters) that will be displayed and reused in the editor. Multiple parameters can be added:

- STRING
- NUMBER
- BOOLEAN
- OBJECT
- ARRAY (which has an additional `item` field)

The screenshot shows a modal window titled "Attributes" for a document template named "ExampleTemplate". The "Data model" tab is selected. The "Attributes" section contains a "Name" field with the value "firstInput" and an "Item type" dropdown set to "STRING". A "Mandatory" toggle switch is turned on. At the bottom are "Close" and "Save" buttons. To the right, a preview pane shows a "New value" input field with the placeholder "New value".

After you defined some parameters in the **Data Model** tab, you can type "#" in the editor to trigger a dropdown where you can choose which one you want to use/reuse.

The screenshot shows the 'Notifications Templates - Test_notification_template' configuration page. On the left, a sidebar lists various system components: Enumerations, Substitution tags, Content models, Languages, Source systems, Plugins (Task Manager, All tasks, Hooks, Stages), Notification templates (selected), Document templates, General Settings (Generic Parameters, Licensing, Access management, Users, Roles, Groups), and a collapsed 'Groups' section. The main area is titled 'Body' and contains fields for 'Type' (set to 'MAIL'), 'Forward on Kafka' (unchecked), 'Language' (set to 'Romanian-ro'), and a subject line ('Romanian-ro subject'). Below these is a WYSIWYG editor toolbar with icons for text, bold, italic, underline, strikethrough, superscript, subscript, and various styling options like alignment, lists, and tables. A 'Source' button is also present.

» [WYSIWYG Editor](#)

Testing the template

You can use the test function to ensure that your template configuration is working as it should before publishing it.

The screenshot shows a web-based editor for creating notification templates. At the top, it says "Notifications Templates - Test_notification_template". Below that, there are two tabs: "Body" (which is selected) and "Data model".

The "Body" tab contains several configuration fields:

- A dropdown menu labeled "Type" set to "MAIL".
- A checkbox labeled "Forward on Kafka".
- A dropdown menu labeled "Language" set to "Romanian-ro".
- A text input field labeled "Romanian-ro subject".
- A "Source" button with a file icon.
- A toolbar with various icons for text styling (bold, italic, underline, etc.) and other document functions.

In the text area below the toolbar, the placeholder text "#test_data" is visible, enclosed in a yellow dashed box.

In the example above, some keys (marked as mandatory) were not used in the template, letting you know that you've missed some important information. After you enter all the mandatory keys, the notification test will go through:



Welcome!

Inbox ×

Notification Test <notification@flowx.ai>

to me ▾

Welcome !

Reply

Forward

Other actions

When opening the contextual menu (accessible by clicking on the breadcrumbs button), you have multiple actions to work with the notifications templates:

- Publish template - publish a template (it will be then displayed in the **Published** tab), you can also clone published templates
- Export template - export a template (JSON format)
- Show history - (version history and last edited)

The screenshot shows a list of notification templates. A context menu is open over the second item in the list, which includes options like Publish template, Export template, Show history, and Delete.

Edited at	Edited by
18 Oct 2022, 3:26 PM	John Doe
18 Oct 2022, 3:21 PM	Jane
Published at	Published by
20 Sep 2022, 4:30 PM	Jane Doe
03 Aug 2022, 11:48 AM	John Doe

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Notifications plugin

/ Using the plugin / Sending a notification

The plugin can be used for sending many kinds of notifications such as emails or SMS notifications. It can be easily integrated in one of your business processes.

Configuring the process

To configure a business process that sends notifications you must follow the next steps:

- use
The fallback content to display on prerendering
web app to create/edit a [notification template](#)
- use
The fallback content to display on prerendering
to add a [Message send task](#) and a [Message received task](#)
- configure the needed [actions](#)
- configure the request body
- configure the needed [Kafka topics](#)

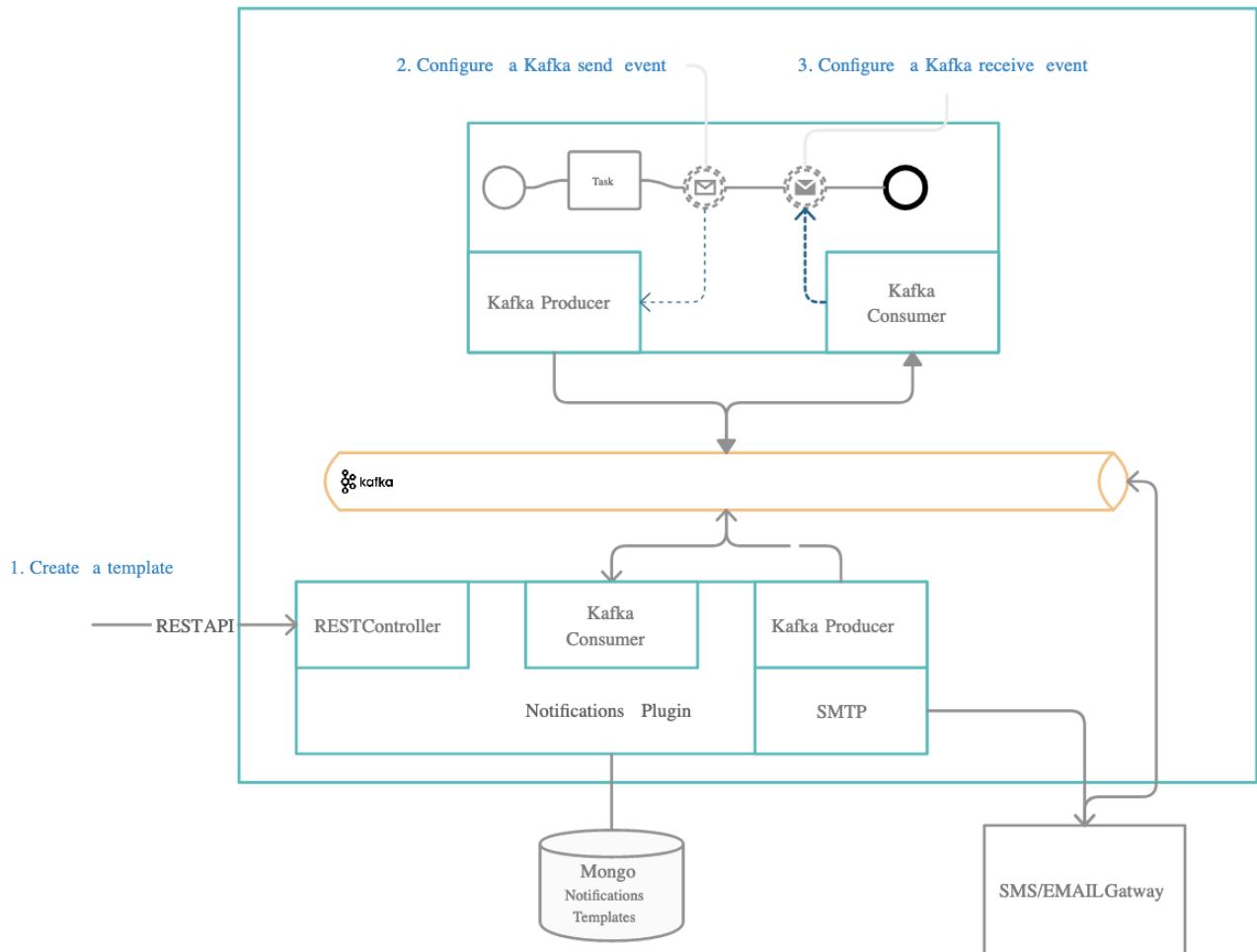
The following values are expected in the request body:

Key	Definition	
language	The language that should be used	Mandatory

Key	Definition	
templateName	The name of the notification template that is used	Mandatory
channel	Notification channel: SMS/MAIL	Mandatory
receivers	Notification receivers: email/phone number	Mandatory
senderEmail	Notification sender email	Optional
senderName	Notification sender name	Optional
attachments	Attachments that are sent with the notification template (only used for MAIL notifications)	Optional

(!) INFO

Check the detailed example below.

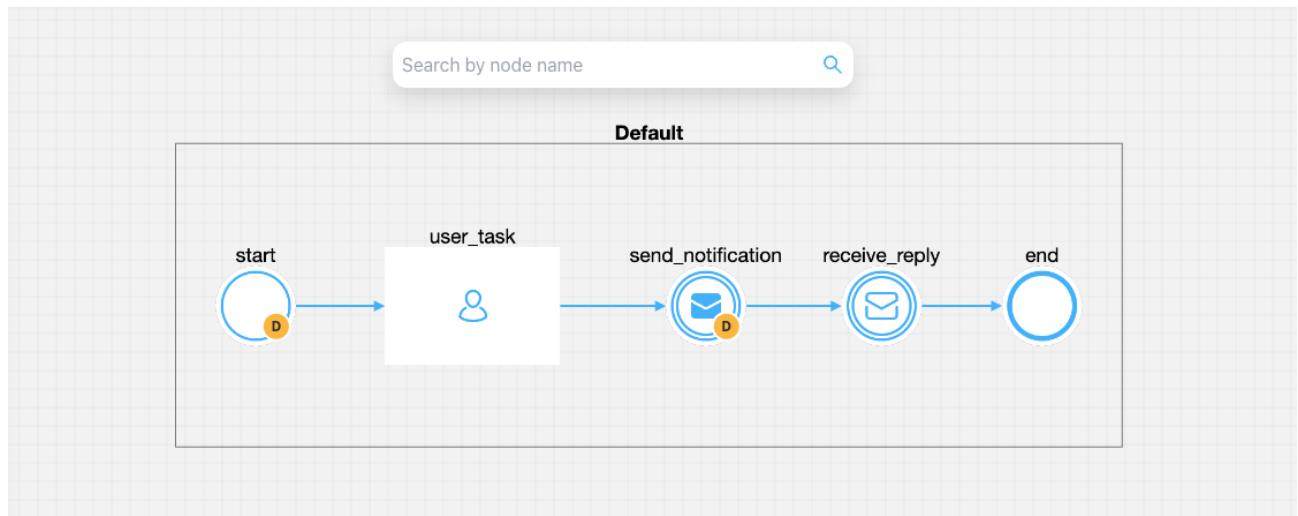


Define needed Kafka topics

Kafka topic names can be set by using environment variables:

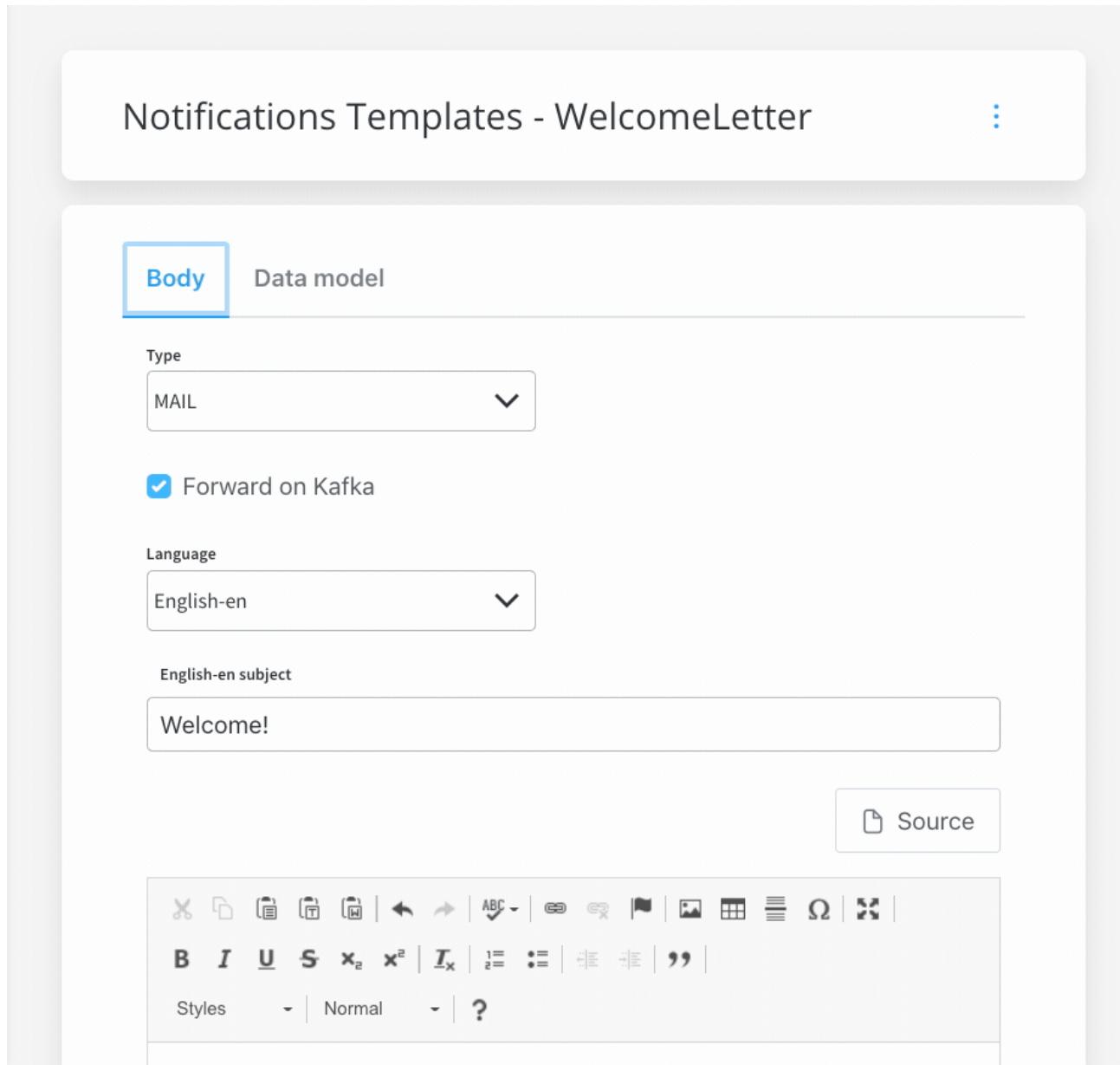
- `KAFKA_TOPIC_NOTIFICATION_INTERNAL_IN` - topic used to trigger the request to send a notification
- `KAFKA_TOPIC_NOTIFICATION_INTERNAL_OUT` - topic used for sending replies after sending the notification

Example: send a notification from a business flow



Let's pick a simple use-case, say we need to send a new welcome letter when we onboard a new customer. The steps are the following:

1. Configure the template that you want to use for the welcome email, see the previous section, [Managing notification templates](#) for more information.



2. Use the FLOWX.AI Designer to add a **Message send task** and a **Message received task**.
3. On the **Message send task** add a proper configuration to the action, the Kafka topic and request body message to be sent:
 - **Topics** - KAFKA_TOPIC_NOTIFICATION_INTERNAL_IN - flowx-notifications-qa

- **Message** (expected parameters):
 - templateName
 - channel
 - language
 - receivers
- **Headers** - it is always `{"processInstanceId": ${processInstanceId}"}`

Kafka Send Action

Automatic Manual

Mandatory Optional

Repeatable

Autorun Children?

Parameters

Custom From integration

Topics

```
flowx-notifications-qa
```

Message

```
1 {  
2   "templateName": "welcomeLetter",  
3   "channel": "MAIL",  
4   "language": "en",  
5   "receivers": ["john_doe@email.com"]  
6 }  
7  
8
```

Advanced configuration

Show Headers

```
1 {"processInstanceId": ${processInstanceId}}
```

Save

4. On the **Message received task** add the needed topic to receive the kafka response - **KAFKA_TOPIC_NOTIFICATION_INTERNAL_OUT** - **ai.flowx.updates.qa.notification.request.v1.**

Node: **generate_notif_receive** (ID: 743211) ▼ X

Node Config

General Config

Node name

generate_notif_receive

Can go back?

Swimlane

Default

Stage

Response Timeout

Response Timeout (PT30S)

Data stream topics

Custom

From integration

Topic Name

ai.flowx.updates.qa.notification.send.v1

Key Name

receiveNotification



Add stream

Save

5. Run the process and look for the response (you can view it via the **Audit log**) or checking the responses on the Kafka topic defined at **KAFKA_TOPIC_NOTIFICATION_INTERNAL_OUT** variable.

Audit log details

Event: process instance, message receive, 19 Oct 2022 at 5:24 PM

Url: ai.flowx.updates.qa.notification.request.v1

Body:

```
1 {"identifier":null,"templateName":"welcomeLetter","language":"en","error":null}
```

dit Log

7b89870... 7442261! 7442261! 7442261! 7442261!

Process Instance Message send 57b89870-dc1c-4e6e-87f0-3ff7442261!

Response example at `KAFKA_TOPIC_NOTIFICATION_INTERNAL_OUT`:

```
{  
  "identifier": null,  
  "templateName": "welcomeLetter",  
  "language": "en",  
  "error": null  
}
```

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Notifications plugin / Using the plugin / Sending an email with attachments

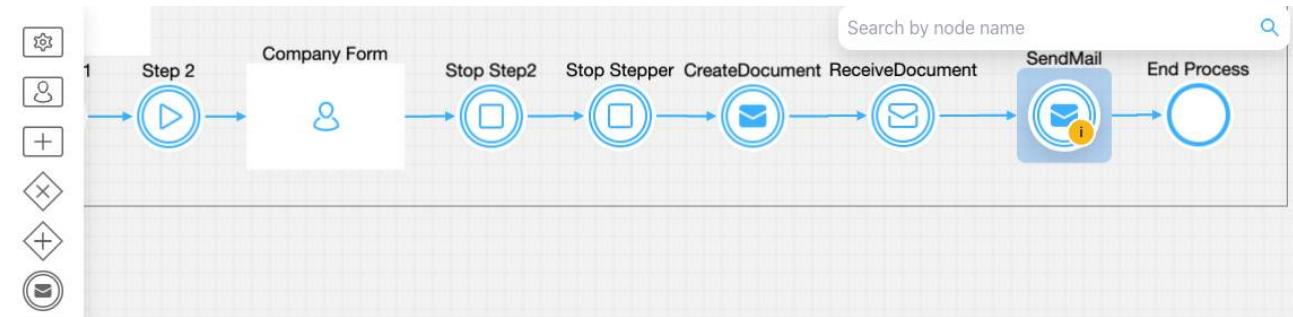
To use the notification plugin for sending emails with attachments, you must define the same topic configuration as for sending regular notifications. A notification template must be created, and the corresponding Kafka topics must be defined.

» [Send a notification](#)

Defining process actions

Example: send an email notification with attached files from a business flow

Let's pick a simple use-case. Imagine we need to send a copy of a contract signed by a new customer. Before setting the action for the notification, another action must be defined, so the first one will save the new contract using the documents plugin.



Node: **SendMail** (ID: 479601)

Node Config	Actions
Actions + sendemail +	Message <pre> 1 { 2 "processInstanceId" : "\${processInstanceId}", 3 "templateName": "Academy_Mail_Template", 4 "channel": "MAIL", 5 "receivers": ["\${application.company.email}"], 6 "language": "en-US", 7 "contentParams": { 8 "firstInput": "\${application.company.id}", 9 "secondInput": "\${application.client.FirstName}" 10 }, 11 "attachments": [12 { 13 "path": "\${generatedDocs.generatedFiles.1234.AcademyTemplate.minioPath}", 14 "filename": "\${generatedDocs.generatedFiles.1234.AcademyTemplate.documentType}" 15 } 16 } </pre>

» Uploading a new document

The steps for sending the notification are the following:

Step 1: Configure the template that you want to use for the email, see the [Managing notification templates](#) section for more information.

Step 2: Check that the needed topics are defined correctly on the following environment variables:

KAFKA_TOPIC_NOTIFICATION_INTERNAL_IN

KAFKA_TOPIC_NOTIFICATION_INTERNAL_OUT

Step 3: Use the

The fallback content to display on prerendering
to add a new **Kafka send event** action to the correct node in the process definition.

Step 4: Add the proper configuration to the action, the Kafka topic and message to be sent.

The message to be sent to Kafka will look something like:

```
{  
  "templateName" : "contractCopy",  
  "identifier" : "text",  
  "language": "en",  
  "receivers" : [ "someone@somewhere.com" ],  
  "contentParams" : {  
    "clientId" : "clientId",  
    "firstName" : "first",  
    "lastName" : "last"  
  },  
  "attachments" : [ {  
    "filename" : "contract",  
    "path" : "MINIO_BUCKET_PATH/contract.pdf"  
  } ]  
}
```

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Notifications plugin / Using the plugin / Forward notifications to an external system

If the Notification service is not directly connected to an SMTP / SMS server and you want to use an external system for sending the notifications, you can use the notification plugin just to forward the notifications to your custom implementation.

Define needed Kafka topics

Kafka topic names can be set by using environment variables:

- `KAFKA_TOPIC_NOTIFICATION_INTERNAL_IN` - topic used to trigger the request to send a notification
- `KAFKA_TOPIC_NOTIFICATION_EXTERNAL_OUT` - the notification will be forwarded on this topic to be handled by an external system
- `KAFKA_TOPIC_NOTIFICATION_INTERNAL_OUT` - topic used for sending replies after sending the notification

Example: send a notification from a business flow

Let's pick a simple use case. Imagine we need to send a new welcome letter when we onboard a new customer. You must follow the next steps:

1. Configure the **template** that you want to use for the welcome email, use the **WYSIWYG Editor**

 CAUTION

Make sure that the **Forward on Kafka** checkbox is ticked, so the notification will be forwarded to an external adapter.

2. Configure the data model for the template.
3. To configure a document template, first, you need to define some information stored in the **Body**:
 - **Type** - MAIL (for email notifications)
 - **! Forward on Kafka** - if this box is checked, the notification is not being sent directly by the plugin to the destination, but forwarded to another adapter
 - **Language** - choose the language for your notification template
 - **Subject** - enter a subject

Notifications Templates - WelcomeLetter

⋮

[Body](#) [Data model](#)

Type

MAIL

Forward on Kafka

Language

English-en

English-en subject

Welcome!

 Source

Hi, #firstName #lastName , this is a notification!

From: #senderName

#senderEmail

body p span

[Save](#)

[Publish](#)

Notifications Templates - WelcomeLetter

Body Data model

Name	Type	Mandatory	Actions
lastName	STRING	false	
firstName	STRING	false	
senderName	STRING	false	
senderEmail	STRING	false	

4. Use the FLOWX.AI Designer to create a process definition.
5. Add a **Kafka send event node** and a **Kafka receive event node** (one to send the request, one to receive the reply).
6. Check if the needed topic (defined at the following environment variable) is configured correctly: `KAFKA_TOPIC_NOTIFICATION_INTERNAL_IN`.
7. Add the proper configuration to the action, the Kafka topic, and the body message.

Kafka Send Action 

Automatic Manual

Mandatory Optional

Repeatable

Autorun Children? 

Parameters

Custom From integration

Topics

flowx-notifications-qa

Message

```
1  {
2    "templateName": "welcomeLetter",
3    "channel": "MAIL",
4    "language": "en",
5    "receivers": ["john_doe@email.com"]
6  }
7
8
```

Advanced configuration

Show Headers 

```
1  {"processInstanceId": ${processInstanceId}}
```

 Save

INFO

Forward on Kafka option will forward the notification to an external adapter, make sure the needed Kafka topic for forwarding is defined/overwritten using the following environment variable: `KAFKA_TOPIC_EXTERNAL_OUT`.

- Run the process and look for the response (you can view it via the **Audit log**) or by checking the responses on the Kafka topic

The screenshot shows a modal window titled "Audit log details". Inside, there's a section for an "Event: process instance, message receive, 19 Oct 2022 at 5:24 PM" and a URL "Url: ai.flowx.updates.qa.notification.request.v1". Below this, under "Body:", a JSON object is displayed:

```
1  {"identifier":null,"templateName":"welcomeLetter","language":"en","error":null}
```

At the bottom of the modal, there are three buttons: "Process Instance", "Message send", and a button with the ID "57b89870-dc1c-4e6e-87f0-3ff7442261". The background of the main interface shows a list of audit logs with IDs like 7b89870..., 7442261..., and 7442261....

Response example at `KAFKA_TOPIC_NOTIFICATION_INTERNAL_OUT`:

```
{  
  "templateName": "welcomeLetter",  
  "receivers": [
```

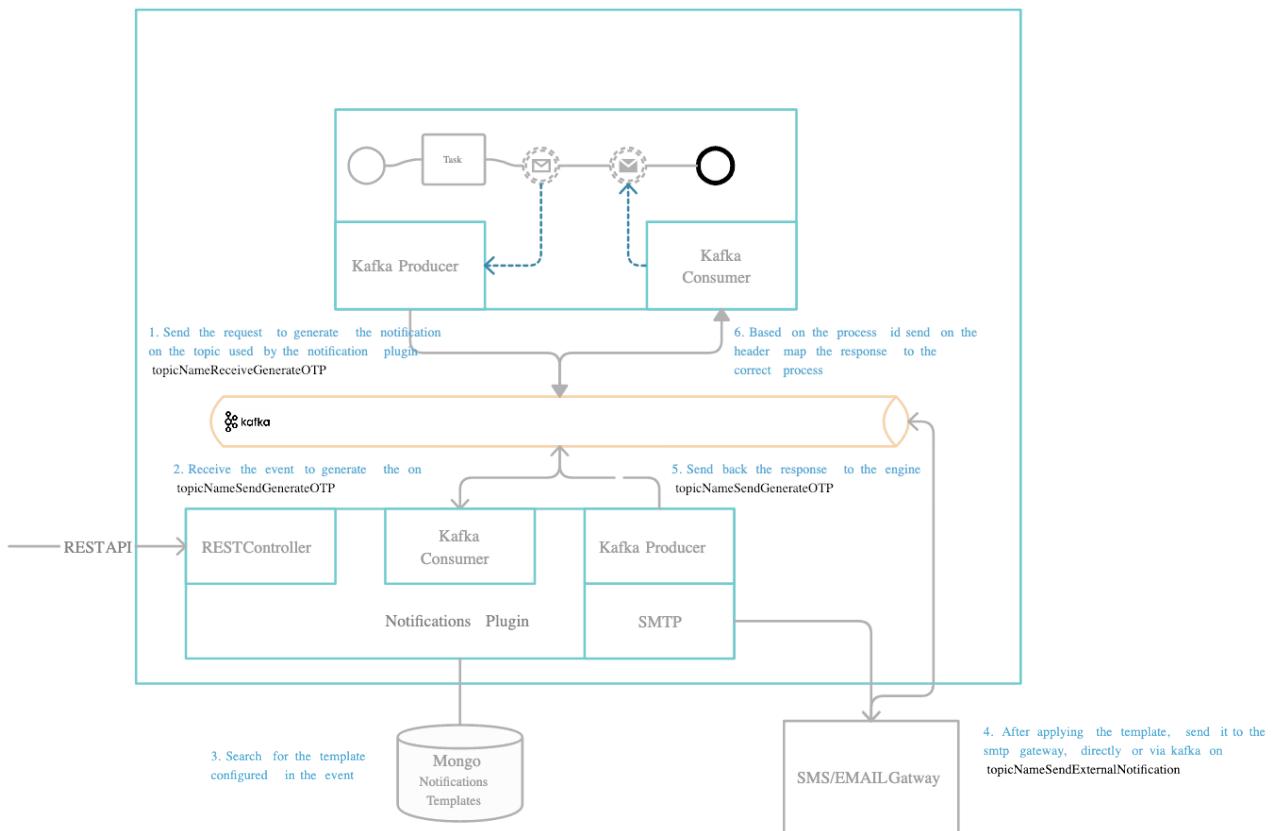
```
    "john.doe@mail.com"  
],  
"channel": "MAIL",  
"language": "en"  
}
```

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Notifications plugin / Using the plugin / OTP flow / Generate OTP

There are some cases when you will need to generate an OTP (One Time Password) from a business flow, for example when validating an email account.

The notifications plugin handles both the actual OTP code generation and sending the code to the user using a defined [notification template](#).



Define needed Kafka topics

Kafka topic names can be set by using environment variables:

- `KAFKA_TOPIC_OTP_GENERATE_IN`
- `KAFKA_TOPIC_OTP_GENERATE_OUT` - after the OTP is generated and sent to the user, this is the topic used to send the response back to the Engine.

⚠ CAUTION

The Engine is listening for messages on topics with names of a certain pattern, make sure to use an outgoing topic name that matches the pattern configured in the Engine.

Request to generate an OTP

Values expected in the request body:

- `templateName`: the name of the notification template that is used (created using the [WYSIWYG editor](#))
- `channel`: notification channel: SMS / MAIL
- `recipient`: notification receiver: email / phone number
- notification template content parameters (for example, `clientId`): parameters that should be replaced in the [notification template](#)

Parameters

Custom From integration

Topics

```
ro.flowx.qa.notification.otp.in.generate.v1
```

Message

```
1  {
2      "templateName": "otpMail",
3      "channel" : "MAIL",
4      "language": "en-US",
5      "clientId" : "${application.client.identificationData.personalIdentificationNumber}",
6      "recipient" : "${application.client.contactData.email.emailAddress}",
7      "contentParams":
8          {
9              "clientId": "${application.client.identificationData.personalIdentificationNumber}"
10         }
11     }
12 }
13 }
```

Advanced configuration

Show Headers

```
1  {"processInstanceId": ${processInstanceId}}
```

Response from generate OTP

Values expected in the reply body:

- processInstanceId = process instance ID
- clientId = the client id (in this case the SSN number of the client)
- channel = notification channel used
- otpSent = confirmation if the notification was sent: true or false

- error = error description, if any

Example:

The screenshot shows a modal window titled "Audit log details". Inside, it displays the following information:

Event: process instance, message receive, 18 Oct 2022 at 4:12 PM

Url: ai.flowx.updates.qa.notification.otp.generate.v1

Body:

```
1 {"processInstanceId":739452,"clientId":"1871201460101",
  "channel":"MAIL","otpSent":true,"error":null}
```

The modal has a close button (X) in the top right corner. The background of the application shows a sidebar with "Active" and "Audit" tabs, and several other audit logs listed below.

flowx.ai

Process Instance

Execute action

action25

Example: generate an OTP from a business flow

It is important to identify what is the business identifier that you are going to use to validate that OTP, it can be, for example, a user identification number.

1. Configure the templates that you want to use (for example, an SMS template).
2. Check that the needed topics are configured correctly: the topic used to generate OTP (`KAFKA_TOPIC_OTP_GENERATE_IN`) and the topic used to receive the response (`KAFKA_TOPIC_OTP_GENERATE_OUT`).
3. Use the FLOWX.AI Designer to add a new Kafka send event to the correct node in the process definition.
4. Add the proper configuration to the action, the Kafka topic, and configure the body message.

Kafka Send Action

 Automatic Manual Mandatory Optional RepeatableAutorun Children? Allow BACK on this action?

Parameters

Topics

ai.flowx.in.qa.notification.otp.generate.v1

Message

```
1  {
2    "templateName": "otpSMS",
3    "language": "en-US",
4    "channel": "SMS",
5    "clientId": "${application.client.identificationData.personalIdentificationNumber}",
6    "recipient": "${application.client.contactData.mobilePhone.phoneNumber}"
7 }
```

Advanced configuration

Show Headers

```
1 {"processInstanceId" : ${processInstanceId}, "destinationId": "register_contact_data", "callbacksForAction": "action21" }
```


5. Add a node to the process definition (for the Kafka receive event).
6. Configure on what key you want to receive the response on the process instance params.

Topic Name	Key Name
ro.flowx.updates.qa.notification.otp.generate.v1	otpCheck

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Notifications plugin / Using the plugin / OTP flow / Validate OTP

Define needed Kafka topics

Kafka topic names can be set by using environment variables:

- `KAFKA_TOPIC_OTP_VALIDATE_IN` - the event sent on this topic (with an OTP and an identifier) will check if the OTP is valid
- `KAFKA_TOPIC_OTP_VALIDATE_OUT` - the response for this request will validate an OTP, the reply is sent back to the Engine on this topic

CAUTION

The Engine is listening for messages on topics with names of a certain pattern, make sure to use an outgoing topic name that matches the pattern configured in the Engine.

Request to validate an OTP

Values expected in the request body:

- `processInstanceId` = process instance ID

- client id = the user unique ID in the system
- channel = notification channel: SMS/MAIL
- otp = OTP code that you received, used to compare with the one that was sent from the system

Example:

```
{  
    "processInstanceId": 12345,  
    "clientId": "1871201460101",  
    "channel": "MAIL",  
    "otp": "1111"  
}
```

Reply from validate OTP

Values expected in the reply body:

- client id = the user unique id in the system
- channel = notification channel used
- otpValid = confirmation if the provided OTP code was the same as the one sent from the system

Example:

The screenshot shows a modal window titled "Audit log details" with a close button (X). Inside, there's a section for "Event" (process instance, message receive, 18 Oct 2022 at 5:33 PM) and "Url" (ai.flowx.updates.qa.notification.otp.validate.v1). Below that is a "Body:" section containing a JSON object:

```
1  {"processInstanceId":0,"clientId":"1871201460101",
  "channel":"MAIL","otpValid":true}
```

Example: validate an OTP from a business flow

Similar to the generation of the OTP you can validate the OTP that was generated for an identifier.

1. Check that the needed topics are configured correctly:
(`KAFKA_TOPIC OTP_VALIDATE_IN` and
`KAFKA_TOPIC OTP_VALIDATE_OUT`)
2. Add the actions for sending the request to validate the OTP on the node that contains the 'Generate OTP' actions
3. Add the proper configuration to the action, the Kafka topic and configure the body message.

Parameters

Custom From integration

Topics

```
ai.flowx.in.qa.notification.otp.validate.v1
```

Message

```
1  {
2    "clientId" : "${application.client.identificationData.personalIdentificationNumber}",
3    "channel" : "MAIL",
4    "otp" : "${email.otpValue}"
5 }
```

Advanced configuration

Show Headers

Data to send

[Add Key](#)

[Save](#)

4. Add a node to the process definition (for the **Kafka receive event**)
5. Configure on what key you want to receive the response on the process instance parameters

Node: **register_contact_data** (ID: 735010)

Node Config

Actions

Custom

From integration

Topic Name

Key Name

ai.flowx.updates.qa.notification.otp.valida

otpValid

Task Management

Update task management?



ⓘ Force Task Management Plugin to update information about this process after this node.

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Task management / Using allocation rules

Allocation rules are meant to define when tasks should be auto-assigned to users when they reach a **swimlane** that has a specific role configured (for example, specific tasks will be assigned for the *front office* and specific tasks for the *back office* only).

Allocation rules				
<input type="text" value="Search by process name"/>				
Add process :				
Process				
Swimlanes Process + :				
Swimlane	Allocation	Edited at	Edited by	
BACKOFFICE	backoffice@email.com	09 Aug 2022, 10:30 AM	John Doe	
OFFICEADMIN	admin@email.com	03 Aug 2022, 7:00 PM	John Doe	
SUPERVISOR	same as execution rights	03 Aug 2022, 9:16 AM	John Doe	
FRONTOFFICE	frontoffice@email.com	03 Aug 2022, 9:16 AM	John Doe	
Parallel Gateways Process + :				

INFO

Tasks will always be allocated depending on the users load (number of tasks) from current/other processes. If there are two or more users with the same number of assigned tasks, the task will be randomly assigned to one of them.

Accessing allocation rules

To access the allocation rules, follow the next steps:

1. Open

The fallback content to display on prerendering

2. From the side menu, under **Task Management**, select the **Allocation rules** entry.

Swimlane	Allocation	Edited at	Edited by
OFFICEADMIN	o...	03 Aug 2022, 7:00 PM	John Doe
SUPERVISOR	s...	03 Aug 2022, 9:16 AM	John Doe
BACKOFFICE	b...	03 Aug 2022, 9:16 AM	John Doe
FRONTOFFICE	f...	03 Aug 2022, 9:16 AM	John Doe

Adding process and allocation rules

To add process and allocation rules, follow the next steps:

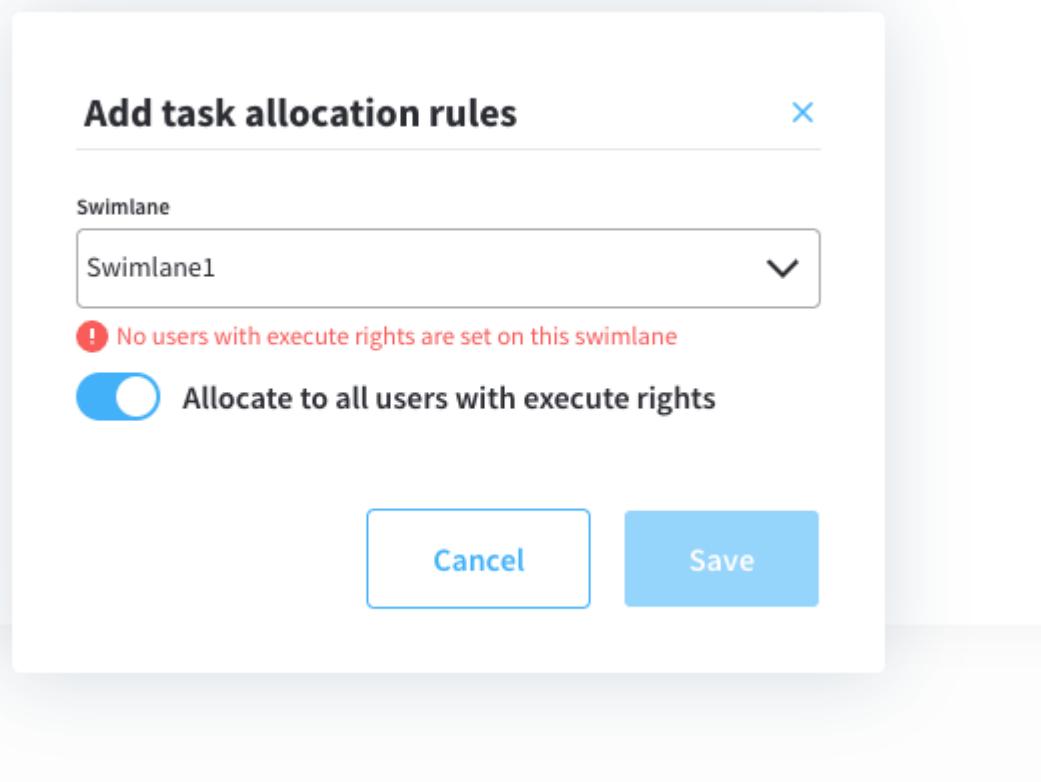
1. Click **Add process** button, in the top-right corner. More details on how to create/configure a process are [here](#).

2. Select a **process definition** from the drop-down list.
3. Click **Add swimlane allocations button (+)** to add allocations.

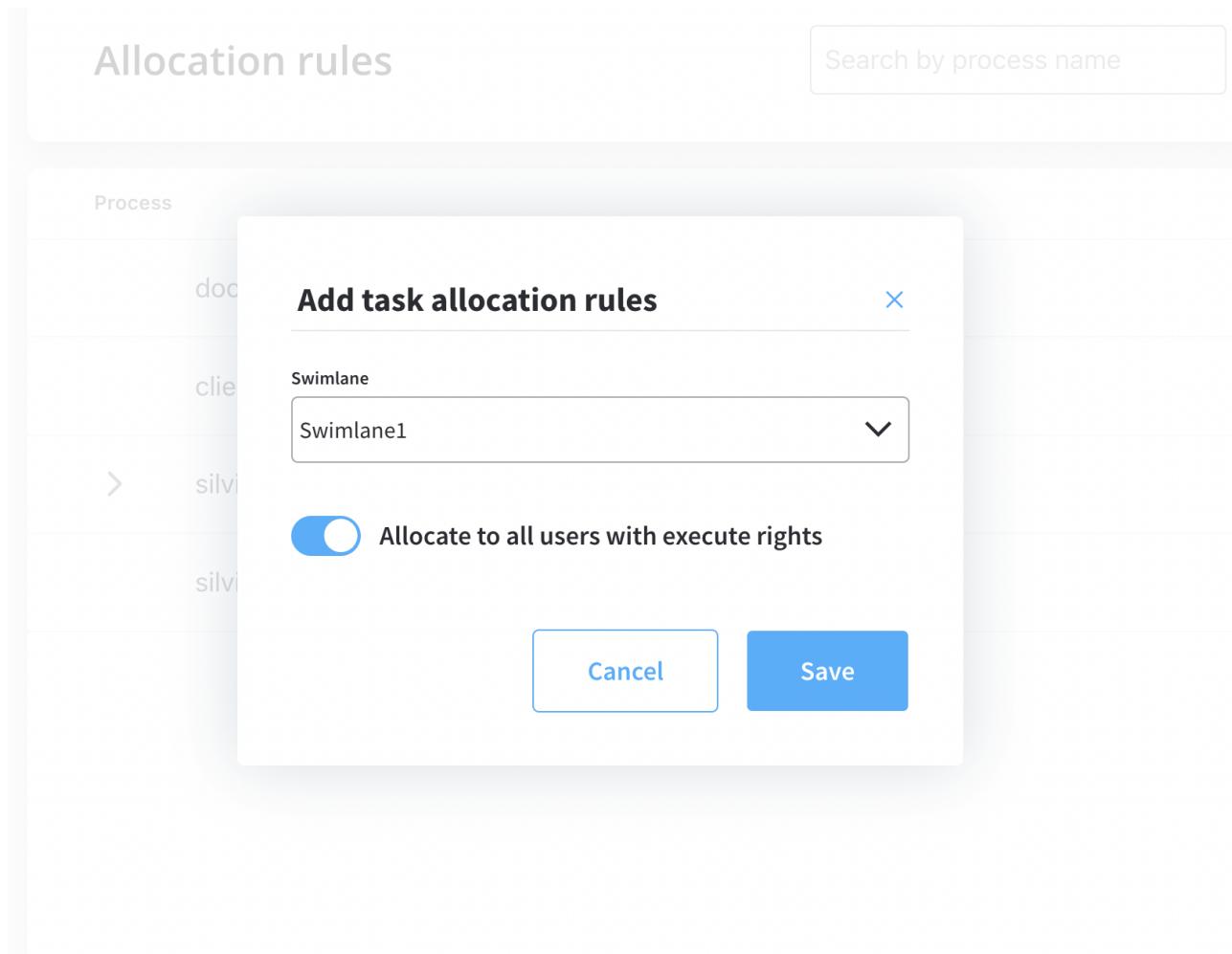
Process			
Swimlane	Allocation	Edited at	Add swimlane allocations
BACKOFFICE	09 Aug 2022, 10:30 AM	John Doe	
OFFICEADMIN	03 Aug 2022, 7:00 PM	John Doe	
SUPERVISOR	03 Aug 2022, 9:16 AM	John Doe	
FRONTOFFICE	03 Aug 2022, 9:16 AM	John Doe	

⚠ CAUTION

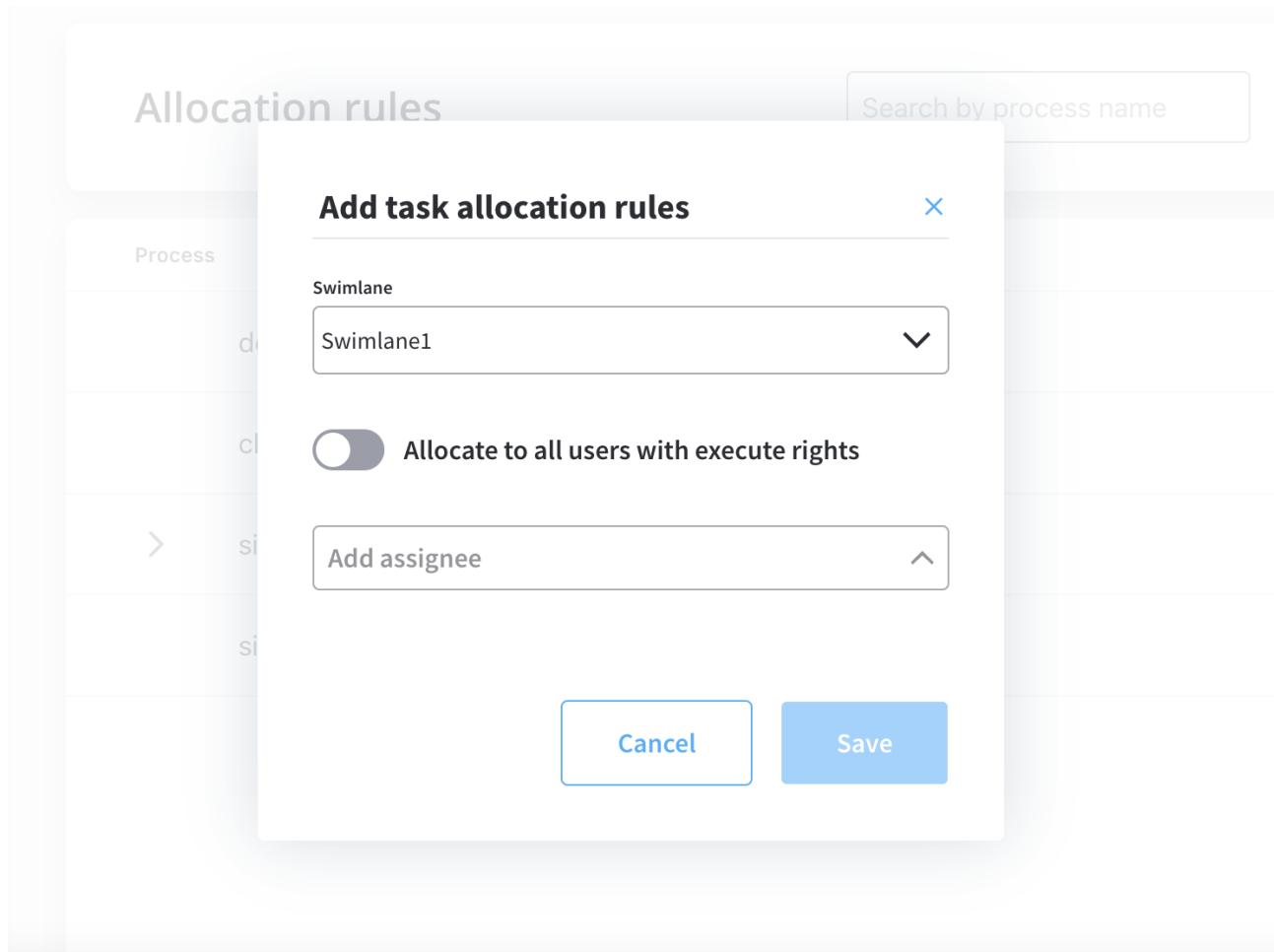
NOTE! If there are no users with execute rights in the swimlane you want to add (`hasExecute: false`), the following error message will be displayed:



4. Option 1: Allocate all users with execute rights.



5. **Option 2:** Allocate only users you choose from the drop-down list. You can use the search function to filter users by name.



6. Click **Save**.

!(INFO)

Users with out-of-office status will be skipped by automatic allocation. More information about out-of-office feature, [here](#).

Editing allocation rules

To edit allocation rules, follow the next steps:

1. Click **Edit** button.

The screenshot shows the 'Allocation rules' section of the Flowx platform. At the top, there is a search bar labeled 'Search by process name', a 'Add process' button, and a more options button. Below this, a table lists processes:

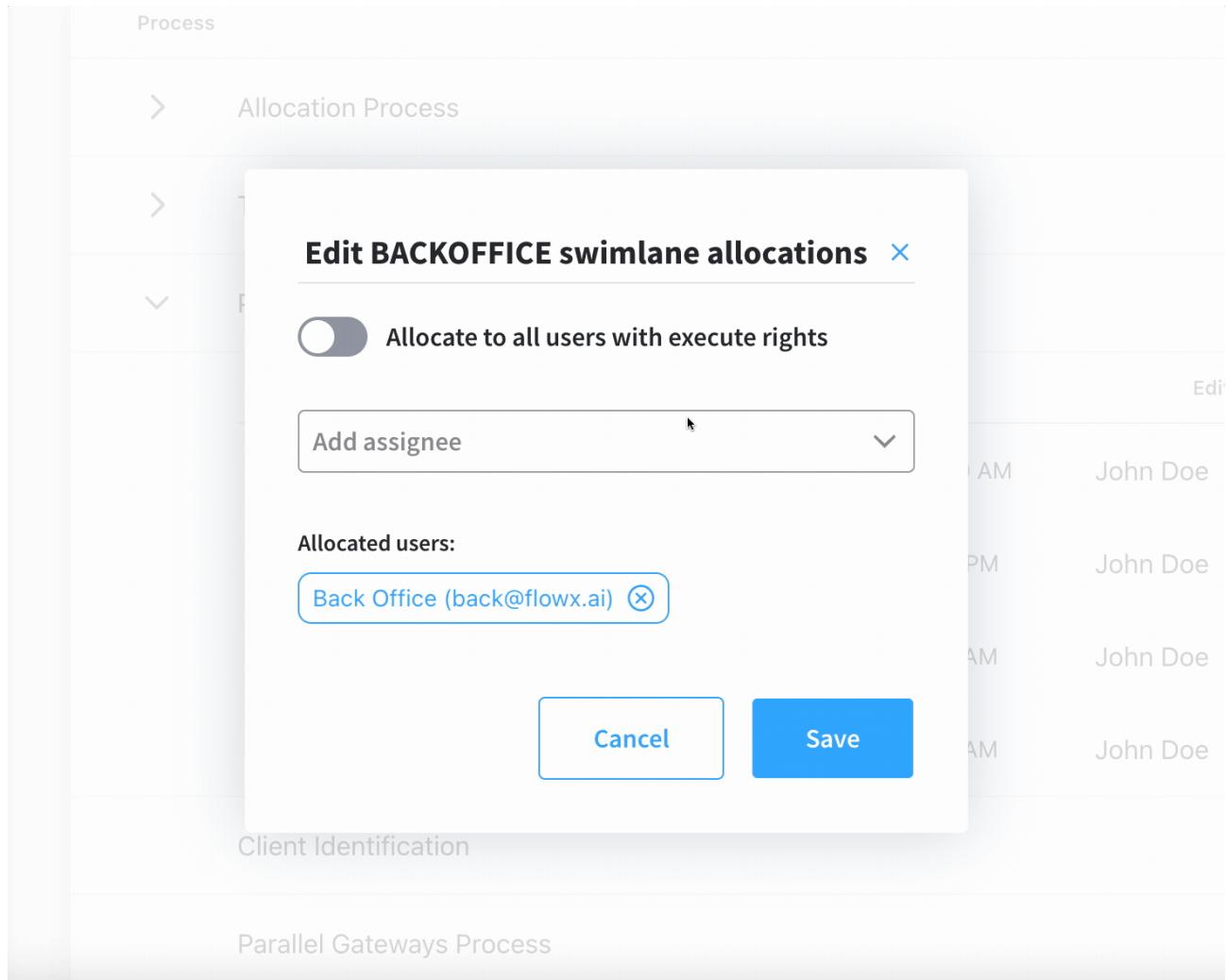
Process	Allocation	Edited at	Edited by	Action
> Allocation Process				+ ::
> Test Process				+ ::
Parallel Swimlanes Process				+ ::

Under the 'Parallel Swimlanes Process' row, a detailed view of allocations is shown:

Swimlane	Allocation	Edited at	Edited by	Action
BACKOFFICE		09 Aug 2022, 10:30 AM	John Doe	Edit
OFFICEADMIN		03 Aug 2022, 7:00 PM	John Doe	Edit
SUPERVISOR		03 Aug 2022, 9:16 AM	John Doe	Edit
FRONTOFFICE		03 Aug 2022, 9:16 AM	John Doe	Edit

The 'Edit' button for the OFFICEADMIN allocation is highlighted with a red box.

2. Change the allocation method.



3. Click **Save**.

Viewing allocation rules

The allocation rules list displays all the configured swimlanes grouped by process:

1. **Process** - the process definition name where the swimlanes were configured
2. **Swimlane** - the name of the swimlane
3. **Allocation** - applied allocation rules
4. **Edited at** - the last time when an allocation was edited

5. Edited by - the user who edited/created the allocation rules

The screenshot shows a table with the following columns:

Process	Swimlane	Allocation	Edited at	Edited by	Action
Swimlanes Process	BACKOFFICE	backoffice@email.com	09 Aug 2022, 10:30 AM	John Doe	
	OFFICEADMIN	admin@email.com	03 Aug 2022, 7:00 PM	John Doe	
	SUPERVISOR	same as execution rights	03 Aug 2022, 9:16 AM	John Doe	
	FRONTOFFICE	frontoffice@email.com	03 Aug 2022, 9:16 AM	John Doe	

Exporting/importing process allocation rules

To copy process allocation rules and move them between different environments, you can use the export/import feature.

You can export process allocation rules as JSON files directly from the allocation rules list:

The screenshot shows the FLOWX.AI platform interface. On the left, there is a sidebar with the following navigation items:

- Processes
 - Definitions
 - Active process
 - Process Instances
 - Failed process start
- Content Management
 - Enumerations
 - Substitution tags
 - Content models
 - Languages
 - Source systems
- Plugins
- Task Manager

At the bottom of the sidebar, there is a user profile for "John Doe" with a yellow icon.

The main content area is titled "Allocation rules". It features a search bar and buttons for "Add process" and more options. The table displays allocation rules for two processes: "Swimlanes Process" and "Parallel Gateways Process".

Process	Swimlane	Allocation	Edited at	Edited by	Actions
Swimlanes Process	BACKOFFICE	back...	09 Aug 2022, 10:30 AM	John Doe	
	OFFICEADMIN	admi...	03 Aug 2022, 7:00 PM	John Doe	
	SUPERVISOR	sam...	03 Aug 2022, 9:16 AM	John Doe	
	FRONTOFFICE	front...	03 Aug 2022, 9:16 AM	John Doe	
Parallel Gateways Process					

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Task management / Using hooks

Hooks allow you to extract stateful logic from a component, so it can be tested and reused independently.

Users with task management permissions can create hooks to trigger specific

The fallback content to display on prerendering
, such as sending notifications when

The fallback content to display on prerendering occur. Follow the instructions below to set up roles for hooks scope usage:

» Manage hooks roles

The screenshot shows the FLOWX.AI platform interface. On the left is a sidebar with navigation links: Processes (Definitions, Active process), Content Management (Enumerations, Substitution tags, Content models, Languages, Source systems), and Plugins (Task Manager, Hooks, Stages). The main area is titled 'Hooks' and contains a table with three rows of data:

Hook Name	Type	Active	Actions
Hook1	Process	Active	
Hook2	Swimlane	Active	
Hook3	Swimlane	Active	

Hooks can be linked to different events and define what will happen when they are triggered. Below you can find a list of all possible triggers for each hook.

Process Swimlane Stage

- unique result
- only one rule will match, or no rule

Creating a hook

To create a new hook, follow the next steps:

1. Open

The fallback content to display on prerendering

2. Go to Task Manager and select **Hooks**.

3. Click **New Hook** (you can also import or export a hook).

4. Fill in the required details.

Hooks

The screenshot shows a 'Hooks' section on the left with three entries: 'Hook Name', 'Hook1', 'Hook2', and 'Hook3'. A central modal window titled 'Add new hook' is open, containing fields for 'Name' (with a note about no special characters or spaces), 'Parent process', 'Type' (dropdown), 'Trigger' (dropdown), 'Triggered Process' (input field), and an 'Active' checkbox which is checked. A blue 'Add' button is at the bottom.

Types of hooks

There are three types of hooks you can create in Task Manager:

- process hooks
- swimlane hooks
- stage hooks

! INFO

Swimlane and stage hooks can be configured with an SLA (time when a triggered process is activated).

Type

Process

Swimlane

Swimlane

Process

Stage

Process

Stage

Add new hook x

Name ⓘ Name shouldn't contain any special characters or white spaces.

Parent process

Type ⓘ Swimlane

Trigger ⓘ Triggered Process

Add SLA
Time to activate

ⓘ Format details [here](#).

Dismiss SLA when Swimlane Exited

Active

Add

! INFO

Dismiss SLA is available only for hooks configured with SLA.

[Here](#) you can find more information about the SLA - duration formatting.

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Task management / Using out of office records

The Out-of-office feature allows you to register users availability to perform a task. It can be allocated manually or automatically.

User	Start Date	End Date	Edited at	Edited by	Actions
Jane Smith (jane.sm...)	03 Aug 2022	07 Aug 2022	03 Aug 2022, 3:24 PM	John Doe	
Bess Twishes (bess....)	03 Aug 2022	31 Dec 2046	03 Aug 2022, 3:16 PM	John Doe	
Gene Eva (gene.eva...)	08 Aug 2022	27 Aug 2022	03 Aug 2022, 9:25 AM	John Doe	

!(INFO)

Users with out-of-office status are excluded from the candidates for automatic task allocation list during the out-of-office period. More information about allocation rules, [here](#).

Accessing out-of-office records

To add out-of-office records, follow the next steps:

1. Open

The fallback content to display on prerendering

2. From the side menu, under **Task Management**, select the **Out office** entry.

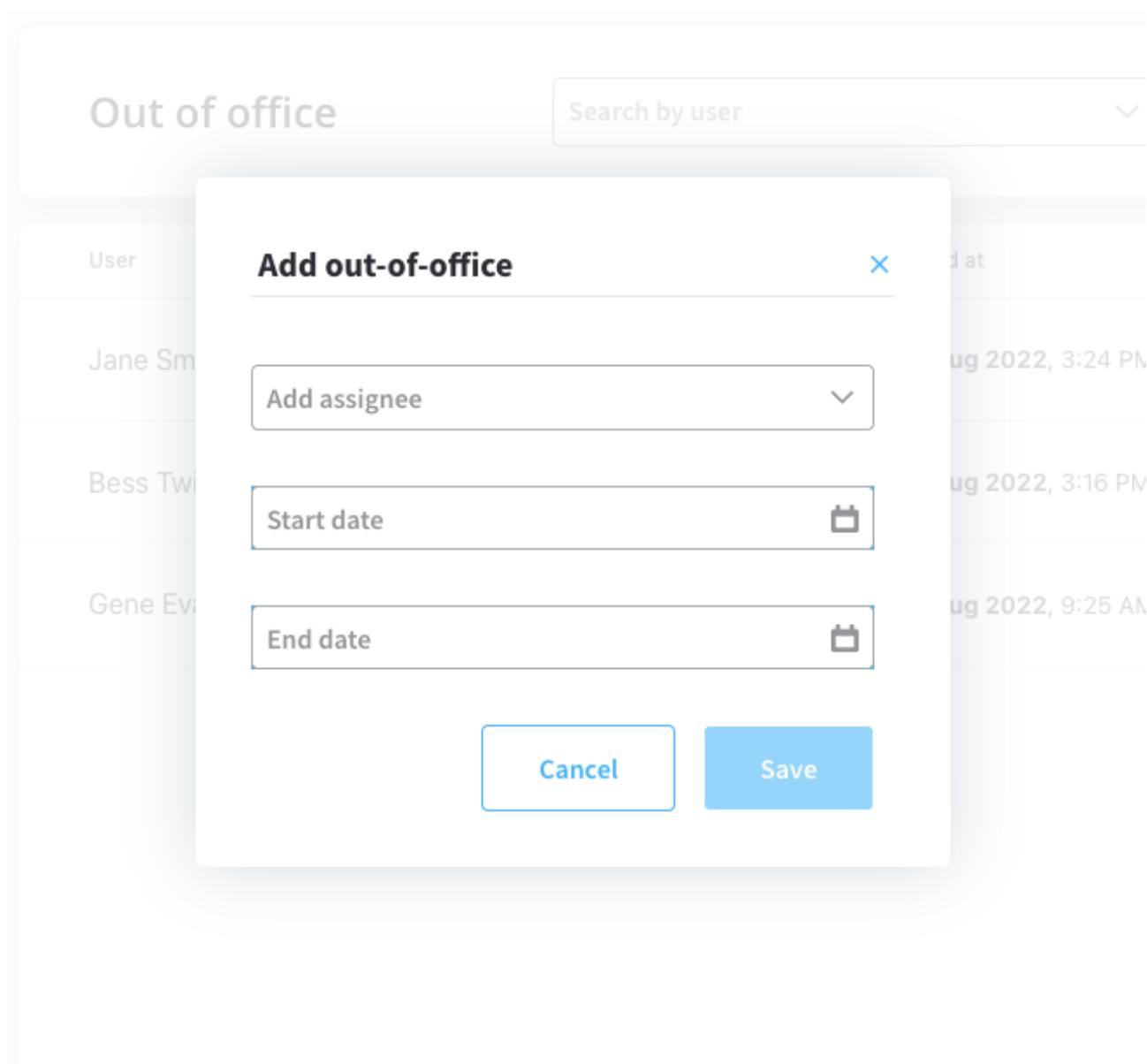
The screenshot shows the FLOWX.AI platform's sidebar navigation on the left and a main content area on the right. The sidebar includes sections for Plugins, Task Manager (with 'All tasks', 'Hooks', 'Stages', 'Allocation rules', and 'Out of office' selected), Notification templates, Document templates, General Settings (with 'Generic Parameters', 'Integration management', 'Licensing', and 'Access management'), and a user profile for 'John Doe'. The main content area displays a table titled 'Out of office' with columns for User, Start Date, End Date, Edited at, and Edited by. Three entries are listed: Jane Smith, Bess Twishes, and Gene Eva, each with edit and delete icons. A search bar and an 'Add out-of-office' button are also present.

User	Start Date	End Date	Edited at	Edited by
Jane Smith (jane....)	03 Aug 2022	07 Aug 2022	03 Aug 2022, 3:24 PM	John Doe
Bess Twishes (be...)	03 Aug 2022	31 Dec 2046	03 Aug 2022, 3:16 PM	John Doe
Gene Eva (gene....)	08 Aug 2022	27 Aug 2022	03 Aug 2022, 9:25 AM	John Doe

Adding out-of-office records

To add out-of-office records, follow the next steps:

1. Click **Add out-of-office** button, in the top-right corner.
2. Fill in the following mandatory details:
 - Assignee - user single select
 - Start Date (! cannot be earlier than tomorrow)
 - End Date (! cannot be earlier than tomorrow)

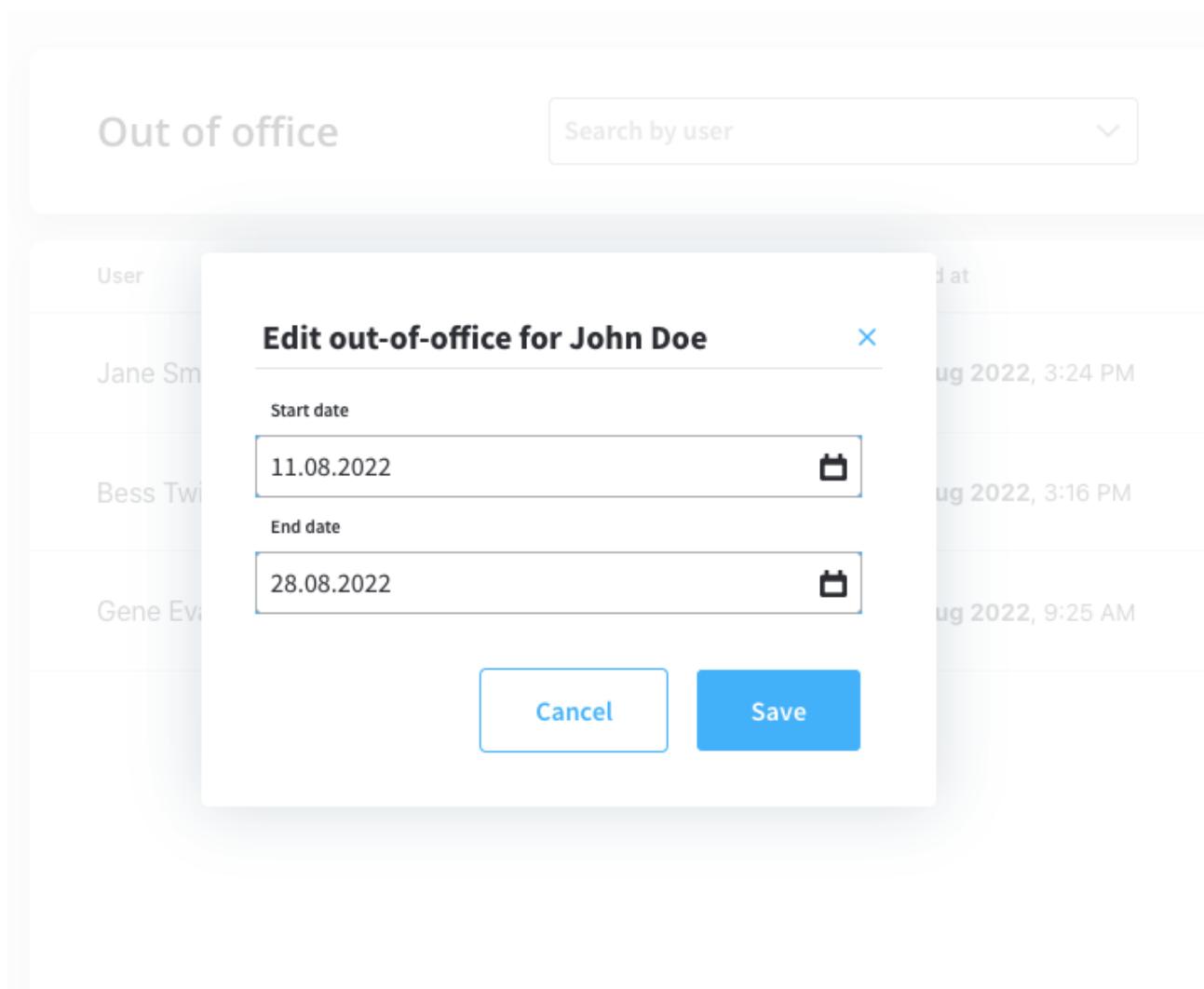


3. Click **Save**.

Editing out-of-office records

To edit out-of-office records, follow the next steps:

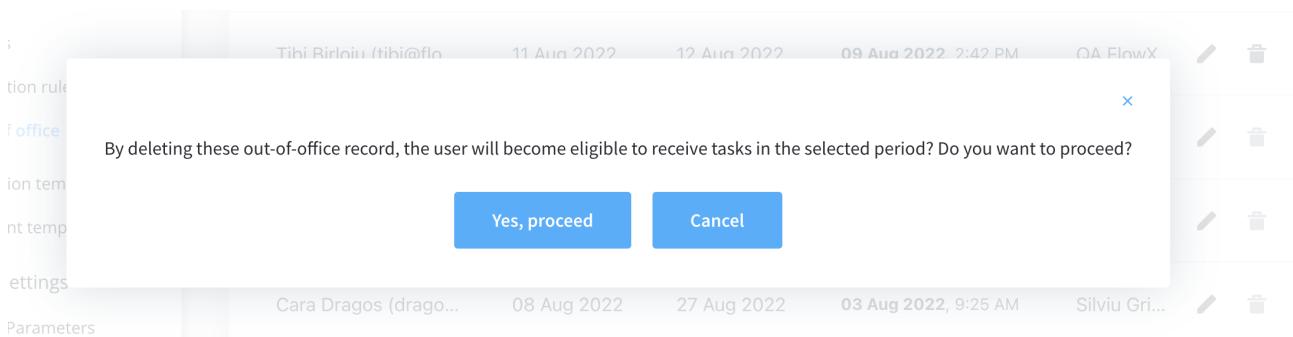
1. Click **Edit** button.
2. Modify the dates (! cannot be earlier than tomorrow).
3. Click **Save**.



Deleting out-of-office records

To delete out-of-office records, follow the next steps:

1. From the **out-of-office list**, select a **record**.
2. Click **Delete** button. A pop-up message will be displayed: *"By deleting this out-of-office record, the user will become eligible to receive tasks in the selected period. Do you want to proceed?"*



🔥 DANGER

If you choose to delete an out-of-office record, the user is eligible to receive tasks allocation during the mentioned period. More information about automatic task allocation, [here](#).

3. Click **Yes, proceed** if you want to delete the record, click **Cancel** if you want to abort the deletion.

⚠ CAUTION

If the out-of-office period contains days selected in the past, the user cannot delete the record, the following message is displayed: *"You can't delete this*

record because it already affected allocations in the past. Try to shorten the period, if it didn't end."

User	Start Date	End Date	Edited at	Edited by	Action
Jane Smith (jane.s...)	03 Aug 2022	07 Aug 2022	03 Aug 2022, 3:24		You can't delete this record because it already affected allocations in the past.
Bess Twishes (bes...	03 Aug 2022	31 Dec 2046	03 Aug 2022, 3:16 PM	John Doe	
Gene Eva (gene.ev...	08 Aug 2022	27 Aug 2022	03 Aug 2022, 9:25 AM	John Doe	

Viewing out-of-office records

The out-of-office records list contains the following elements:

1. **User** - firstName, lastName, userName
2. **Start Date** - the date when the out-of-office period will be effective
3. **End Date** - the date when the out-of-office period will end
4. **Edited at** - the last time when an out-of-office record was edited
5. **Edited by** - the user who edited/created the out-of-office record

The screenshot shows a table titled "Out of office" with the following columns and data:

1 User	2 Start Date	3 End Date	4 Edited at	5 Edited by
Jane Smith (jane....)	03 Aug 2022	07 Aug 2022	03 Aug 2022, 3:24 PM	John Doe
Bess Twishes (be...	03 Aug 2022	31 Dec 2046	03 Aug 2022, 3:16 PM	John Doe
Gene Eva (gene.e...	08 Aug 2022	27 Aug 2022	03 Aug 2022, 9:25 AM	John Doe

!(INFO)

The list is sorted in reverse chronological order by “edited at” `dateTime` (newest added on top).

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Task management / Using stages

You can define specific stages during the execution of a process. Stages are configured on each node and they will be used to trigger an event when passing from one stage to another.

Creating a new stage

To create a new stage, follow the next steps:

1. Open
The fallback content to display on prerendering
2. Go to Task Manager and select **Stages**.
3. Click **New Stage**.
4. Fill in the required details.

Stages

The screenshot shows a list of stages on the left: PF, Offboa, Onboa, Identifi, testirin, and Silviu. A modal dialog box titled "Add new stage" is open in the center. It contains a "Name" input field, a validation message, and an "Add" button.

Name

PF

Offboa

Onboa

Identifi

testirin

Silviu

Add new stage x

Name

i Name shouldn't contain any special characters or white spaces.

Add

Assigning a node to a stage

To assign a node to a stage, follow the next steps:

1. Open **FLOWX Designer** and then select your **process**.
2. Choose the node you want to assign and select the **Node Config** tab.
3. Scroll down until you find the **Stage** field and click the dropdown button.
4. Choose the stage you want to assign.

Node: Client Form (ID: 477506) ^ X

Node Config Actions

Swimlane Default ^

Stage Onboarding X V

Response Timeout

Response Timeout (PT30S)

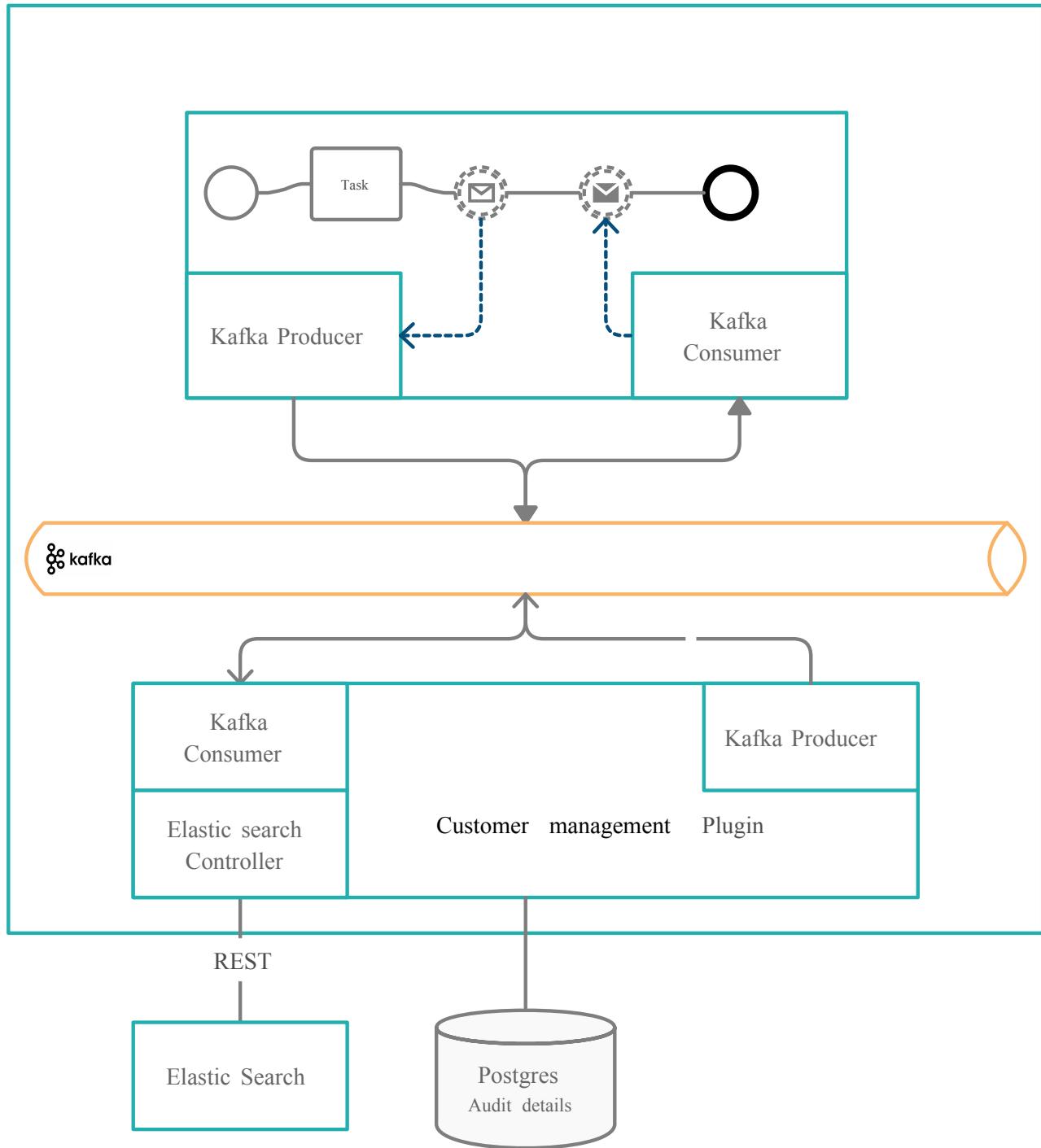
Data stream topics

Save

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Customer management / Using the customer management plugin

The customer management plugin offers the possibility of retrieving customer details from an elasticSearch engine.



The plugin listens for incoming requests on a Kafka topic and sends the reply to the Engine on an outgoing topic.

» Kafka topics for Customer Management

Kafka topics for customer management

Customer Search

!(INFO)

The kafka topics used for the Customer Management plugin can be defined/overwritten using the following environment variables (that can be found in the deployment of the service):

- `KAFKA_TOPIC_CUSTOMER_SEARCH_IN` - used to search customers in the customer management plugin
- `KAFKA_TOPIC_CUSTOMER_SEARCH_OUT` - used to get the response from the customer management plugin to the Engine.

The request sent to the plugin can use any key that was previously configured in the elasticsearch index where the customers are saved.

Example of an elastic search index:

```
{  
  "settings": {  
    "analysis": {  
      "normalizer": {  
        "lowercase_normalizer": {  
          "type": "custom",  
          "filter": ["lowercase"]  
        }  
      }  
    }  
  }  
}
```

```
        }
    }
},
},
"mappings": {
    "properties": {
        "CIF": {
            "type": "keyword",
            "normalizer": "lowercase_normalizer"
        },
        "ClientUniqueIdentifier": {
            "type": "keyword",
            "normalizer": "lowercase_normalizer"
        },
        "CNPFlex": {
            "type": "text"
        },
        "ClientType": {
            "type": "text"
        },
        "ClientCategory": {
            "type": "text"
        },
        "FirstName": {
            "type": "text",
            "fields": {
                "keyword": {
                    "type": "keyword",
                    "normalizer": "lowercase_normalizer"
                }
            }
        },
        "LastName": {
            "type": "text",
            "fields": {

```

```
"keyword":{  
    "type": "keyword",  
    "normalizer": "lowercase_normalizer"  
}  
}  
},  
"CompanyName":{  
    "type": "text",  
    "fields": {  
        "keyword":{  
            "type": "keyword",  
            "normalizer": "lowercase_normalizer"  
        }  
    }  
},  
"DateOfBirth":{  
    "type": "date",  
    "format": "dd.MM.yyyy"  
},  
"IDDocType":{  
    "type": "text"  
},  
"IDSeries":{  
    "type": "text"  
},  
"IDNumber":{  
    "type": "text"  
},  
"IDIssueDate": {  
    "type": "date",  
    "format": "dd.MM.yyyy"  
},  
"IDEpiryDate":{  
    "type": "date",  
    "format": "dd.MM.yyyy"
```

```
        },
        "LegalForm": {
            "type": "text"
        },
        "CreatedDatePJ": {
            "type": "date",
            "format": "dd.MM.yyyy"
        },
        "ClientClosedDate": {
            "type": "date",
            "format": "dd.MM.yyyy"
        },
        "LastModifiedDate": {
            "type": "date",
            "format": "dd.MM.yyyy"
        },
        "ListID": {
            "type": "text"
        },
        "MobilePhone": {
            "type": "text"
        }
    }
}
```

With this index configuration we can search for customers using any key:

Key examples

Example 1 - using only the "ClientUniqueIdentifier" key:

```
{  
  "ClientUniqueIdentifier": "1900101223344"  
}
```

Example 2 - using "FirstName" and "LastName" keys:

```
{  
  "FirstName": "TestFirstName",  
  "LastName": "Test Last Name"  
}
```

Example 3 - using "FirstName", "DateOfBirth" and "LegalForm" keys:

```
{  
  "FirstName": "TestFirstName",  
  "DateOfBirth": "01.01.1990",  
  "LegalForm": "PF"  
}
```

Keys description

- **customers** - list of customers found in the customer management, in the used elasticsearch index, maximum 10 results
- **hasMore** - boolean, true if number of results are bigger than 10, false if the number of results are equal or smaller than 10
- **error** - error description if the request returned an error

Topic name example:

```
ro.flowx.updates.sandbox.customer.management.response
```

Sent body example:

```
"searchResults" : {  
    "customers" : [ {  
        "id" : "CL12345",  
        "firstName" : "John Doe",  
        "lastName" : "Doe",  
        "birthDate" : "27.02.1982",  
        "cui" : "1820227103865_84",  
        "companyName" : "",  
        "clientCategory" : "PF",  
        "clientType" : "PF",  
        "idSeries" : "RT",  
        "idNumber" : "879948",  
        "idDocType" : "CI",  
        "idExpiryDate" : "27.02.2023",  
        "legalForm" : "",  
        "listId" : "4691602",  
        "mobilePhone" : "0711111111",  
        "attributes" : null,  
        "type" : "PF"}],  
    "hasMore" : false,  
    "error" : null  
}
```

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Customer management / Customer management plugin example

Integrate a customer search in a business flow

Follow the next steps to use the user personal number to perform a search query in the customer management plugin.

1. First make sure the details about customers are indexed in the search engine (for example, elasticSearch).
2. Open
The fallback content to display on prerendering
web app and create a
The fallback content to display on prerendering
.
3. Add a **Kafka send event** node.
4. Configure the **Kafka send event** node by adding the following elements:
 - Kafka topic - defined on the `KAFKA_TOPIC_CUSTOMER_SEARCH_IN` environment variable
 - Message body (example of identifiers for an indexed customers):

Parameters

Custom From integration

Topics

```
customerSearchTopic-qa
```

Message

```
1  {
2    "id": "${clientIdentification.cif}",
3    "cui": "${clientIdentification.uniqueCode}",
4    "firstName": "${clientIdentification.firstName}",
5    "lastName": "${clientIdentification.lastName}",
6    "birthDate": "${clientIdentification.dateOfBirth}",
7    "companyName": "${clientIdentification.identificationData.name}"
8 }
```

Advanced configuration

Show Headers

```
1 {"processInstanceId": ${processInstanceId}}
```

INFO

For more examples of keys, check [Using the customer management plugin](#).

5. Add a **Kafka receive event**.

6. Configure the topic on which you want to receive the response from the CRM, on the value of `KAFKA_TOPIC_CUSTOMER_SEARCH_OUT` environment variable.

Data stream topics

Custom From integration

Topic Name	Key Name
ai.flowx.updates.qa.customer.management.response	receiveReply

trash

Response example:

```
"searchResults" : {  
    "customers" : [ {  
        "id" : "ID3456",  
        "firstName" : "Jane Doe",  
        "lastName" : "Doe",  
        "birthDate" : "27.02.1980",  
        "cui" : "1820227103840_84",  
        "companyName" : "",  
        "clientCategory" : "PF_INTL",  
        "clientType" : "PF",  
        "idSeries" : "RT",  
        "idNumber" : "879948",  
        "idDocType" : "CI",  
        "idExpiryDate" : "27.02.2023",  
        "legalForm" : "",  
        "listId" : "4691602",  
        "mobilePhone" : "0711111111",  
    } ]}
```

```
    "attributes" : null,  
    "type" : "PF"}],  
    "hasMore" : false,  
    "error" : null  
}
```

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / OCR plugin

The OCR (Optical Character Recognition) plugin is a powerful tool that enables you to read barcodes and extract handwritten signatures from .pdf documents with ease.

Before using the OCR service for reading barcodes and extracting signatures, please note the following requirements:

⚠ CAUTION

- All *.pdf documents that are sent to the OCR service for reading barcodes and extracting handwritten signatures should be scanned at a minimum resolution of 200DPI (approximately 1654x2339 px for A4 pages)
- Barcode is searched on the top 15% of each image (scanned page)
- Signatures are detected on boxes with a border: 4px black solid

- Only two signatures per image (scanned page) are detected.
- All *.pdf documents should be scanned at a minimum resolution of 200DPI (approximately 1654x2339 px for A4 pages).
- The barcode is searched in the top 15% of each scanned page.
- Signatures are detected within boxes with a 4px black solid border.
- The plugin detects up to two signatures per scanned page.
- Only two signatures per image (scanned page) are detected.

INFO

The plugin supports **1D Code 128** barcodes. For more information about this barcode type, please refer to the documentation [here](#).

Using the OCR plugin

You can utilize the OCR plugin to process generic document templates by either using a specific flow on FLOWX.AI (HTML template) or any other document editor.

INFO

Using a specific flow on FLOWX.AI offers several advantages:

- Centralized management of templates and flows within a single application.
- Access to template history and version control.

Use case

1. Prepare and print generic document templates.
2. End-users complete, sign, and scan the documents.
3. Upload the scanned documents to the flow.
4. FLOWX validates the template (barcode) and the signatures.

Scenario for FLOWX.AI generated documents

1. Utilize the **Documents plugin** to create a **document template**.

» Generating documents based on templates

The screenshot shows the FLOWX.AI platform interface. On the left, there is a sidebar with the following navigation items:

- Processes**
 - Definitions
 - Active process
 - Process Instances
 - Failed process start
- Content Management**
 - Enumerations
 - Substitution tags
 - Content models
 - Languages
 - Source systems
- Plugins**
 - Task Manager
 - All tasks
 - Hooks
 - Stages
 - Allocation rules
 - Out of office
- Notification templates

At the bottom of the sidebar, there is a user profile icon for "John Doe" and a three-dot menu icon.

The main content area is titled "Process Definitions". It features a search bar and a three-dot menu icon. Below the title, it says "Drafts / In progress" and "Published".

Drafts / In progress

Name	Version	Edited at	Edited by	Actions
s...	1	30 Sep 2022, 5:41 PM	Silviu Grigore	
d...	2	30 Sep 2022, 2:32 PM	andrei antal	
t...	1	30 Sep 2022, 11:15 AM	QA FlowX	
T...	4	30 Sep 2022, 10:20 AM	QA FlowX	

Published

Name	Version	Published at	Published by	Actions
A...	1	03 Oct 2022, 8:12 AM	QA FlowX	
C...	19	30 Sep 2022, 3:08 PM	Silviu Grigore	
d...	1	30 Sep 2022, 10:34 AM	Bogdan Ionescu	
T...	1	30 Sep 2022, 10:18 AM	QA FlowX	

2. Create a process and add a **Kafka Send Action** to a **Message event send** node. Here you specify the **kafka topic** (address) where the template will be generated.

The screenshot shows the 'Action Edit' screen for an action named 'send_main_applicant_consent'. The action has an ID of 584701 and is set to run in order 1. It is configured as an automatic, mandatory, non-repeatable Kafka send action. The 'Autorun Children?' toggle is off. The 'Parameters' section shows the 'Topics' field set to 'ai.flowx.in.sandbox.document.html.generate.v1'. There are two tabs for parameters: 'Custom' (selected) and 'From integration'.

(!) INFO

The Kafka topic for generating the template must match the topic defined in the `KAFKA_TOPIC_DOCUMENT_GENERATE_HTML_IN` variable. Refer to the **Kafka configuration guide** for more details. For additional information, please see the **Documents plugin setup guide**.

3. Fill in the **Message**. The request body should include the following values:

- **documentList** - a list of documents to be generated, including properties such as name and values 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 replace the placeholders in the document template, the keys used in the map should match those defined in the HTML template

Parameters

Custom From integration

Topics

ai.flowx.in.sandbox.document.html.generate.v1

Message

```
1  {
2    "documentList": [
3      {
4        "customId": "1234",
5        "templateName": "Agreement",
6        "language": "ro",
7        "data": {
8          "offerName": "The greatest offer - deluxe edition",
9          "companyName": "Test Company SRL",
10         "cui": "R01234567",
11         "firstName": "Bess",
12         "lastName": "Twishes",
13         "offerExpiryDate": "31.12.2099"
14       },
15     },
16   }
```

Advanced configuration

Show Headers



The **data** parameters must be defined in the document template beforehand. For more information, check the **WYSIWYG Editor** section.

Documents Templates - Test_docs

Body **Data model**

Name	Type	Mandatory	+
customId	STRING	false	 
templateName	STRING	false	 
language	STRING	false	 
version	NUMBER	false	 
draft	BOOLEAN	false	 
offerName	STRING	false	 
companyName	STRING	false	 
cui	STRING	false	 
lastName	STRING	false	 
firstName	STRING	false	 

4. Add a barcode.



- to include a **default barcode**, add the following parameter to the message body: `includeBarCode: true`.
- to include a **custom barcode**, set `includeBarCode: false` and provide the desired data in the `data` field

5. Add a **message received event** node and specify the topic where you want to receive the response.

⚠ CAUTION

Ensure that the topic matches the one defined in the `KAFKA_TOPIC_DOCUMENT_GENERATE_HTML_OUT` variable.

Node: **receive_consent** (ID: 584502)

Node Config

General Config

Node name

receive_consent

Can go back?

Swimlane

Default



Stage



Response Timeout

Response Timeout (PT30S)

Data stream topics

Custom

From integration

Topic Name

ai.flowx.updates.sandbox.document.html.gener

Key Name

ocrBarcodeResponse



Add stream

6. Add a **user task node** and configure an **Upload file action** to send the file (defined by the **KAFKA_TOPIC_DOCUMENT_PERSIST_IN** variable) to the storage solution (for example, S3).

Parameters

Topics

```
ai.flowx.in.sandbox.document.persist.v1
```

Replace Values

Document Type

```
BULK
```

Replace Values

Folder

```
1234_${processInstanceId}
```

Replace Values

Advanced configuration

Show Headers

Data to send

7. Next, the response will be sent back to the kafka topic defined by **KAFKA_TOPIC_DOCUMENT_PERSIST_OUT** environment variable through a callback action/subprocess.
8. Next, send the response to the OCR Kafka topic defined at **KAFKA_TOPIC_OCR_IN** variable (representing the path to the S3 file)

9. Display the result of the OCR validation on the kafka topic defined at
KAFKA_TOPIC_OCR_OUT.

Setup guide

Refer to the OCR plugin setup guide for detailed instructions on setting up the OCR plugin:

» [OCR plugin setup](#)

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Custom Plugins / Reporting / Authorization & access roles

IAM solution

INFO

Superset is using by default flask-openid, as implemented in flask-security.

Superset can be integrated with Keycloak, an open-source identity and access management solution. This integration enables users to manage authentication and authorization for their Superset dashboards.

» Configuring an IAM solution

Prerequisites

- Keycloak server
 - Keycloak Realm
 - Keycloak Client & broker configured with OIDC protocols
 - client_secret.json
- admin username & password of postgres instance
- Superset Database created in postgresql
- optionally Cert-manager if you want to have SSL certificates on hostnames.

» Superset + Keycloak configuration

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Plugins setup guides / Customer management plugin setup

Infrastructure Prerequisites:

The Customer management plugin is available as a docker image so we need to configure:

Elastic Search

In order to install elasticsearch instance Elastic Cloud on Kubernetes (ECK) can be used.

Use ECK quickstart to deploy CRDs and create elasticsearch instances:

Elasticsearch instance:

```
apiVersion: elasticsearch.k8s.elastic.co/v1
kind: Elasticsearch
metadata:
  name: elasticsearch-flowx
  namespace: elastic-system
spec:
  version: 7.9.3
  updateStrategy:
    changeBudget:
      maxSurge: 3
      maxUnavailable: 1
  nodeSets:
    # 3 dedicated master nodes
    - name: master
      count: 3
      config:
        node.master: true
        node.data: false
        node.ingest: false
        node.remote_cluster_client: false
        # this allows ES to run on nodes even if their
```

```
vm.max_map_count has not been increased, at a performance
cost
    # node.store.allow_mmap: false
podTemplate:
  spec:
    initContainers:
      - name: sysctl
        securityContext:
          privileged: true
        command: ['sh', '-c', 'sysctl -w
vm.max_map_count=262144']
      - name: install-plugins
        command:
          - sh
          - -c
          - |
            bin/elasticsearch-plugin install --batch
repository-gcs
  containers:
    - name: elasticsearch
      resources:
        limits:
          memory: 6Gi
          cpu: 2
        requests:
          memory: 2Gi
          cpu: 1
      env:
        - name: ES_JAVA_OPTS
          value: "-Xms2g -Xmx2g"
        - name: READINESS_PROBE_TIMEOUT
          value: "10"
      readinessProbe:
        exec:
          command:
```

```
        - bash
        - -c
        - /mnt/elasticsearch/internal/scripts/readiness-
probe-script.sh
            failureThreshold: 3
            initialDelaySeconds: 10
            periodSeconds: 12
            successThreshold: 1
            timeoutSeconds: 12
        affinity:
        podAntiAffinity:

preferredDuringSchedulingIgnoredDuringExecution:
    - weight: 100
        podAffinityTerm:
            labelSelector:
                matchLabels:
                    elasticsearch.k8s.elastic.co/cluster-
name: elasticsearch-flowx
            topologyKey: kubernetes.io/hostname
        # request 2Gi of persistent data storage for pods in
        this topology element
        volumeClaimTemplates:
            - metadata:
                name: elasticsearch-data
            spec:
                accessModes:
                    - ReadWriteOnce
                resources:
                    requests:
                        storage: 5Gi
                storageClassName: standard
        # 3 ingest-data nodes
        - name: ingest-data
            count: 3
```

```
config:
  node.master: false
  node.data: true
  node.ingest: true
  # this allows ES to run on nodes even if their
  vm.max_map_count has not been increased, at a performance
  cost
  # node.store.allow mmap: false
podTemplate:
  spec:
    initContainers:
      - name: sysctl
        securityContext:
          privileged: true
        command: ['sh', '-c', 'sysctl -w
vm.max_map_count=262144']
    containers:
      - name: elasticsearch
        resources:
          limits:
            memory: 8Gi
            cpu: 2
          requests:
            memory: 4Gi
            cpu: 1
        env:
          - name: ES_JAVA_OPTS
            value: "-Xms2g -Xmx2g"
    affinity:
      podAntiAffinity:
        preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 100
            podAffinityTerm:
              labelSelector:
```

```
        matchLabels:  
         .elasticsearch.k8s.elastic.co/cluster-  
name: elasticsearch-flowx  
          topologyKey: kubernetes.io/hostname  
# nodeSelector:  
#   diskType: ssd  
#   environment: production  
# request 2Gi of persistent data storage for pods in  
this topology element  
volumeClaimTemplates:  
- metadata:  
  name: elasticsearch-data  
spec:  
  accessModes:  
  - ReadWriteOnce  
  resources:  
    requests:  
      storage: 20Gi  
  storageClassName: standard
```

(Optional) Kibana instance:

```
apiVersion: kibana.k8s.elastic.co/v1  
kind: Kibana  
metadata:  
  name: kibana-flowx  
  namespace: elastic-system  
spec:  
  version: 7.9.3  
  count: 1  
  elasticsearchRef:  
    name: elasticsearch-flowx  
    namespace: elastic-system
```

```
config:
  elasticsearch.requestHeadersWhitelist:
    - authorization
podTemplate:
  spec:
    containers:
      - name: kibana
        resources:
          requests:
            memory: 1Gi
            cpu: 0.5
          limits:
            memory: 3Gi
            cpu: 2
```

The index used by customer management plugin should be created.

Postgres database

This plugin can work without this database, it will not store the audit data.

Basic Postgres configuration

```
cldb:
  existingSecret: {{secretName}}
metrics:
  enabled: true
  service:
    annotations:
      prometheus.io/port: {{phrometeus port}}
      prometheus.io/scrape: "true"
  type: ClusterIP
  serviceMonitor:
```

```
additionalLabels:
  release: prometheus-operator
enabled: true
interval: 30s
scrapeTimeout: 10s
persistence:
  enabled: true
  size: 4Gi
postgresqlDatabase: {{postgres databaseName}}
postgresqlUsername: {{postgres user}}
resources:
  limits:
    cpu: 500m
    memory: 512Mi
  requests:
    cpu: 200m
    memory: 256Mi
service:
  annotations:
    fabric8.io/expose: "false"
```

Configuration

Authorization configuration

The following variables need to be set in order to connect to the identity management platform:

SECURITY_OAUTH2_BASE_SERVER_URL

SECURITY_OAUTH2_CLIENT_CLIENT_ID

SECURITY_OAUTH2_REALM

Datasource configuration

To store audit for searches this plugins use a postgres database.

The following configuration details need to be added using environment variables:

SPRING_DATASOURCE_URL**SPRING_DATASOURCE_USERNAME****SPRING_DATASOURCE_PASSWORD**

You will need to make sure that the user, password, connection link and db name are configured correctly, otherwise you will receive errors at start time.

If you are going to use a database to store the audit, you can use the built-in script to maintain the database schema.

Elastic search configuration

The connection to elastic search cluster is done over https using the elastic search api. To connect to the it you will need to configure the connection details and index use to store customers.

```
elasticsearch:  
  ssl: false  
  nodes:  
    -  
      hostname: ${ELASTICSEARCH_HOST}
```

```
port: ${ELASTICSEARCH_PORT}  
scheme: ${ELASTICSEARCH_HTTP_SCHEME}  
user: ${ELASTICSEARCH_USER}  
password: ${ELASTICSEARCH_PASSWORD}  
customer-index: ${ELASTICSEARCH_CUSTOMER_INDEX}  
size: 10
```

Kafka configuration

The following Kafka related configurations can be set by using environment variables:

`SPRING_KAFKA_BOOTSTRAP_SERVERS` - address of the Kafka server

`SPRING_KAFKA_CONSUMER_GROUP_ID` - group of consumers

`KAFKA_CONSUMER_THREADS` - the number of Kafka consumer threads

`KAFKA_AUTH_EXCEPTION_RETRY_INTERVAL` - the interval between retries after `AuthorizationException` is thrown by `KafkaConsumer`

`KAFKA_MESSAGE_MAX_BYTES` - this is the largest size of the message that can be received by the broker from a producer.

Each action available in the service corresponds to a Kafka event. A separate Kafka topic must be configured for each use-case.

⚠ CAUTION

The Engine is listening for messages on topics with names of a certain pattern, make sure to use correct outgoing topic names when configuring the

documents plugin.

Needed topics:

KAFKA_TOPIC_CUSTOMER_SEARCH_IN

KAFKA_TOPIC_CUSTOMER_SEARCH_OUT

⚠ CAUTION

In order to match a request made to the customer management plugin, the engine will have to send the process id on a Kafka header.

Logging

The following environment variables could be set in order to control log levels:

LOGGING_LEVEL_ROOT - root spring boot microservice logs

LOGGING_LEVEL_APP - app level logs

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Plugins setup guides / Documents plugin setup / Configuring access rights for Documents

Granular access rights can be configured for restricting access to the Documents plugin component.

The following access authorizations is provided, with the specified access scopes:

1. **Manage-document-templates** - for configuring access for managing document templates

Available scopes:

- import - users are able to import document templates
 - read - users are able to view document templates
 - edit - users are able to edit document templates
 - admin - users are able to publish or delete document templates
- The Document plugin is preconfigured with the following default users roles for each of the access scopes mentioned above:
- manage-document-templates
 - import:
 - ROLE_DOCUMENT_TEMPLATES_IMPORT
 - ROLE_DOCUMENT_TEMPLATES_EDIT
 - ROLE_DOCUMENT_TEMPLATES_ADMIN
 - read:
 - ROLE_DOCUMENT_TEMPLATES_READ
 - ROLE_DOCUMENT_TEMPLATES_IMPORT
 - ROLE_DOCUMENT_TEMPLATES_EDIT

- ROLE_DOCUMENT_TEMPLATES_ADMIN
- edit:
 - ROLE_DOCUMENT_TEMPLATES_EDIT
 - ROLE_DOCUMENT_TEMPLATES_ADMIN
- admin:
 - ROLE_DOCUMENT_TEMPLATES_ADMIN

ROLES

These roles need to be defined in the chosen identity provider solution.

In case other custom roles are needed, you can configure them using environment variables. More than one role can be set for each access scope.

To configure access for each of the roles above, adapt the following input:

```
SECURITY_ACCESSAUTHORIZATIONS_AUTHORIZATIONNAME_SCOPES_SCOPENAME_ROLESALLOWED: NEEDED_ROLE_NAMES
```

Possible values for AUTHORIZATIONNAME: MANAGEDOCUMENTTEMPLATES.

Possible values for SCOPENAME: import, read, edit, admin.

For example, if you need to configure role access for read, insert this:

```
SECURITY_ACCESSAUTHORIZATIONS_MANAGEDOCUMENTTEMPLATES_SCOPES_READROLE_NAME_TEST
```

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Plugins setup guides / Notification templates plugin setup / Configuring access rights for Notifications

Granular access rights can be configured for restricting access to the Notification plugin component.

The following access authorizations are provided, with the specified access scopes:

1. **Manage-notification-templates** - for configuring access for managing notification templates

Available scopes:

- import - users are able to import notification templates
- read - users are able to view notification templates
- edit - users are able to edit notification templates
- admin - users are able to publish or delete notification templates

The Notification plugin is preconfigured with the following default users roles for each of the access scopes mentioned above:

- manage-notification-templates
 - import
 - ROLE_NOTIFICATION_TEMPLATES_IMPORT

- ROLE_NOTIFICATION_TEMPLATES_EDIT
- ROLE_NOTIFICATION_TEMPLATES_ADMIN
- read:
 - ROLE_NOTIFICATION_TEMPLATES_READ
 - ROLE_NOTIFICATION_TEMPLATES_IMPORT
 - ROLE_NOTIFICATION_TEMPLATES_EDIT
 - ROLE_NOTIFICATION_TEMPLATES_ADMIN
- edit:
 - ROLE_NOTIFICATION_TEMPLATES_EDIT"
 - ROLE_NOTIFICATION_TEMPLATES_ADMIN"
- admin:
 - ROLE_NOTIFICATION_TEMPLATES_ADMIN

⚠ CAUTION

These roles need to be defined in the chosen identity provider solution.

In case other custom roles are needed, you can configure them using environment variables. More than one role can be set for each access scope.

To configure access for each of the roles above, adapt the following input:

```
SECURITY_ACCESSAUTHORIZATIONS_AUTHORIZATIONNAME_SCOPES_SCOPENAME_ROLESALLOWED: NEEDED_ROLE_NAMES
```

Possible values for AUTHORIZATIONNAME: **MANAGENOTIFICATIONTEMPLATES**.

Possible values for SCOPENAME: import, read, edit, admin.

For example, if you need to configure role access for read, insert this:

```
SECURITY_ACCESSAUTHORIZATIONS_MANAGENOTIFICATIONTEMPLATES_SCOPES  
ROLE_NAME_TEST
```

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Plugins setup guides / OCR plugin setup

The OCR plugin is a docker image that can be deployed using the following infrastructure prerequisites:

Infrastructure Prerequisites:

- S3 bucket or alternative (for example, minio)
- Kafka cluster

⚠️ IMPORTANT

Starting with `ocr-plugin 1.X` it no longer requires RabbitMQ.

The following environment from previous releases must be removed in order to use OCR plugin: `CELERY_BROKER_URL`.

Deployment/Configuration

To deploy the OCR plugin, you will need to deploy `ocr-plugin` helm chart with custom values file.

Most important sections are these, but more can be extracted from helm chart.

```
image:  
  repository: <repository>/ocr-plugin  
  
applicationSecrets: {}  
  
replicaCount: 2  
  
resources: {}  
  
env: []
```

Credentials

S3 bucket:

```
applicationSecrets:  
  enable: true  
  envSecretKeyRef:  
    STORAGE_S3_ACCESS_KEY: access-key # default empty  
    STORAGE_S3_SECRET_KEY: secret-key # default empty  
  existingSecret: true  
  secretName: ocr-plugin-application-config
```

Kafka configuration

You can override the following environment variables:

Environment Variable	Definition	Default Value
ENABLE_KAFKA_SASL	Indicates whether Kafka SASL authentication is enabled	False
KAFKA_ADDRESS	The address of the Kafka bootstrap server in the format <code><hostname>:<port></code>	-
KAFKA_CONSUME_SCHEDULE	The interval (in seconds) at which Kafka messages are consumed	30

Environment Variable	Definition	Default Value
KAFKA_INPUT_TOPIC	The Kafka topic from which input messages are consumed	-
KAFKA_OCR_CONSUMER_GROUPID	The consumer group ID for the OCR Kafka consumer	ocr_group
KAFKA_CONSUMER_AUTO_COMMIT	Determines whether Kafka consumer commits offsets automatically	True

Environment Variable	Definition	Default Value
KAFKA_CONSUMER_AUTO_COMMIT_INTERVAL	The interval (in milliseconds) at which Kafka consumer commits offsets automatically	1000
KAFKA_CONSUMER_TIMEOUT	The timeout (in milliseconds) for Kafka consumer operations	28000
KAFKA_CONSUMER_MAX_POLL_INTERVAL	The maximum interval (in milliseconds) between consecutive polls for Kafka consume	25000

Environment Variable	Definition	Default Value
KAFKA_CONSUMER_AUTO_OFFSET_RESET	The strategy for resetting the offset when no initial offset is available or if the current offset is invalid	earliest
KAFKA_OUTPUT_TOPIC	The Kafka topic to which output messages are sent	-

Please note that the default values and examples provided here are for illustrative purposes. Make sure to replace them with the appropriate values based on your Kafka configuration.

CAUTION

When configuring the OCR plugin, make sure to use the correct outgoing topic names that match **the pattern expected by the Engine**, which listens for messages on topics with specific names.

Authorization

You can override the following environment variables:

Environment Variable	Definition	Default Value	Example
OAUTH_CLIENT_ID	The client ID for OAuth authentication	-	your_client_id
OAUTH_CLIENT_SECRET	The client secret for OAuth authentication	-	your_client_secret
OAUTH_TOKEN_ENDPOINT_URI	The URI of the token endpoint for OAuth authentication	-	https://oauth.com/token

Please note that the default values and examples provided here are for illustrative purposes. Make sure to replace them with the appropriate values based on your OAuth authentication configuration.

Storage (S3 configuration)

You can override the following environment variables:

Environment Variable	Definition	Default Value
STORAGE_S3_HOST	The host address of the S3 storage service	- minio: west-1
STORAGE_S3_SECURE_CONNECTION	Indicates whether to use a secure connection (HTTPS) for S3 storage	False
STORAGE_S3_LOCATION	The location of the S3 storage service	- eu-wes

Environment Variable	Definition	Default Value
STORAGE_S3_0CR_SCANS_BUCKET	The name of the S3 bucket for storing OCR scans	- pdf-scans
STORAGE_S3_0CR_SIGNATURE_BUCKET	The name of the S3 bucket for storing OCR signatures	- extracted-signatures
STORAGE_S3_0CR_SIGNATURE_FILENAME	The filename pattern for extracted OCR signatures	- extracted-signature

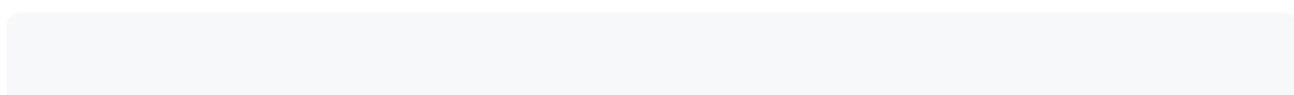
Environment Variable	Definition	Default Value	
STORAGE_S3_SECRET_KEY	The secret key for connecting to the S3 storage service	-	

Please note that the default values and examples provided here are for illustrative purposes. Make sure to replace them with the appropriate values based on your S3 storage configuration.

Performance

Environment Variable	Definition	Default Value
ENABLE_PERFORMANCE_PAYLOAD	When set to true, the response payload will contain performance metrics related to various stages of the process.	true

Example



```
"perf": {  
    "total_time": 998,  
    "split": {  
        "get_file": 248,  
        "extract_images": 172,  
        "extract_barcodes": 37,  
        "extract_signatures": 238,  
        "minio_signature_save": 301  
    }  
}
```

Certificates

You can override the following environment variables:

- `REQUESTS_CA_BUNDLE`- the path to the certificate bundle file used for secure requests

Workers behavior

You can override the following environment variables:

Environment Variable	Definition	Default Value
<code>OCR_WORKER_COUNT</code>	Number of workers	5

Environment Variable	Definition	Default Value
OCR_WORK_QUEUE_TIMEOUT	If no activity has occurred for a certain number of seconds, an attempt will be made to refresh the workers	10

!(INFO)

If no worker is released after `OCR_WORK_QUEUE_TIMEOUT` seconds, the application will verify whether any workers have become unresponsive and need to be restarted.

If none of the workers have died, it means they are likely blocked in some process. In this case, the application will terminate all the workers and shut down itself, hoping that the container will be restarted.

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Plugins setup guides / Reporting setup guide

The reporting plugin is available a docker image, and it has the following dependencies:

Dependencies

- a reporting PostgreSQL instance
- reporting-plugin helm chart - containing cronJob which performs the following actions:
 - reads from FLOWX.AI Engine db
 - writes in the FLOWX.AI Reporting plugin db
- Superset:
 - a Superset PostgreSQL db
 - a Redis instance for caching
 - exposes the UI through an ingress -> host needed

Postgres database

Basic Postgres configuration:

```
postgresql:
  enabled: true
  postgresqlUsername: {{userName}}
  postgresqlPassword: ""
  postgresqlDatabase: "reporting"
  existingSecret: {{secretName}}
  persistence:
    enabled: true
    storageClass: standard-rwo
    size: 5Gi
  resources:
    limits:
      cpu: 1000m
      memory: 1024Mi
    requests:
```

```
memory: 256Mi
cpu: 100m
metrics:
  enabled: true
  serviceMonitor:
    enabled: false
  prometheusRule:
    enabled: false
primary:
  nodeSelector:
    preemptible: "false"
```

Reporting plugin helm chart (containing CRON)

reporting-plugin helm.yaml

```
sync:
  cronjob:
    image:
      repository: {{env}}/reporting-plugin

    schedule: "*/5 * * * *"

  extraEnvVarsMultipleSecretsCustomKeys:
    - name: process-engine-application-config
      secrets:
        ENGINE_DATABASE_PASSWORD: {{db password}}
      secrets:
        REPORTING_DATABASE_PASSWORD: {{db password}}

  env:
    ENGINE_DATABASE_USER: {{engine db user}}
```

```
ENGINE_DATABASE_URL: {{engine db URL}}
ENGINE_DATABASE_NAME: {{engine db name}}
```



```
REPORTING_DATABASE_USER: {{reporting db user}}
REPORTING_DATABASE_URL: {{reporting db URL}}
REPORTING_DATABASE_NAME: {{reporting db name}}
```

Superset

» [Superset configuration](#)

» [Superset documentation](#)

After installation

- datasource URL -> FLOWX.AI Reporting database
- Datasets
- Dashboards

Datasource configuration

To store data related to document templates and documents the service uses a Postgres database.

The following configuration details need to be added using environment variables:

SPRING_DATASOURCE_URL**SPRING_DATASOURCE_USERNAME****SPRING_DATASOURCE_PASSWORD**

You will need to make sure that the user, password, connection link and db name are configured correctly, otherwise you will receive errors at start time.

The datasource is configured automatically via a liquibase script inside the service. All updates will include migration scripts.

! INFO

Database schema is managed by a liquibase script that will create, manage and migrate future versions.

Redis configuration

The following values should be set with the corresponding Redis-related values:

SPRING_REDIS_HOST**SPRING_REDIS_PORT**

Keycloak configuration

To enable a different user authentication than the regular one (database), you need to override the **AUTH_TYPE** parameter in your **superset .yml** file.

It would look something like this:

```
AUTH_TYPE: AUTH_OID
```

You will also need to provide a reference to your `openid-connect` realm:

```
OIDC_OPENID_REALM: 'flowx'
```

With this configuration, the login page changes to a prompt where the user can select the desired OpenID provider (in our case keycloak)

Extend the Security Manager

Firstly, you will want to make sure that flask stops using `flask-openid` and starts using `flask-oidc` instead.

To do so, you will need to create your own security manager that configures `flask-oidc` as its authentication provider.

```
extraSecrets:  
keycloak_security_manager.py: |  
    from flask_appbuilder.security.manager import AUTH_OID  
    from superset.security import SupersetSecurityManager  
    from flask_oidc import OpenIDConnect
```

To enable OpenID in Superset, you would previously have had to set the authentication type to `AUTH_OID`.

The security manager still executes all the behavior of the super class, but overrides the OID attribute with the `OpenIDConnect` object.

Further, it replaces the default OpenID authentication view with a custom one:

```
from flask_appbuilder.security.views import AuthOIDView
from flask_login import login_user
from urllib.parse import quote
from flask_appbuilder.views import expose
from flask import request, redirect

class AuthOIDCView(AuthOIDView):
    @expose('/login/', methods=['GET', 'POST'])
    def login(self, flag=True):
        sm = self.appbuilder.sm
        oidc = sm.oid
        superset_roles = ["Admin", "Alpha", "Gamma",
"Public", "granter", "sql_lab"]
        default_role = "Admin"
        @self.appbuilder.sm.oid.require_login
        def handle_login():
            user =
sm.auth_user_oid(oidc.user_getfield('email'))
            if user is None:
                info =
oidc.user_getinfo(['preferred_username', 'given_name',
'family_name', 'email', 'roles'])
                roles = [role for role in superset_roles
if role in info.get('roles', [])]
                roles += [default_role, ] if not roles
            else []:
                user =
sm.add_user(info.get('preferred_username'),
info.get('given_name', ''), info.get('family_name', ''),
info.get('email'),
[sm.find_role(role) for role in roles])
                login_user(user, remember=False)
```

```
        return
    redirect(self.appbuilder.get_url_for_index)
        return handle_login()
    @expose('/logout/', methods=['GET', 'POST'])
    def logout(self):
        oidc = self.appbuilder.sm.oid
        oidc.logout()
        super(AuthOIDCView, self).logout()
        redirect_url = request.url_root.strip('/')
        # redirect_url = request.url_root.strip('/') +
        self.appbuilder.get_url_for_login
        return redirect(
            oidc.client_secrets.get('issuer') +
            '/protocol/openid-connect/logout?redirect_uri=' +
            quote(redirect_url))
```

On authentication, the user is redirected back to Superset.

Configure Superset

Finally, we need to add some parameters to the superset .yml file:

```
...
-----
-----KEYCLOACK-----
-----
...
curr  = os.path.abspath(os.getcwd())
AUTH_TYPE = AUTH_OID
OIDC_CLIENT_SECRETS = curr +
'/pythonpath/client_secret.json'
OIDC_ID_TOKEN_COOKIE_SECURE = True
OIDC_REQUIRE_VERIFIED_EMAIL = True
OIDC_OPENID_REALM: 'flowx'
```

```
OIDC_INTROSPECTION_AUTH_METHOD: 'client_secret_post'  
CUSTOM_SECURITY_MANAGER = OIDCSecurityManager  
AUTH_USER_REGISTRATION = False  
AUTH_USER_REGISTRATION_ROLE = 'Admin'  
OVERWRITE_REDIRECT_URI = 'https://{{  
.Values.flowx.ingress.reporting }}/oidc_callback'  
...  
-----  
...  
-----
```

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / Plugins setup guides / Task Manager plugin setup / Configuring access rights for Task management

Granular access rights can be configured for restricting access to the Task management plugin component.

Two different access authorizations are provided, each with specified access scopes:

1. **manage-tasks** - for configuring access for viewing the tasks lists

Available scopes:

- read - users are able to view tasks

2. **manage-hooks** - for configuring access for managing hooks

Available scopes:

- import - users are able to import hooks
- read - users are able to view hooks
- edit - users are able to edit hooks
- admin - users are able to delete hooks

3. **manage-process-allocation-settings** - for configuring access for managing process allocation settings

Available scopes:

- import - users are able to import allocation rules
- read - users are able to read/export allocation rules
- edit - users are able to edit access - create/edit allocation rules
- admin - users are able to delete allocation rules

4. **manage-out-of-office-users** - for configuring access for managing out-of-office users

Available scopes:

- read - users are able to view out-of-office records
- edit - users are able to create and edit out-of-office records
- admin - users are able to delete out-of-office records

The Task management plugin is preconfigured with the following default users roles for each of the access scopes mentioned above:

- manage-tasks
 - read:
 - ROLE_TASK_MANAGER_TASKS_READ
- manage-hooks
 - import:
 - ROLE_TASK_MANAGER_HOOKS_IMPORT
 - ROLE_TASK_MANAGER_HOOKS_EDIT
 - ROLE_TASK_MANAGER_HOOKS_ADMIN
 - read:
 - ROLE_TASK_MANAGER_HOOKS_READ
 - ROLE_TASK_MANAGER_HOOKS_IMPORT
 - ROLE_TASK_MANAGER_HOOKS_EDIT
 - ROLE_TASK_MANAGER_HOOKS_ADMIN
 - edit:
 - ROLE_TASK_MANAGER_HOOKS_EDIT
 - ROLE_TASK_MANAGER_HOOKS_ADMIN
 - admin:
 - ROLE_TASK_MANAGER_HOOKS_ADMIN
- manage-process-allocation-settings
 - import:
 - ROLE_TASK_MANAGER_PROCESS_ALLOCATION_SETTINGS_IMPORT
 - ROLE_TASK_MANAGER_PROCESS_ALLOCATION_SETTINGS_EDIT

- ROLE_TASK_MANAGER_PROCESS_ALLOCATION_SETTINGS_ADMIN
 - read:
 - ROLE_TASK_MANAGER_PROCESS_ALLOCATION_SETTINGS_READ
 - ROLE_TASK_MANAGER_PROCESS_ALLOCATION_SETTINGS_IMPORT
 - ROLE_TASK_MANAGER_PROCESS_ALLOCATION_SETTINGS_EDIT
 - ROLE_TASK_MANAGER_PROCESS_ALLOCATION_SETTINGS_ADMIN
 - edit:
 - ROLE_TASK_MANAGER_PROCESS_ALLOCATION_SETTINGS_EDIT
 - ROLE_TASK_MANAGER_PROCESS_ALLOCATION_SETTINGS_ADMIN
 - admin:
 - ROLE_TASK_MANAGER_PROCESS_ALLOCATION_SETTINGS_ADMIN
- manage-out-of-office-users
 - read:
 - ROLE_TASK_MANAGER_OOO_READ
 - ROLE_TASK_MANAGER_OOO_EDIT
 - ROLE_TASK_MANAGER_OOO_ADMIN
 - edit:
 - ROLE_TASK_MANAGER_OOO_EDIT
 - ROLE_TASK_MANAGER_OOO_ADMIN

- admin:
 - ROLE_TASK_MANAGER_OOO_ADMIN

⚠ CAUTION

These roles need to be defined in the chosen identity provider solution.

In case other custom roles are needed, you can configure them using environment variables. More than one role can be set for each access scope.

To configure access for each of the roles above, adapt the following input:

```
SECURITY_ACCESSAUTHORIZATIONS_AUTHORIZATIONNAME_SCOPES_SCOPENAME_ROLESALLOWED: NEEDED_ROLE_NAMES
```

Possible values for AUTHORIZATIONNAME: MANAGETASKS, MANAGEHOOKS.

Possible values for SCOPENAME: import, read, edit, admin.

For example, if you need to configure role access for read, insert this:

```
SECURITY_ACCESSAUTHORIZATIONS_MANAGEHOOKS_SCOPES_READ_ROLESALLOWED:ROLE_NAME_TEST
```

Was this page helpful?

PLATFORM DEEP DIVE / Plugins / WYSIWYG editor

FLOWX.AI Designer's WYSIWYG ("What You See Is What You Get") editor enables you to create and modify **notification** and **document** templates without the need for complicated coding from the developers. WYSIWYG editors make the creation/editing of any type of document or notification easier for the end-user.

Displaying how the document will be published or printed on the screen, the user can adjust the text, graphics, photos, or other document/notification elements before generating the final output.

WYSIWYG Components

Header

The formatting head of the editor allows users to manipulate/format the content of the document.

Body

The Body is the main part of the editor where you can edit your template.

ⓘ INFO

After you defined some parameters in the **Data Model** tab, you can type "#" in the body to trigger a dropdown where you can choose which one you want to use.

Source

The **Source** button can be used to switch to the HTML editor. You can use the HTML view/editor as a debugging tool, or you can edit the template directly by

writing code here.

The screenshot shows the FLOWX.AI platform interface. On the left is a sidebar with navigation links: Processes (Definitions, Active process), Content Management (Enumerations, Substitution tags, Content models, Languages, Source systems), Plugins (Task Manager, All tasks, Hooks, Stages, Notification templates, Document templates), General Settings (Generic Parameters), and Platform status. The main area is titled "Documents Templates - ExampleTemplate". It has tabs for "Body" (selected) and "Data model". The "Body" tab shows a rich text editor toolbar and a preview area containing placeholder text "#firstInput" and "#secondInput". Below the preview is the generated HTML code. The "Data model" tab is currently empty. At the bottom are "Test", "Save", and "Publish" buttons. A "Logout" link is at the bottom left.

Document Templates

One of the main features of the **document management plugin** is the ability to generate new documents based on custom templates and prefilled with data related to the current process instance.

The screenshot shows the FLOWX.AI platform interface. On the left is a sidebar with navigation links: Processes (Definitions, Active process), Content Management (Enumerations, Substitution tags, Content models, Languages, Source systems), Plugins (Task Manager, Hooks, Stages, Notification templates, Document templates), General Settings (Generic Parameters), and Platform status. At the bottom of the sidebar is a Logout link. The main area is titled "Documents Templates - ExampleTemplate". It has tabs for "Body" (selected) and "Data model". The "Body" tab shows a rich text editor toolbar and a preview area containing placeholder text "#firstInput" and "#secondInput", followed by a long block of placeholder text. Below the preview is a code editor showing the corresponding HTML and CSS. At the bottom are buttons for "Test", "Save", and "Publish".

» Documents plugin

Notification Templates

Notification WYSIWYG body has some additional fields (other than documents template):

- **Type** - that could be either MAIL or SMS (SMS, only if there is an external adapter)
- **Forward on Kafka** - if this box is checked, the notification is not being sent directly by the plugin to the destination, but forwarded to another adapter

The screenshot shows the FLOWX.AI platform interface for creating a notification template. On the left, a sidebar navigation includes: Processes (Definitions, Active process), Content Management (Enumerations, Substitution tags, Content models, Languages, Source systems), Plugins (Task Manager, All tasks, Hooks, Stages), Notification templates (selected), Document templates, General Settings (Generic Parameters), and Platform status. At the bottom left is a Logout link.

The main workspace is titled "Notifications Template - ExampleTemplate". It has tabs for "Body" (selected) and "Data model". Under "Body", settings include "Type" (MAIL selected), a checkbox for "Forward on Kafka" (unchecked), and a "Language" dropdown set to "Romanian (Romania)-ro-RO". Below these are sections for "Romanian (Romania)-ro-RO subject" (Contract [\${firstInput}]) and "Content" (WYSIWYG editor). The WYSIWYG editor toolbar includes standard text and style tools. The content area contains a header with the FLOWX.AI logo, a greeting "Salut", a message "Salut #firstInput #secondInput, ne bucurăm sa te avem alături!", and a footer "Găsești toate detaliele în brosura atașata acestui email." There are three yellow placeholder boxes at the bottom. At the bottom right are "Save" and "Publish" buttons.

» Managing notifications templates

Data Model

Data Model

Using the data model, you can define key pair values (parameters) that will be displayed and reused in the body. Multiple parameters can be added:

- STRING
- NUMBER
- BOOLEAN
- OBJECT
- ARRAY (which has an additional `item` field)

Documents Templates - ExampleTemplate

Body Data model

Attributes

Name	firstInput
Item type	STRING
Mandatory	<input checked="" type="checkbox"/>

New value

Actions

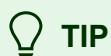
Close **Save**

! INFO

Parameters can be defined as mandatory or not. When you try to generate a template without filling in all the mandatory parameters, the following error message will be displayed: "*Provided data cannot be empty if there are any required properties defined.*"

Was this page helpful?

PLATFORM DEEP DIVE / Integrations / Creating a Kafka consumer



This guide focuses on creating a

The fallback content to display on prerendering consumer using Spring Boot.

Here are some tips, including the required configurations and code samples, to help you implement a Kafka consumer in Java.

Required dependencies

Ensure that you have the following dependencies in your project:

```
<dependency>
    <groupId>org.springframework.kafka</groupId>
    <artifactId>spring-kafka</artifactId>
</dependency>

<dependency>
    <groupId>io.strimzi</groupId>
    <artifactId>kafka-oauth-client</artifactId>
    <version>0.6.1</version>
</dependency>
```

```
<dependency>
    <groupId>org.apache.kafka</groupId>
    <artifactId>kafka-clients</artifactId>
    <version>2.5.1</version>
</dependency>

<dependency>
    <groupId>io.opentracing.contrib</groupId>
    <artifactId>opentracing-kafka-client</artifactId>
    <version>0.1.13</version>
</dependency>
```

Configuration

Ensure that you have the following configuration in your `application.yml` or `application.properties` file:

```
spring.kafka:
  bootstrap-servers: URL_OF_THE_KAFKA_SERVER
  consumer:
    group-id: ADD_CONSUMER_NAME
    auto-offset-reset: earliest
    key-deserializer:
      org.apache.kafka.common.serialization.StringDeserializer
    value-deserializer:
      org.apache.kafka.common.serialization.StringDeserializer
    properties:
      interceptor:
        classes:
          io.opentracing.contrib.kafka.TracingConsumerInterceptor
      security.protocol: "SASL_PLAINTEXT"
      sasl.mechanism: "OAUTHBEARER"
```

```
sasl.jaas.config:  
"org.apache.kafka.common.security.oauthbearer.OAuthBearerLogin  
required ;"  
    sasl.login.callback.handler.class:  
io.strimzi.kafka.oauth.client.JaasClientOauthLoginCallbackHand  
  
kafka:  
    consumerThreads: 1  
    authorizationExceptionRetryInterval: 10  
    ADD_NEEDED_TOPIC_NAMES_HERE
```

Code sample for a Kafka Listener

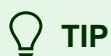
Here's an example of a Kafka listener method:

```
@KafkaListener(topics = "TOPIC_NAME_HERE")  
public void listen(ConsumerRecord<String, String> record)  
throws JsonProcessingException {  
  
    SomeDTO request = objectMapper.readValue(record.value(),  
    SomeDTO.class);  
  
    // process received DTO  
}
```

Make sure to replace "TOPIC_NAME_HERE" with the actual name of the Kafka topic you want to consume from. Additionally, ensure that you have the necessary serialization and deserialization logic based on your specific use case.

Was this page helpful?

PLATFORM DEEP DIVE / Integrations / Creating a Kafka producer



This guide focuses on creating a

The fallback content to display on prerendering
producer using Spring Boot.

Here are some tips, including the required configurations and code samples, to help you implement a Kafka producer in Java.

Required dependencies

Ensure that you have the following dependencies in your project:

```
<dependency>
    <groupId>org.springframework.kafka</groupId>
    <artifactId>spring-kafka</artifactId>
</dependency>

<dependency>
    <groupId>io.strimzi</groupId>
    <artifactId>kafka-oauth-client</artifactId>
    <version>0.6.1</version>
</dependency>
```

```
<dependency>
    <groupId>org.apache.kafka</groupId>
    <artifactId>kafka-clients</artifactId>
    <version>2.5.1</version>
</dependency>

<dependency>
    <groupId>io.opentracing.contrib</groupId>
    <artifactId>opentracing-kafka-client</artifactId>
    <version>0.1.13</version>
</dependency>
```

Configuration

Ensure that you have the following configuration in your `application.yml` or `application.properties` file:

```
spring.kafka:
  bootstrap-servers: URL_OF_THE_KAFKA_SERVER
  producer:
    key-deserializer:
      org.apache.kafka.common.serialization.StringSerializer
    value-serializer:
      org.springframework.kafka.support.serializer.JsonSerializer
    properties:
      interceptor:
        classes:
          io.opentracing.contrib.kafka.TracingProducerInterceptor
        security.protocol: "SASL_PLAINTEXT"
        sasl.mechanism: "OAUTHBEARER"
        sasl.jaas.config:
          "org.apache.kafka.common.security.oauthbearer.OAuthBearerLogin"
```

```
required ;"
    sasl.login.callback.handler.class:
io.strimzi.kafka.oauth.client.JaasClientOauthLoginCallbackHand

kafka:
    authorizationExceptionRetryInterval: 10
    ADD_NEEDED_TOPIC_NAMES_HERE # make sure to use the correct name
pattern for topics used to send data to the FLOWX Engine
```

Code sample for a Kafka producer

🔥 DANGER

Ensure that you have the necessary KafkaTemplate bean autowired in your producer class. The sendMessage method demonstrates how to send a message to a Kafka topic with the specified headers and payload. Make sure to include all the received Kafka headers in the response that is sent back to the

The fallback content to display on prerendering

```
private final KafkaTemplate<String, Object> kafkaTemplate;

public void sendMessage(String topic, Headers headers,
Object payload) {
    ProducerRecord<String, Object> producerRecord = new
    ProducerRecord<>(topic, payload);
    // make sure to send all the received headers back to the
    FlowX Engine
```

```
headers.forEach(header ->
producerRecord.headers().add(header));
kafkaTemplate.send(producerRecord);
}
```

Was this page helpful?

PLATFORM DEEP DIVE / Integrations / Jaeger setup for microservices

The scope of this document is to present some basic information on how to include Jaeger tracing into a Java based project.

Required dependencies

```
<dependency>
    <groupId>io.jaegertracing</groupId>
    <artifactId>jaeger-client</artifactId>
    <version>1.4.0</version>
</dependency>
<dependency>
    <groupId>io.opentracing.contrib</groupId>
    <artifactId>opentracing-kafka-client</artifactId>
    <version>0.1.13</version>
</dependency>
```

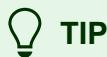
Needed configs

Add Kafka interceptors for Tracing

```
kafka:  
  producer:  
    properties:  
      interceptor:  
        classes:  
          io.opentracing.contrib.kafka.TracingProducerInterceptor  
  
  consumer:  
    properties:  
      interceptor:  
        classes:  
          io.opentracing.contrib.kafka.TracingConsumerInterceptor
```

Extract Jaeger span context from received Kafka message

```
@KafkaListener(topics = "${TOPIC_NAME}")  
public void listen(ConsumerRecord<String, String> record) {  
    // some code  
    SpanContext spanContext =  
    TracingKafkaUtils.extractSpanContext(record.headers(),  
    tracer);  
    // some other code  
}
```



Use this context to create child spans of it and log events from adapter:

```
Span span =  
tracer.buildSpan(JAEGER_SPAN_NAME).asChildOf(spanContext).
```

Send span context with outgoing Kafka messages

```
ProducerRecord<String, Object> producerRecord = new  
ProducerRecord<>(responseTopic, responseMessage);  
  
TracingKafkaUtils.inject(span.context(),  
producerRecord.headers(), tracer);  
  
kafkaTemplate.send(producerRecord);
```

Was this page helpful?

PLATFORM DEEP DIVE / Integrations / Mock integrations

If you need to test the business process flow but haven't completed all integrations, you can still do so by utilizing the mock integrations server included in the platform.

Setup

To begin, configure the microservice's DB settings to use a Postgres DB. Then, deploy the mocked adapter microservice.

Adding a new integration

Setting up a mocked integration requires only one step: adding a mock Kafka request and response.

You have two options for accomplishing this:

1. Add the information directly to the DB.
2. Use the provided API.

For each Kafka message exchange between the engine and the integration, you need to create a separate entry.

▶ **POST** `MOCK_ADAPTER_URL/api/kafka-exchanges/`

▶ **GET** `MOCK_ADAPTER_URL/api/kafka-exchanges/`

Was this page helpful?

PLATFORM DEEP DIVE / Third-party components

FLOWX.AI uses a number of third-party software components:

Open-source

- Keycloak
- Kafka / ZooKeeper
- Jaeger
- AKHQ
- PostgreSQL
- MongoDB
- Redis
- NGINX
- EFK (Elastic Search, Fluentd, Kibana)
- S3 (MinIO)
- RabbitMQ (for OCR plugin)

Not open-source

- OracleDB

Third-party open-source components supported/tested versions

⚠️ COMPATIBILITY

FlowX.AI supports any version of the third-party components listed as prerequisites.

For optimal performance and reliability, our internal QA process validates new releases using specific versions as indicated in the provided table. While exploring alternative versions that suit your company's specific requirements, we recommend referring to the compatibility matrix for guidance.

In the unlikely event that you encounter any compatibility issues with FlowX.AI, please open a support ticket [here](#), and our dedicated team will address and resolve any identified bugs following our standard support process.

Compatibility Matrix:

- FLOWX.AI Platform: Recommended and tested versions
- Third-Party Components: Supported versions based on specific requirements and client preferences

FLOWX.AI Platform Version	Component name	Supported/tested versions
2.3.0 → 3.3.0	Keycloak	18.0.x
2.3.0 → 3.3.0	Kafka / Zookeeper*	3.0.1 / 3.6.6
2.3.0 → 3.3.0	Jaeger	1.34.1
2.3.0 → 3.3.0	AKHQ	0.17.0
2.3.0 → 3.3.0	PostgreSQL	14.3.0

FLOWX.AI Platform Version	Component name	Supported/tested versions
2.3.0 → 3.3.0	MongoDB	5.0.8
2.3.0 → 3.3.0	Redis	6.2.6
2.3.0 → 3.3.0	NGINX Ingress Controller	1.2.0
2.3.0 → 3.3.0	Elasticsearch	7.17
2.3.0 → 3.3.0	Fluentd	3.3.0
2.3.0 → 3.3.0	Kibana	7.17
2.3.0 → 3.3.0	S3 (Min.IO) / minio-operator	2022-05-26T05-48-41Z / 4.5.4

Third-party components supported/tested versions

FLOWX.AI Platform version	Component name	Supported/tested versions
2.3.0 → 3.3.0	OracleDB	12C / 18-XE



Since Kafka version 2.8, the self-managed (Kraft) mode is available alongside ZooKeeper. It was released as a preview feature in version 3.0. Finally, with several improvements, it has been declared production ready in version 3.3.1. Kafka may deprecate ZooKeeper in version 3.4.

Summary

Keycloak

Keycloak is an open-source software product to allow single sign-on with Identity and Access Management aimed at modern applications and services.

» [Keycloak documentation](#)

Kafka

Apache Kafka is an open-source distributed event streaming platform that can handle a high volume of data and enables you to pass messages from one endpoint to another.

Kafka is a unified platform for handling all the real-time data feeds. Kafka supports low latency message delivery and gives a guarantee for fault tolerance in the presence of machine failures. It has the ability to handle a large number of diverse consumers.

Kafka is very fast and performs 2 million writes/sec. Kafka persists all data to the disk, which essentially means that all the writes go to the page cache of the OS

(RAM). This makes it very efficient to transfer data from a page cache to a network socket.

» [Intro to Kafka](#)

» [Kafka documentation](#)

ZooKeeper

ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. All of these kinds of services are used in some form or another by distributed applications.

» [Zookeeper documentation](#)

Jaeger

Jaeger is a popular open-source distributed tracing tool that is used to monitor and troubleshoot applications based on microservices architecture.

» [Jaeger documentation](#)

AKHQ

AKHQ is a tool used by FLOWX.AI to manage and display the data inside the Apache Kafka cluster.

» [AKHQ documentation](#)

PostrgreSQL

PostgreSQL, also known as Postgres, is a free and open-source relational database management system emphasizing extensibility and SQL compliance.

» [PostgreSQL documentation](#)

MongoDB

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional [schemas](#).

Used by FLOWX.AI to store business process data and configuration information on the core/plugin components.

» [MongoDB documentation](#)

Redis

Redis is a fast, open-source, in-memory key-value data store that is commonly used as a cache to store frequently accessed data in memory so that applications

can be responsive to users.

It delivers sub-millisecond response times enabling millions of requests per second for applications.

It is also be used as a Pub/Sub messaging solution, allowing messages to be passed to channels and for all subscribers to that channel to receive that message. This feature enables information to flow quickly through the platform without using up space in the database as messages are not stored.

It is used by FLOWX.AI for caching the process definitions-related data.

» [Intro to Redis](#)

» [Redis documentation](#)

NGINX

Nginx Is a web server that can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache.

FLOWX utilizes the Nginx engine as a load balancer and for routing the web traffic (API calls) from the SPA (single page application) to the backend service, to the engine, and to various plugins.

The FLOWX.AI Designer SPA will use the backend service to manage the platform via REST calls, will use API calls to manage specific content for the plugins, and will use REST and SSE calls to connect to the engine.

» [Intro to NGINX](#)

» [NGINX documentation](#)

EFK (Kibana, fluentd, Elastic Search)

Elasticsearch is a distributed, RESTful search and analytics engine capable of addressing a growing number of use cases.

As the heart of the Elastic Stack, it centrally stores your data for lightning-fast search, fine-tuned relevancy, and powerful analytics that scale with ease.

Used by FLOWX.AI in the core component and optionally to allow searching for business process transaction data.

» [Elastic stack documentation](#)

» [Fluentd documentation](#)

Kafka Connect Elasticsearch Service Sink

The Kafka Connect Elasticsearch Service Sink connector moves data from Apache Kafka® to Elasticsearch. It writes data from a topic in Kafka to an index in Elasticsearch. All data for a topic have the same type in Elasticsearch. This allows an independent evolution of schemas for data from different topics. This simplifies

the schema evolution because Elasticsearch has one enforcement on mappings; that is, all fields with the same name in the same index must have the same mapping type.

S3 (MinIO)

FLOWX.AI uses [Min.IO](#) as a cloud storage solution.

» [MIN.IO documentation](#)

» [Docker available here](#)

Oracle DB

Oracle Database is a relational database management system (RDBMS).

» [Oracle DB documentation](#)

Superset

Apache Superset is a business intelligence web application. It helps users to explore and visualize their data, from simple pie charts to detailed dashboards.

» [Superset](#)

Was this page helpful?

PLATFORM DEEP DIVE / User roles management / Swimlanes

ⓘ INFO

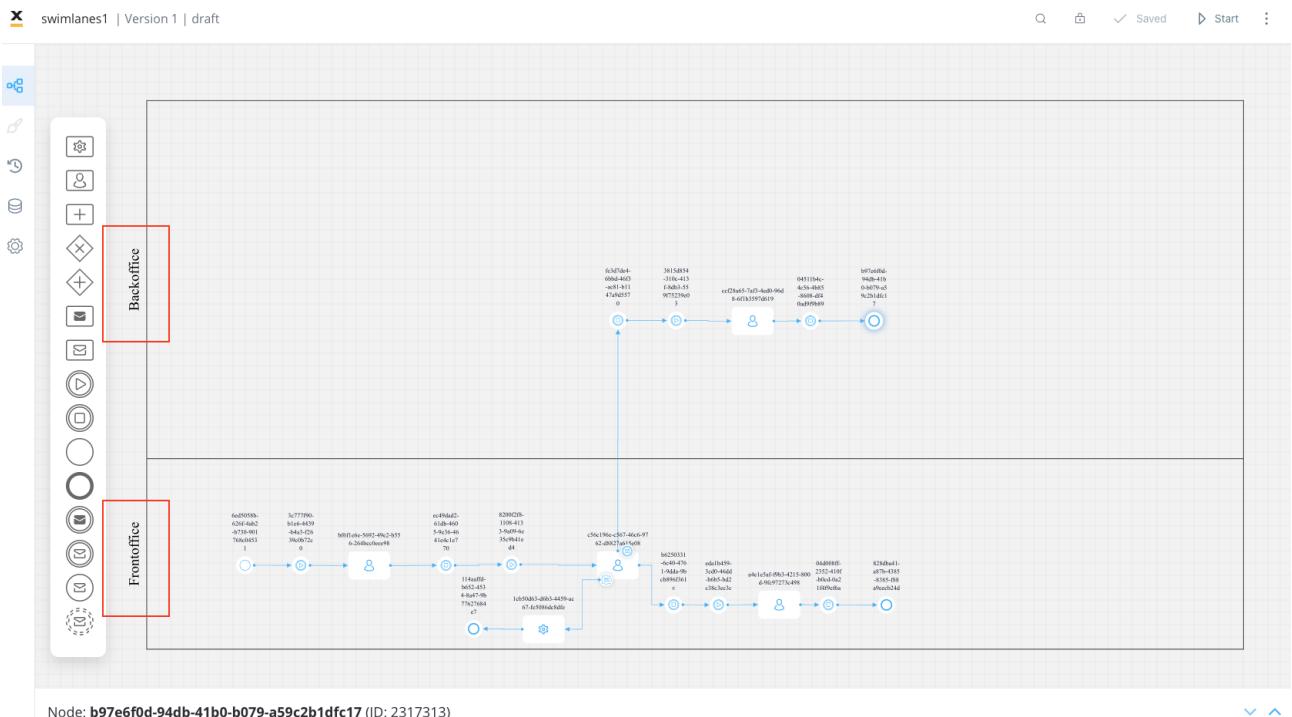
What is it? Swimlanes provide a way of grouping process nodes by process participants.

Why is it useful? Using swimlanes we can make sure only certain user roles have access to certain process nodes.

In certain scenarios, it is necessary to restrict access to specific process

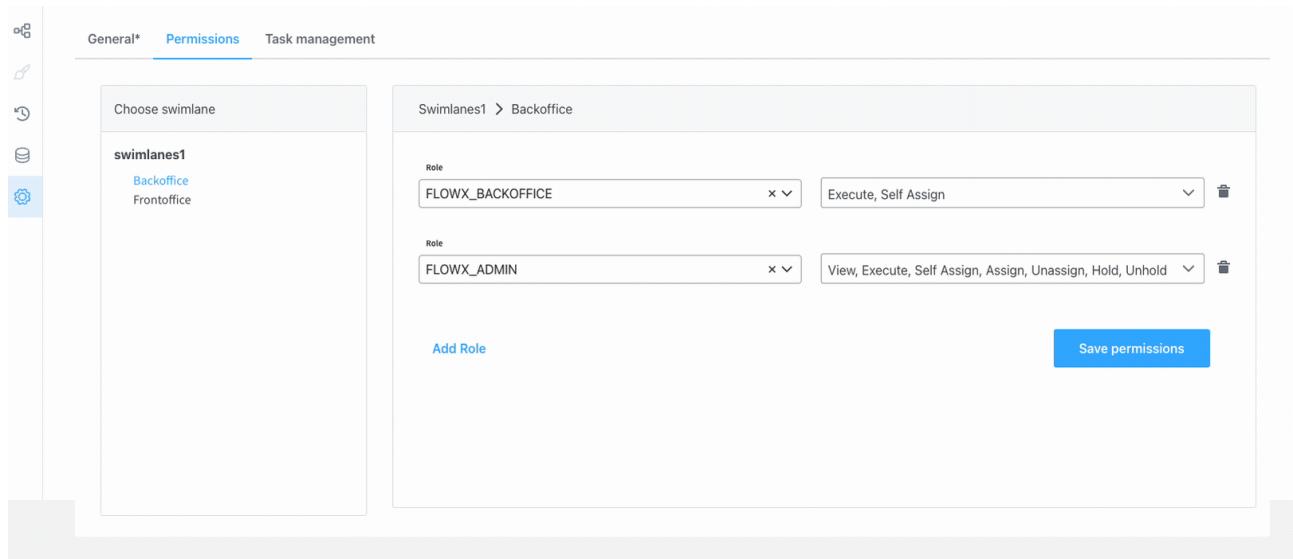
The fallback content to display on prerendering based on user roles. This can be achieved by organizing nodes into different swimlanes.

Each swimlane can be configured to grant access only to users with specific roles defined in the chosen identity provider platform.



Depending on the type of node added within a swimlane, only users with the corresponding swimlane roles will have the ability to initiate process instances, view process instances, and perform actions on them.

» Click here to view the list of scopes and roles for managing processes



When creating a new process definition, a default swimlane will automatically be added.

Name	Version	Edited at	Edited by
Loan_application_processing	1	24 May 2023, 1:35 PM	admin flowx
m_35	1	24 May 2023, 9:26 AM	QA FlowX

Name	Version	Published at	Published by
AutoTestProcess2052253127	1	07 Apr 2023, 9:35 AM	QA FlowX
AutoTestProcess-888325220	1	07 Apr 2023, 9:34 AM	QA FlowX

As the token moves from one node to the next, it may transition between swimlanes. If a user interacting with the process instance no longer has access to

the new swimlane, they will observe the process in read-only mode and will be unable to interact with it until the token returns to a swimlane they have access to.

Users will receive notifications when they can no longer interact with the process or when they can resume actions on it.

» [Configuring access roles for processes](#)

[Was this page helpful?](#)

PLATFORM DEEP DIVE / User roles management / Business filters

INFO

What is it? An optional attribute, from the authorization token, that can be set in order to restrict access to process instances based on a business specific value (ex. bank branch name).

Why is it useful? Using business filters we can make sure only the allowed users, with the same attribute, can access a

The fallback content to display on prerendering

In some cases it might be necessary to restrict access to process nodes based on certain

The fallback content to display on prerendering , for example only users from a specific bank branch can view the process instances started from that branch. This can be done by using business filters.

Before they can be used in the process definition the business filter attributes need to be set in the identity management platform. They have to be configured as a list of filters and should be made available on the authorization token. Application users will also have to be assigned this value.

When this filter needs to be applied, the process definition should include nodes with actions that will store the current business filter value to a custom `task.businessFilters` key on process parameters.

If this value is set in the process instance parameters, only users that have the correct business filter attribute will be able to interact with that process instance.

Was this page helpful?