
FLOW^X.AI

FLOWX.AI

Contents

- What is FLOWX.AI?
 - Why does it matter?
 - Next steps
- GETTING STARTED / Building your first process
 - Prerequisites
 - Designing the BPMN process: request a new credit card from a bank app
 - Sample process steps
 - Sample process diagram
- GETTING STARTED / Learn more
 - Additional support
- PLATFORM OVERVIEW / Frameworks and standards / Business process industry and standards / Intro to BPMN / BPMN basic concepts
- PLATFORM OVERVIEW / Frameworks and standards / Business process industry and standards / Intro to DMN
- PLATFORM OVERVIEW / Frameworks and standards / Business process industry and standards / Intro to MVEL
 - What is MVEL?
 - Example
 - In depth docs
- PLATFORM OVERVIEW / Frameworks and standards / Event-driven architecture frameworks / Intro to Elasticsearch
 - What is Elasticsearch?
 - How it works?
 - Why it is useful?
 - Indexing & sharding
 - Indexing
 - Sharding

- Leveraging Elasticsearch for advanced indexing with FLOWX.AI
 - Kafka transport strategy
- PLATFORM OVERVIEW / Frameworks and standards / Event-driven architecture frameworks / Intro to Kafka concepts
 - Key Kafka concepts
 - Events
 - Topics
 - Producer
 - Consumer
 - In-depth docs
- PLATFORM OVERVIEW / Frameworks and standards / Event-driven architecture frameworks / Intro to Kubernetes
- PLATFORM OVERVIEW / Frameworks and standards / Event-driven architecture frameworks / Intro to NGINX
- PLATFORM OVERVIEW / Frameworks and standards / Event-driven architecture frameworks / Intro to Redis
 - In depth docs
- PLATFORM OVERVIEW / Frameworks and standards / Timer expressions
- PLATFORM OVERVIEW / FLOWX.AI architecture
- BUILDING BLOCKS / Process Designer / Process definition
 - History
 - Versions
 - Audit log
 - Data model
 - Attributes type
 - Data model reference
 - Sensitive data
 - Reporting
 - Generating data model

- Swimlanes
- Settings
 - General
 - Permissions
 - Task management
- BUILDING BLOCKS / Process Designer / Active process / Process instance
 - Overview
 - Checking the Process Status
 - Understanding the Process Status Data
 - Process menu
 - Color coding
 - Starting a new process instance
 - Troubleshooting possible errors
- BUILDING BLOCKS / Process Designer / Active process / Failed process start
 - Exceptions
 - Exceptions data
 - Exceptions type
- BUILDING BLOCKS / Process Designer / Subprocess
 - Configuring & starting subprocesses
 - Executing subprocesses
- BUILDING BLOCKS / Node / Start/End nodes
 - Start node
 - Configuring a start node
 - End node
 - Configuring an end node
- BUILDING BLOCKS / Node / Message send/Message received task nodes
 - Message send task
 - Configuring a message send task node

- Example of a message send event
- Message receive task
 - Configuring a message receive task node
- BUILDING BLOCKS / Node / Task node
 - Configuring task nodes
 - Configuring task nodes actions
 - Business Rule action
 - Send data to user interface
 - Upload File action
 - Start Subprocess action
 - Append Params to Parent Process
- BUILDING BLOCKS / Node / User task node
 - Configuring a user task node
 - Configuring the UI
 - Accessing the UI Designer
 - Predefined components
 - Custom components
 - Displaying a UI element
 - Values
- BUILDING BLOCKS / Node / Exclusive gateway
 - Configuring an Exclusive gateway node
- BUILDING BLOCKS / Node / Parallel gateway
 - Configuring a Parallel gateway node
- BUILDING BLOCKS / Node / Milestone node
 - Configuring a Milestone node
 - Available Components
 - Modal
 - Page
 - Stepper + Steps

- Container
- BUILDING BLOCKS / Node / Subprocess run node
- BUILDING BLOCKS / Node / Message events / Message Throw Intermediate Event
 - Configuring a Message Throw Intermediate Event
- BUILDING BLOCKS / Node / Message events / Message Catch Boundary Events
 - Message Catch Interrupting Event
 - Message Catch Non-Interrupting Event
 - Configuring a Message Catch Interrupting/Non-Interrupting Event
- BUILDING BLOCKS / Node / Message events / Message Catch Intermediate Event
 - Configuring a Message Catch Intermediate Event
- BUILDING BLOCKS / Node / Message events / Message Catch Start Event
 - Configuring a Message Catch Start Event
- BUILDING BLOCKS / Actions / Business Rule action / DMN Business Rule action
- BUILDING BLOCKS / Actions / Send data to user interface
- BUILDING BLOCKS / Actions / Upload File action
- BUILDING BLOCKS / Actions / Start Subprocess action
 - Configuring a Start Subprocess action
 - Action Edit
 - Back in steps
 - Parameters
 - Data to send
 - Example
- BUILDING BLOCKS / Actions / Append Params to Parent Process
- BUILDING BLOCKS / UI Designer / UI component types / Root components / Container

- BUILDING BLOCKS / UI Designer / UI component types / Root components / Card
- BUILDING BLOCKS / UI Designer / UI component types / Root components / Custom
- BUILDING BLOCKS / UI Designer / UI component types / Collection / Collection Prototype
 - Description
 - Configurable properties:
 - Example
 - Adding elements with UI Actions
 - Step 1 - Defining the Node Action
 - Step 2 - Adding the Button & UI Action
 - Result
- BUILDING BLOCKS / UI Designer / UI component types / Buttons
 - Basic button
 - Configuring a basic button
 - Button styling
 - File upload
 - Configuring a file upload button
 - Button styling
- BUILDING BLOCKS / UI Designer / UI component types / File Preview
 - Configuring a File Preview element
 - File Preview properties (web)
 - File Preview properties (mobile)
 - File preview styling
 - File Preview example
- BUILDING BLOCKS / UI Designer / UI component types / Form elements / Input
 - Configuring the input element

- Input settings
 - Input styling
- BUILDING BLOCKS / UI Designer / UI component types / Form elements / Text area
 - Configuring the text area element
 - Text area settings
 - Text area styling
- BUILDING BLOCKS / UI Designer / UI component types / Form elements / Select
 - Configuring the Select element
 - Select Settings
 - Select styling
 - Example - Dynamic dropdowns
 - Creating the process
 - Configuring the nodes
 - Configuring the UI
- BUILDING BLOCKS / UI Designer / UI component types / Form elements / Checkbox
 - Configuring the checkbox element
 - Checkbox settings
 - Checkbox styling
- BUILDING BLOCKS / UI Designer / UI component types / Form elements / Radio
 - Configuring the radio field element
 - Radio settings
 - Radio styling
- BUILDING BLOCKS / UI Designer / UI component types / Form elements / Switch
 - Configuring the switch element

- Switch settings
- BUILDING BLOCKS / UI Designer / UI component types / Form elements / Datepicker
 - Configuring the datepicker element
 - Datepicker settings
 - Datepicker styling
- BUILDING BLOCKS / UI Designer / UI component types / Form elements / Slider
 - Configuring the slider element
 - Slider settings
 - Multiple sliders
 - Slider styling
- BUILDING BLOCKS / UI Designer / UI component types / Form elements / Segmented button
 - Configuring the segmented button
 - Segmented button settings
 - Segmented button styling
- BUILDING BLOCKS / UI Designer / UI component types / Image
 - Configuring an image
 - Image settings
 - Media library
 - Process Data
 - External
 - UI actions
 - Image styling
- BUILDING BLOCKS / UI Designer / UI actions
 - Process UI actions
 - Manual action configuration example - Save Data
 - UI action configuration example

- UI actions elements
 - Events
 - Action types
- External UI actions
- BUILDING BLOCKS / UI Designer / Validators
 - Predefined validators
 - required validator
 - minlength validator
 - maxlength validator
 - min validator
 - max validator
 - email validator
 - pattern validator
 - datepicker - isSameOrBeforeToday
 - datepicker - isSameOrAfterToday
 - Custom validators
 - sidebar_position: 3
- BUILDING BLOCKS / UI Designer / Dynamic & computed values
 - Dynamic values
 - Example using Substitution tags
 - Example using process parameters
 - Computed values
 - Slider example
 - Usage
- BUILDING BLOCKS / UI Designer / Layout configuration
- BUILDING BLOCKS / UI Designer / Rendering and UI Designer changelog
 - Notes for post-migration
- BUILDING BLOCKS / Token
- BUILDING BLOCKS / Supported scripts

- Supported scripts
 - Python
 - DMN
 - MVEL
 - Groovy
 - Nashorn Engine (JavaScript)
- FLOWX.AI DESIGNER / What is the FLOWX Designer?
- FLOWX.AI DESIGNER / Overview
 - Managing process definitions
 - Viewing active process instances
 - Managing CMS
 - Managing tasks
 - Managing notification templates
 - Managing document templates
 - Managing generic parameters
 - Managing users access
 - Managing integrations
 - Checking platform status
- FLOWX.AI DESIGNER / Managing a process flow / Creating a new process definition
- FLOWX.AI DESIGNER / Managing a process flow / Adding a new node
- FLOWX.AI DESIGNER / Managing a process flow / Adding an action to a node
- FLOWX.AI DESIGNER / Managing a process flow / Handling decisions in the flow
- FLOWX.AI DESIGNER / Managing a process flow / Adding more flow branches
- FLOWX.AI DESIGNER / Managing a process flow / Creating a user interface
 - Creating a stepper structure
 - Configuring the UI

- Testing the flow that we have
- Adding a card with one input
 - Testing our first input
 - Adding second input and a submit action
- FLOWX.AI DESIGNER / Managing a process flow / Moving a token backwards in a process
- FLOWX.AI DESIGNER / Managing a process flow / Exporting / importing a process definition
 - Export a process definition
 - Import a process definition
- FLOWX.AI DESIGNER / Managing a process flow / Process definition states & versioning
 - Testing draft versions
- FLOWX.AI DESIGNER / Designer setup guide / Configuring access rights for Admin
- 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 SETUP GUIDES / Overview
 - Environment variables
 - Authorization & access roles
 - Datasource configuration
 - Kafka
 - Redis configuration
 - Debugging
 - Logging
 - Tracing via Jaeger
 - License model
 - Third-party components
- PLATFORM SETUP GUIDES / FLOWX.AI Engine setup guide / Advancing Controller setup guide
 - Infrastructure prerequisites
 - Dependencies
 - Database configuration
 - Configuration
 - Configuring datasource
- PLATFORM SETUP GUIDES / FLOWX.AI Engine setup guide / Configuring access rights for Engine
- PLATFORM SETUP GUIDES / FLOWX.AI Engine setup guide / Configuring access roles for processes

- PLATFORM SETUP GUIDES / FLOWX.AI Engine setup guide / Process instance indexing / Configuration guidelines
 - Indexing strategy
 - Shard and replica configuration
 - Recommendations for resource management
- PLATFORM SETUP GUIDES / FLOWX.AI Engine setup guide / Old access roles
- Old access roles
- PLATFORM SETUP GUIDES / Access management / Configuring an IAM solution
 - Recommended Keycloak setup
 - Creating a new realm
 - Creating/importing user groups and roles
 - Creating new users
 - Adding clients
 - Adding protocol mappers
 - Group Membership mapper
 - User Attribute mapper
 - User realm role
 - Examples
 - Authorizing client
 - Minimal auth config for microservices
 - Adding service accounts
 - Admin service account
 - Task management service account
 - Process engine service account
- PLATFORM SETUP GUIDES / Access management / Default roles
- PLATFORM SETUP GUIDES / Audit setup guide
 - Introduction

- Infrastructure prerequisites
- Dependencies
- Configuration
 - Configuring Kafka
 - Configuring Elasticsearch
 - Configuring logging
- PLATFORM SETUP GUIDES / CMS setup guide / Configuring access rights for CMS
- PLATFORM SETUP GUIDES / Events gateway setup guide
 - Introduction
 - Infrastructure prerequisites
 - Dependencies
 - Configuration
 - Configuring Kafka
 - Configuring authorization & access roles
 - Redis
 - Configuring logging
- PLATFORM SETUP GUIDES / License engine setup guide / Configuring access rights for License
- PLATFORM SETUP GUIDES / License engine setup guide / Configuring access roles
- PLATFORM SETUP GUIDES / Scheduler setup guide
 - Introduction
 - Infrastructure prerequisites
 - Dependencies
 - Dependencies
 - MongoDB helm example
 - Configuration
 - Configuring MongoDB

- Configuring Kafka
- Configuring logging
- PLATFORM SETUP GUIDES / Data search service setup guide
 - Introduction
 - Infrastructure prerequisites
 - Dependencies
 - Configuration
 - Configuring Kafka
 - Configuring Elasticsearch
 - Configuring authorization & access roles
 - Configuring logging
 - Elasticsearch

What is FLOWX.AI?

FLOWX is an AI multi-experience development platform that sits on top of legacy systems and creates **unified, scalable digital experiences**.

- With the world's first AI-generated UI, FLOWX generates in **real-time omnichannel interfaces for customers and employees**.
- It captures and unifies data offering enterprises **AI-based optimization** and innovation capabilities.
- **It integrates easily with any infrastructure** and scales it as necessary.

It is a modern

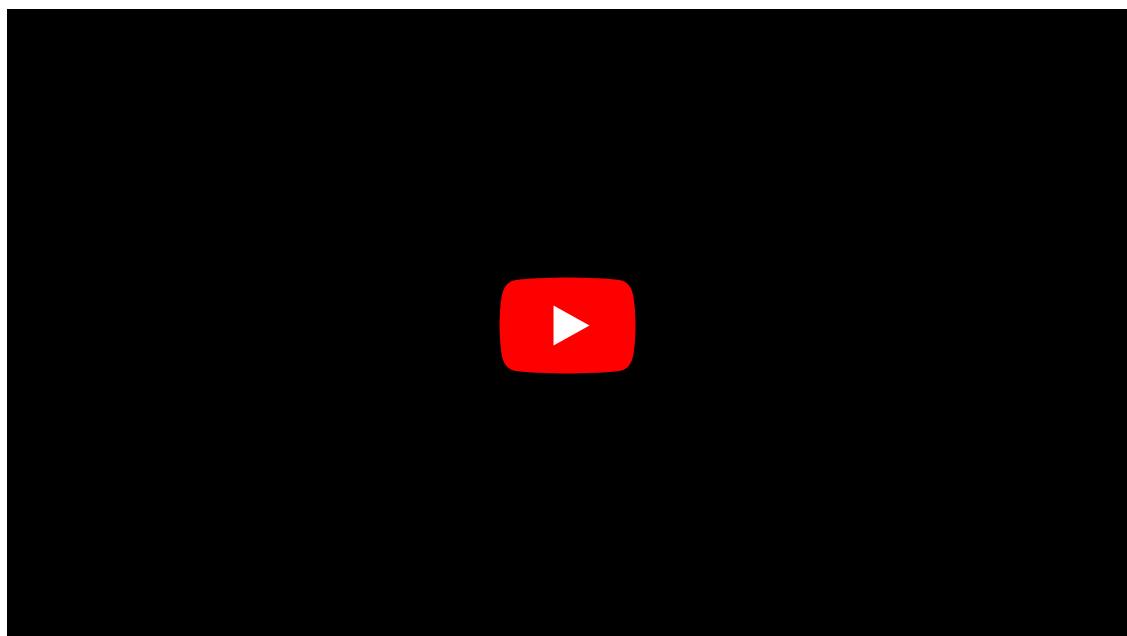
The fallback content to display on prerendering
built on a

The fallback content to display on prerendering

. It uses the most popular **industry standards** for process modeling, business rule management and integrates as easily with legacy systems as with the latest APIs and RPAs.

Also, all applications you create are **containerized, portable, scalable, and resilient** out of the box. You're free to deploy anywhere and scale to any size without redesign.

FLOWX.AI can be deployed in a private cloud, in a public cloud or on-prem, depending on your requirements.



Why does it matter?

FLOWX.AI can be **deployed on top of existing legacy systems, so there is no need for costly or risky upgrade projects**. This helps lower the stress on the IT team and the technology budget since studies show that around 65-75% of the IT budget goes towards maintaining current infrastructure. Now, It's not reasonable to

expect enterprises are just going to rip and replace the legacy stack with new applications. They will do so at some point but for now they need something that enables them to run existing business and gives them some leeway or headspace to create modern digital experiences.

FLOWX.AI platform brings **a layer of scalability to your existing stack**, beyond their current capabilities. This is thanks to our Kafka and Redis core that queue messages until the system is able to respond. And best of all, the app user is not experiencing any lag, since data is pre-pulled in the front-end ahead of his actions. A typical use case might sustain 100,000 users per minute, but of course, given our containerized architecture, it can be scaled even more.

Unified interface across multiple systems or platforms - often, say for an in-branch onboarding process, a teller has to use 4 or 5 different applications - to access various customer data such as a CRM, public reference checks, a KYC system and so on. With a process designed in FLOWX, you create just one application that unifies the purpose and the data from all those other applications. And this is very liberating for employees, it saves up time, eliminates the possibility of errors and overall, makes the experience of using the onboarding application a pleasant one.

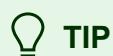
With FLOWX you **build omnichannel experiences across all digital channels**, be they web applications, mobile apps or in-branch terminals. What's more, our applications are built with a hand-off capability - meaning the user can start the process on the web and then pick up on the mobile app later that evening.

The UI is generated on the fly, by our AI model. This means that you don't need coding or design skills to create interfaces. Of course, you can inject your own code for CSS styling, apply your own design system with logo, corporate colors and fonts - but this is just if you want it. By default, you don't need it.

And of course, when it comes to processes, **we support a no-code/full-code framework that makes the platform available to any citizen developer**. This brings speed to development, since there is no disconnect between business and IT, supports agile ways of working and overall, has a positive impact over creativity and innovation.

Next steps

We'll guide you through everything you need to know in order to understand FLOWX.AI, deploy it and use it successfully inside your organization.



If you have any questions regarding the content here or anything else that might be missing and you'd like to know, please get in touch with us! We'd be happy to help!

So, to start with, let's dive into FLOWX.AI!

» Quick start

Read about the frameworks and standards used to build the platform:

» Frameworks and standards

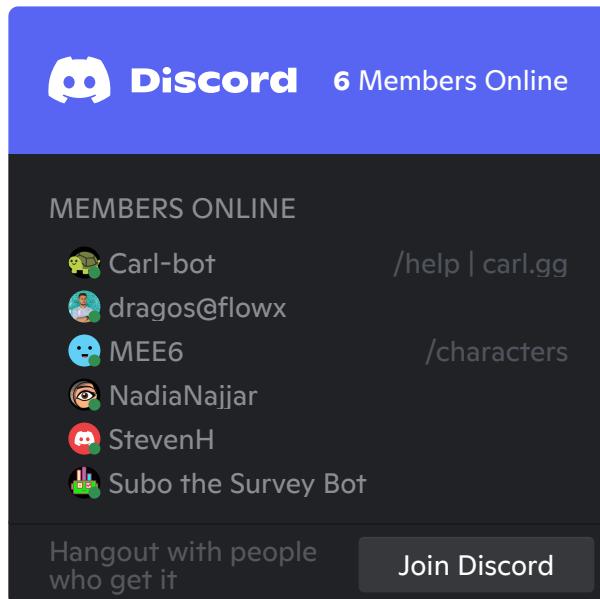
Find about the core platform components:

» FLOWX.AI Architecture

See the Release Notes:

» Realease notes

Build and launch mission critical software products with FLOWX.AI - Learn and share tips and tricks with our community on Discord:



Was this page helpful?

GETTING STARTED / Building your first process

Prerequisites

Let's dive into an example. 

- **Step 1:** Design a BPMN process
- **Step 2:** Define and manage a process flow using
 - The fallback content to display on prerendering
- **Step 3:** Run a process instance in
 - The fallback content to display on prerendering
- **Step 4:** Create the
 - The fallback content to display on prerendering
- **Step 5:** Connect
 - The fallback content to display on prerendering

Designing the BPMN process: request a new credit card from a bank app

Let's start with designing the BPMN process diagram for our sample use case: requesting a new credit card from a bank app.

Sample process steps

We'll take as a

The fallback content to display on prerendering
a credit card application. It will have the following steps:

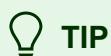
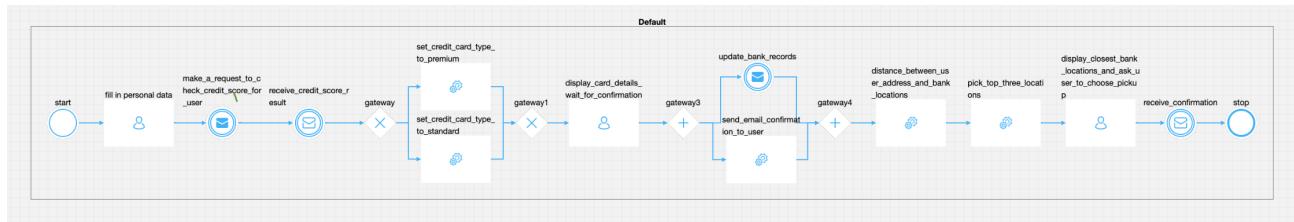
- a user makes a request for a new credit card - **start event**
- the user has to fill in a form with their personal data - **user task**
- the bank system must check the users credit score, this is done automatically using a send event that sends a credit score check request to the credit score adapter and a receive event that waits for the reply from the adapter - **automatic task**
- the process is split in two branches depending on the credit score - **exclusive gateway**
- on each of those branches are a service task that saves the appropriate credit card type to the proces data - **automatic task**
- the two branches are merged back into one by a **closing gateway**
- the user is shown the details of the credit card and they have to confirm it - **user task**
- after the user confirmation, the process is split again into two branches, this time they take place in parallel - **parallel gateway**. An action to register the request in the banks systems (bank system adapter / integration) and a confirmation email (notification plugin) to be sent to the user
- another automatic task follows, a call to an external API to compute the distance between the users address and the bank locations (<https://developers.google.com/maps/documentation/distance-matrix/overview>) - **automatic task**
- a new task is used to sort the location distances and pick the top three to be displayed to the user - **automatic task**

- the user has to pick the card pickup point from the bank location suggestions - ***user task***
- a receive task will wait to the confirmation from the bank that the user has picked up the new card and the process flow ends - ***end event***

Sample process diagram

This is what the

The fallback content to display on prerendering
diagram looks like:



Download sample here.

Was this page helpful?

GETTING STARTED / Learn more

🚀 Based on what you need to accomplish and understand, find below-

suggested tracks you can follow. Choose the track that suits you best.

Platform overview

Take a look on the frameworks and standards used, our architecture and the latest features that we are releasing.

- 📌 [Overview](#)
- 📌 [Frameworks and standards](#)
- 📌 [Architecture](#)
- 📌 [Release Notes](#)

Design a process

I want to design a process using FLOWX.AI.

- 📌 [Overview](#)
- 📌 [Building blocks](#)
- 📌 [Designer](#)
- 📌 [Academy - Your first FLOWX.AI processs](#)

Build an application

I want to build an application using FLOWX.AI.

- 📌 **Overview**
- 📌 **Building blocks**
- 📌 **Core components**
- 📌 **Integrations**
- 📌 **Plugins**

Additional support

Find additional support when you're stuck.

- 📌 **FAQs**
- <http://localhost:3000/docs/>📌
- Troubleshooting [TBD]**
- <http://localhost:3000/docs/>📌 **Best practices [TBD]**

Was this page helpful?

PLATFORM OVERVIEW / Frameworks and standards / Business process industry and standards / Intro to BPMN / BPMN basic concepts

Let's get into a bit more details on the main types of BPMN process elements.

Events

Events are **signals that something happens** – this includes the start and end of a process as well as any interaction with the process' environment.

There are 3 types of events:

- start events
- end events
- intermediate events

Start and End events

Start & End events

Start Event Icon	End Event Icon
------------------	----------------

Start Event Icon	End Event Icon
	
event that triggers the process	event that defines the state that terminates the process

Intermediate events

Message events

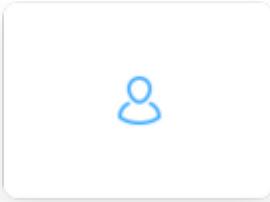
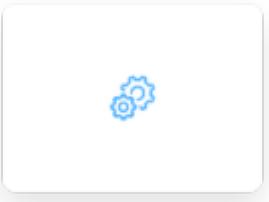
- represents incoming or outgoing messages from external parties - information, email, bank transfer
- Receive Message Event - incoming message occurring during the process flow, somewhere between start and end
- Send Message Event - outgoing message

Send Message Event Icon	Receive Message Event Icon
	
outgoing message	incoming message

Activities

Task

- it is an atomic activity within a process flow. You create a task when the activity cannot be broken down to a finer level of detail. A task can only belong to one lane.

User task	Service task
	
a task that requires the human to perform an action	a task that uses a Web service, an automated application, or other kinds of service in completing the task.

Send Task

- represents a task that sends a Message to another lane or pool. The Task is completed once the Message has been sent.

Receive Task

- indicates that the process has to wait for a message to arrive in order to continue. The Task is completed once the message has received.

User Task

- is a Task that is performed without the aid of any business process execution engine or any application. It is performed when the user performs a certain action in the application.

Service Task

- is executed by a business process engine. The task defines a script that the engine can interpret. When the task begins, the engine will execute the script. The Task will be completed when the script is completed. It also provides a mechanism for a process to run a script on the process data.

BPMN Subprocesses

In BPMN, a subprocess is a compound activity that represents a collection of other tasks and subprocesses. Generally, we create BPMN diagrams to communicate processes with others. To facilitate effective communications, we really do not want to make a business process diagram too complex. By using subprocesses, you can split a complex process into multiple levels, which allows you to focus on a particular area in a single process diagram.

Gateways

Gateways allow to control as well as merge and split the process flow.

The fallback content to display on prerendering

.

Exclusive gateways

In business processes, you typically need to make choices — **business decisions**. The most common type of decision is choosing **either/or**. Exclusive Gateways limit the possible outcome of a decision to a single path, and circumstances choose which one to follow.

Parallel gateways

In many cases, you want to split up the flow within your business process. For example the sales and risk departments may examine a new mortgage application at the same time. This reduces the total cycle time for a case. To express parallel flow in BPMN, you use a **parallel gateway**.

Exclusive gateway (XOR)	Parallel gateway (AND)
<ul style="list-style-type: none">defines a decision point	<ul style="list-style-type: none">no decision making;all outgoing branches are activated

Closing gateway

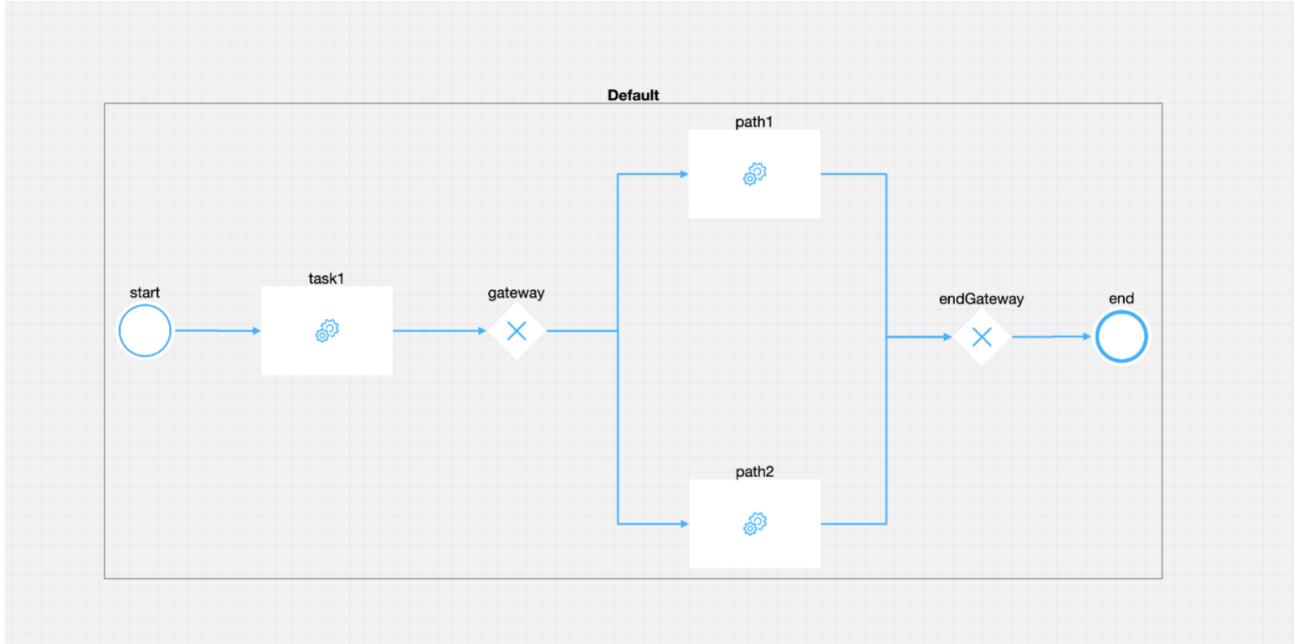
- closes gateways by connecting branches with no logic involved
- symbol used depends on the initial gateway
- parallel gateways - waits for all input tokens and merges all into one single token

- inclusive gateways
 - waits for all active inputs
 - is informed about all preceding token flows - knows the path selected and are expecting the token from these

[Was this page helpful?](#)

PLATFORM OVERVIEW / Frameworks and standards / Business process industry and standards / Intro to DMN

As we've seen in the previous chapter, Business Process Model and Notation (**BPMN**) is used to define business processes as a sequence of activities. If we need to branch off different process paths, we use gateways. These have rules attached to them in order to decide on which outgoing path should the process continue on.



!(INFO)

For more information on how to define DMN gateway decisions, check the **Exclusive gateway node** section.

We needed a convenient way of specifying the

The fallback content to display on prerendering
and we picked two possible ways of writing business rules:

- defining them as DMN decisions

!(INFO)

You can now define a DMN Business Rule Action directly in

The fallback content to display on prerendering
. For more information, check the **DMN Business Rule Action** section.

- adding **MVEL** scripts

What is Decision Model and Notation (DMN)?

Decision Model and Notation (or DMN) is a graphical language that is used to specify business decisions. DMN acts as a translator, converting the code behind complex decision-making into easily readable diagrams.

The Business Process Model and Notation is used to create the majority of process models (**BPMN**). The DMN standard was developed to complement BPMN by providing a mechanism for modeling decision-making represented by a Task within a process model. DMN does not have to be used in conjunction with BPMN, but it is highly compatible.

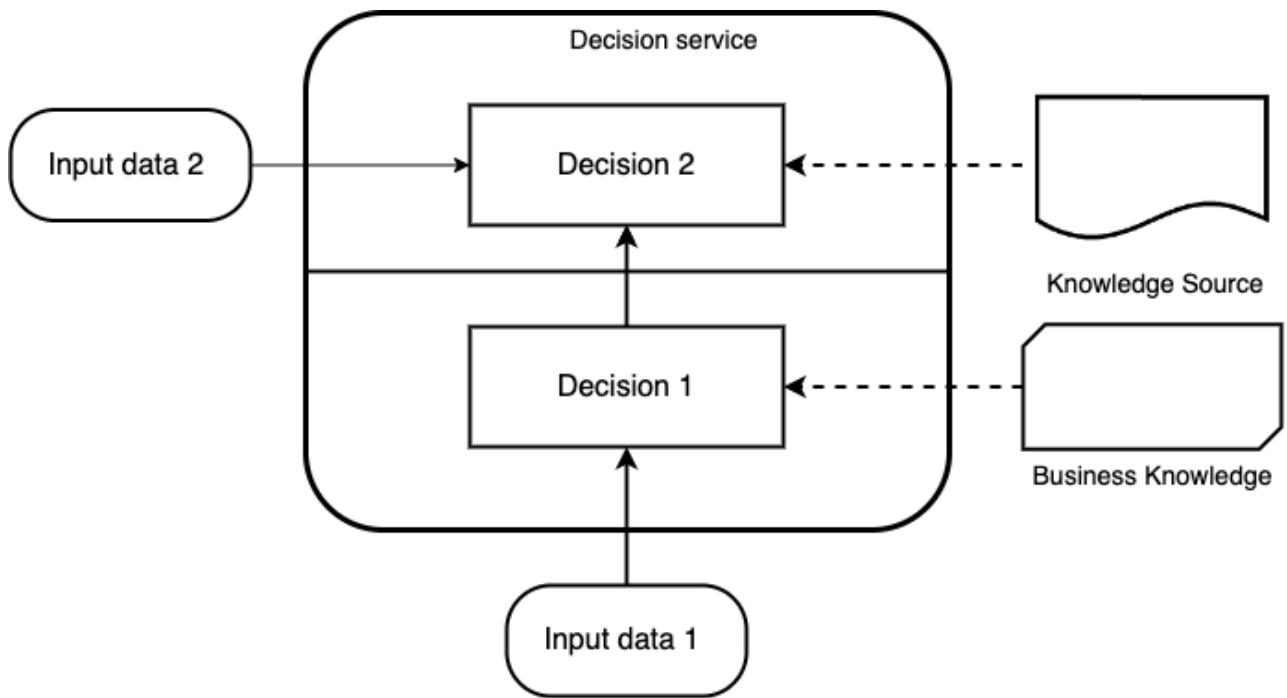


FLOWX.AI supports DMN 1.3 version.

DMN Elements

There are 4 basic elements of the **Decision Model and Notation**:

- **Decision**
- **Business Knowledge Model**
- **Input Data**
- **Knowledge Source**



Decision

It's the center point of a DMN diagram and it symbolizes the action that determines as output the result of a decision.

Decision service

A decision service is a high-level decision with well-defined inputs that is made available as a service for invocation. An external application or business process can call the decision service (BPMN).

Business Knowledge Model

It portrays a specific knowledge within the business. It stores the origin of the information. Decisions that have the same logic but depend on different sub-input data or sub-decisions use business knowledge models to determine which procedure to follow.

Example: a decision, rule, or standard table.

Input Data

This is the information used as an input to the normal decision. It's the variable that configures the result. Input data usually includes business-level concepts or objects relevant to the business.

Example: Entering a customer's tax number and the amount requested in a credit assessment decision.

Knowledge Source

It's a source of knowledge that conveys a kind of legitimacy to the business.

Example: policy, legislation, rules.

DMN Decision Table

A decision table represents decision logic which can be depicted as a table in Decision Model and Notation. It consists of inputs, outputs, rules, and hit policy.

Decision table elements	
Inputs	A decision table can have one or more input clauses, that represent the attributes on which the rule should be applied.

Decision table elements	
Outputs	Each entry with values for the input clause needs to be associated with output clauses. The output represents the result that we set if the rules applied to the input are met.
Rules	Each rule contains input and output entries. The input entries are the condition and the output entries are the conclusion of the rule. If each input entry (condition) is satisfied, then the rule is satisfied and the decision result contains the output entries (conclusion) of this rule.
Hit policy	The hit policy specifies what the result of the decision table is in cases of overlapping rules, for example, when more than one rule matches the input data.

Hit Policy examples

Unique **First** **Priority** **Any** **Rule order** **Collect order**

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

DMN Model

DMN defines an XML schema that allows DMN models to be used across multiple DMN authoring platforms.

You can use this XML example with FLOWX Designer, adding it to a Business Rule Action - using an MVEL script. Then you can switch to DMN if you need to generate a graphical representation of the model.

Using DMN with FLOWX Designer

As mentioned previously, DMN can be used with FLOWX Designer for the following scenarios:

- For defining gateway decisions, using **exclusive gateways**.
- For defining **business rules actions** attached to a **task node**.

In depth docs

» [DMN Documentation](#)

Was this page helpful?

PLATFORM OVERVIEW / Frameworks and standards / Business process industry and standards / Intro to MVEL

We can also **specify the business rules logic using MVEL scripts**. As opposed to DMN, with MVEL you can create complex business rules with multiple parameters and sub-calculations.

What is MVEL?

MVFLEX Expression Language (MVEL) is an expression language with a syntax similar to the Java programming language. This makes it relatively easy to use in order to define more complex business rules and that cannot be defined using DMN.

The runtime allows MVEL expressions to be executed either interpretively, or through a pre-compilation process with support for runtime byte-code generation to remove overhead. We pre-compile most of the MVEL code in order to make sure the process flow advances as fast as possible.

Example

```
if( input.get("user.credit_score") >= 700 ) {  
  
    output.setNextNodeName("TASK_SET_CREDIT_CARD_TYPE_PREMIUM");  
} else {  
  
    output.setNextNodeName("TASK_SET_CREDIT_CARD_TYPE_STANDARD");  
}
```

In depth docs

» [MVEL Documentation](#)

[Was this page helpful?](#)

PLATFORM OVERVIEW / Frameworks and standards / Event-driven architecture frameworks / Intro to Elasticsearch

Elasticsearch itself is not inherently event-driven, it can be integrated into event-driven architectures or workflows. External components or frameworks detect and trigger events, and Elasticsearch is utilized to efficiently index and make the event data searchable. This integration allows event-driven systems to leverage Elasticsearch's powerful search and analytics capabilities for real-time processing and retrieval of event data.

What is Elasticsearch?

Elasticsearch is a powerful and highly scalable open-source search and analytics engine built on top of the [Apache Lucene](#) library. It is designed to handle a wide range of data types and is particularly well-suited for real-time search and data analysis use cases. Elasticsearch provides a distributed, document-oriented architecture, making it capable of handling large volumes of structured, semi-structured, and unstructured data.

How it works?

At its core, Elasticsearch operates as a distributed search engine, allowing you to store, search, and retrieve large amounts of data in near real-time. It uses a schema-less JSON-based document model, where data is organized into indices, which can be thought of as databases. Within an index, documents are stored, indexed, and made searchable based on their fields. Elasticsearch also provides powerful querying capabilities, allowing you to perform complex searches, filter data, and aggregate results.

Why it is useful?

One of the key features of Elasticsearch is its distributed nature. It supports automatic data sharding, replication, and node clustering, which enables it to handle massive amounts of data across multiple servers or nodes. This distributed architecture provides high availability and fault tolerance, ensuring that data remains accessible even in the event of hardware failures or network issues.

Elasticsearch integrates with various programming languages and frameworks through its comprehensive RESTful API. It also provides official clients for popular languages like Java, Python, and JavaScript, making it easy to interact with the search engine in your preferred development environment.

Indexing & sharding

Indexing

Indexing refers to the process of adding, updating, or deleting documents in Elasticsearch. It involves taking data, typically in JSON format, and transforming it into indexed documents within an index. Each document represents a data record and contains fields with corresponding values. Elasticsearch uses an inverted index data structure to efficiently map terms or keywords to the documents containing those terms. This enables fast full-text search capabilities and retrieval of relevant documents.

Sharding

Sharding, on the other hand, is the practice of dividing index data into multiple smaller subsets called shards. Each shard is an independent, self-contained index that holds a portion of the data. By distributing data across multiple shards, Elasticsearch achieves horizontal scalability and improved performance. Sharding allows Elasticsearch to handle large amounts of data by parallelizing search and indexing operations across multiple nodes or servers.

Shards can be configured as primary or replica shards. Primary shards contain the original data, while replica shards are exact copies of the primary shards, providing redundancy and high availability. By having multiple replicas, Elasticsearch ensures data durability and fault tolerance. Replicas also enable parallel search operations, increasing search throughput.

Sharding offers several advantages. It allows data to be distributed across multiple nodes, enabling parallel processing and faster search operations. It also provides fault tolerance, as data is replicated across multiple shards. Additionally, sharding allows Elasticsearch to scale horizontally by adding more nodes and distributing the data across them.

The number of shards and their allocation can be determined during index creation or modified later. It is important to consider factors such as the size of the dataset, hardware resources, and search performance requirements when deciding on the number of shards.

For more details, check Elasticsearch documentation:

» [Elasticsearch](#)

Leveraging Elasticsearch for advanced indexing with FLOWX.AI

The integration between FLOWX.AI and Elasticsearch involves the indexing of specific keys or data from the

The fallback content to display on prerendering

to

The fallback content to display on prerendering using Elasticsearch. This indexing process is initiated by the

The fallback content to display on prerendering in a synchronous manner, sending the data to Elasticsearch. The data is then indexed or updated in the "process_instance" index.

To ensure effective indexing of process instances' details, a crucial step involves defining a mapping that specifies how Elasticsearch should index the received messages. This mapping is essential as the process instances' details often have specific formats. The process-engine takes care of this by automatically creating an index template during startup if it doesn't already exist. The index template acts

as a blueprint, providing Elasticsearch with the necessary instructions on how to index and organize the incoming data accurately. By establishing and maintaining an appropriate index template, the integration between FLOWX.AI and Elasticsearch can seamlessly index and retrieve process instance information in a structured manner.

Kafka transport strategy

The fallback content to display on prerendering transport strategy implies process-engine sending messages to a Kafka topic whenever there is data from a process instance to be indexed. Kafka Connect is then configured to read these messages from the topic and forward them to Elasticsearch for indexing.

This approach offers benefits such as fire-and-forget communication, where the process-engine no longer needs to spend time handling indexing requests. By decoupling the process-engine from the indexing process and leveraging Kafka as a messaging system, the overall system becomes more efficient and scalable. The process-engine can focus on its core responsibilities, while Kafka Connect takes care of transferring the messages to Elasticsearch for indexing.

To optimize indexing response time, Elasticsearch utilizes multiple indices created dynamically by the Kafka Connect connector. The creation of indices is based on the timestamp of the messages received in the Kafka topic. The frequency of index creation, such as per minute, hour, week, or month, is determined by the timestamp format configuration of the Kafka connector.

It's important to note that the timestamp used for indexing is the process instance's start date. This means that subsequent updates received for the same object will be directed to the original index for that process instance. To ensure proper

identification and indexing, it is crucial that the timestamp of the message in the Kafka topic corresponds to the process instance's start date, while the key of the message aligns with the process instance's UUID. These two elements serve as unique identifiers for determining the index in which a process instance object was originally indexed.

For more details on how to configure process instance indexing through Kafka transport, check the following section:

» [Configuring elasticsearch indexing](#)

» [Configuration guidelines](#)

Was this page helpful?

PLATFORM OVERVIEW / Frameworks and standards / Event-driven architecture frameworks / Intro to Kafka concepts

What is 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 end-

point 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 can 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.

Benefits of using Kafka

- **Reliability** – Kafka is distributed, partitioned, replicated, and fault tolerant
- **Scalability** – Kafka messaging system scales easily without downtime
- **Durability** – Kafka uses Distributed commit log which means messages persist on disk as fast as possible
- **Performance** – Kafka has high throughput for both publishing and subscribing messages. It maintains a stable performance even though many TB of messages are stored.

Key Kafka concepts

Events

Kafka encourages you to see the world as sequences of events, which it models as key-value pairs. The key and the value have some kind of structure, usually represented in your language's type system, but fundamentally they can be

anything. Events are immutable, as it is (sometimes tragically) impossible to change the past.

Topics

Because the world is filled with so many events, Kafka gives us a means to organize them and keep them in order: topics. A topic is an ordered log of events. When an external system writes an event to Kafka, it is appended to the end of a topic.

In FLOWX.AI, Kafka handles all communication between the **FLOWX Engine** and external plugins and integrations. It is also used for notifying running process instances when certain events occur. More information about KAFKA configuration on the section below:

» [FLOWX engine setup guide](#)

Producer

A producer is an external application that writes messages to a Kafka cluster, communicating with the cluster using Kafka's network protocol.

Consumer

The consumer is an external application that reads messages from Kafka topics and does some work with them, like filtering, aggregating, or enriching them with other information sources.

» How to create a Kafka producer

» How to create a Kafka consumer

In-depth docs

» Kafka documentation

» How Kafka works

Was this page helpful?

PLATFORM OVERVIEW / Frameworks and standards / Event-driven architecture frameworks / Intro to Kubernetes

What is Kubernetes?

Kubernetes is an open-source container orchestration platform that automates many of the manual processes involved in containerized application deployment,

management, and scaling.

The purpose of Kubernetes is to orchestrate containerized applications to run on a cluster of hosts. **Containerization** enables you to deploy multiple applications using the same operating system on a single virtual machine or server.

Kubernetes, as an open platform, enables you to build applications using your preferred programming language, operating system, libraries, or messaging bus. To schedule and deploy releases, existing continuous integration and continuous delivery (CI/CD) tools can be integrated with Kubernetes.

Benefits of using Kubernetes

- A proper way of managing containers
- High availability
- Scalability
- Disaster recovery

Key Kubernetes Concepts

Node & PODs

A Kubernetes node is a machine that runs containerized workloads as part of a Kubernetes cluster. A node can be a physical machine or a virtual machine, and can be hosted on-premises or in the cloud.

A pod is composed of one or more containers that are colocated on the same host and share a network stack as well as other resources such as volumes. Pods are the foundation upon which Kubernetes applications are built.

Kubernetes uses pods to run an instance of your application. A pod represents a single instance of your application.

Pods are typically ephemeral, disposable resources. Individually scheduled pods miss some of the high availability and redundancy Kubernetes features. Instead, pods are deployed and managed by Kubernetes *Controllers*, such as the Deployment Controller.

Service & Ingress

Service is an abstraction that defines a logical set of pods and a policy for accessing them. In Kubernetes, a Service is a REST object, similar to a pod. A Service definition, like all REST objects, can be POSTed to the API server to create a new instance. A Service object's name must be a valid [RFC 1035](#) label name.

Ingress is a Kubernetes object that allows access to the Kubernetes services from outside of the Kubernetes cluster. You configure access by writing a set of rules that specify which inbound connections are allowed to reach which services. This allows combining all routing rules into a single resource.

Ingress controllers are pods, just like any other application, so they're part of the cluster and can see and communicate with other pods. An Ingress can be configured to provide Services with externally accessible URLs, load balance traffic, terminate SSL / TLS, and provide name-based virtual hosting. An Ingress controller is in charge of fulfilling the Ingress, typically with a load balancer, but it may also configure your edge router or additional frontends to assist with the traffic.

FlowX.AI offers a predefined NGINX setup as Ingress Controller. The **NGINX Ingress Controller** works with the **NGINX** web server (as a proxy). For more information, check the below sections:

» [Intro to NGINX](#)

» [Designer setup guide](#)

ConfigMap & Secret

ConfigMap is an API object that makes it possible to store configuration for use by other objects. A ConfigMap, unlike most Kubernetes objects with a spec, has `data` and `binaryData` fields. As values, these fields accept key-value pairs. The `data` field and `binaryData` are both optional. The `data` field is intended to hold UTF-8 strings, whereas the `binaryData` field is intended to hold binary data as base64-encoded strings.

!(INFO)

The name of a ConfigMap must be a valid DNS subdomain name.

Secret represents an amount of sensitive data, such as a password, token, or key. Alternatively, such information could be included in a pod specification or a container image. Secrets are similar to ConfigMaps but they are designed to keep confidential data.

Volumes

A Kubernetes volume is a directory in the orchestration and scheduling platform that contains data accessible to containers in a specific pod. Volumes serve as a plug-in mechanism for connecting ephemeral containers to persistent data stores located elsewhere.

Deployment

A deployment is a collection of identical pods that are managed by the Kubernetes Deployment Controller. A deployment specifies the number of pod replicas that will be created. If pods or nodes encounter problems, the Kubernetes Scheduler ensures that additional pods are scheduled on healthy nodes.

Typically, deployments are created and managed using `kubectl create` or `kubectl apply`. Make a deployment by defining a manifest file in YAML format.

Kubernetes Architecture

Kubernetes architecture consists of the following main parts:

- Control Plane (master)
 - kube-apiserver
 - etcd
 - kube-scheduler
 - kube-controller-manager
 - cloud-controller-manager
- Node components
 - kubelet
 - kube-proxy
 - Container runtime

Install tools

kubectl

`kubectl` makes it possible to run commands against Kubernetes clusters using the `kubectl` command-line tool. `kubectl` can be used to deploy applications, inspect and manage cluster resources, and inspect logs. See the `kubectl` [reference documentation](#) for more information.

kind

`kind` command makes it possible to run Kubernetes on a local machine. As a prerequisite, Docker needs to be installed and configured. What `kind` is doing is to run local Kubernetes clusters using Docker container “nodes”.

In depth docs

» [Kubernetes documentation](#)

Was this page helpful?

PLATFORM OVERVIEW / Frameworks and standards / Event-driven architecture frameworks / Intro to NGINX

What is NGINX?

NGINX is a free, open-source, high-performance web server with a rich feature set, simple configuration, and low resource consumption that can also function as a reverse proxy, load balancer, mail proxy, HTTP cache, and many other things.

How NGINX is working?

NGINX allows you to hide a server application's complexity from a front-end application. It uses an event-driven, asynchronous approach to create a new process for each web request, with requests handled in a single thread.

Using NGINX with FLOWX Designer

The **NGINX Ingress Controller for Kubernetes** - `ingress-nginx` is an ingress controller for Kubernetes using NGINX as a reverse proxy and load balancer.

Ingress allows you to route requests to services based on the host or path of the request, centralizing a number of services into a single entry point.

The **ingress resource** simplifies the configuration of **SSL/TLS termination, HTTP load-balancing, and layer routing**.

For more information, check the following section:

» [Using NGINX as a K8S ingress controller](#)

Integrating with FLOWX Designer

FLOWX Designer is using NGINX ingress controller for the following actions:

1. For routing calls to plugins
2. For routing calls to the **FLOWX Engine**:
 - Viewing current instances of processes running in the FLOWX engine
 - Testing process definitions from the FLOWX Designer - route the API calls and SSE communications to the FLOWX engine backend
 - Accessing REST API of the backend microservice
3. For configuring the Single Page Application (SPA) - FLOWX Designer SPA will use the backend service to manage the platform via REST calls

In the following section, you can find a suggested NGINX setup, the one used by FLOWX.AI:

» [Designer setup guide](#)

Installing NGINX Open Source

For more information on how to install NGINX Open Source, check the following guide:

» [NGINX Install Guide](#)

Was this page helpful?

PLATFORM OVERVIEW / Frameworks and standards / Event-driven architecture frameworks / Intro to Redis

What is 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.

Redis offers a primary-replica architecture in a single node primary or a clustered topology. This allows you to build highly available solutions providing consistent performance and reliability. Scaling the cluster size up or down is done very easily, this allows the cluster to adjust to any demands.

In depth docs

» [Redis.io](#)

» Redis overview

Was this page helpful?

PLATFORM OVERVIEW / Frameworks and standards / Timer expressions

When working with FLOWX.AI components, there are multiple scenarios in which timer expressions are needed.

There are two timer expressions formats supported:

- **Cron Expressions** - used to define the expiry date on processes
- **ISO 8601** - used to define the duration of a response timeout or for a timer expression

Cron Expressions

A cron expression is a string made up of **six mandatory subexpressions (fields)** that each specifies an aspect of the schedule (for example, `* * * * *`).

These fields, separated by white space, can contain any of the allowed values with various combinations of the allowed characters for that field.

ⓘ INFO

A field may be an asterisk (`*`), which always stands for “first-last”. For the day-of-the-month or day-of-the-week fields, a question mark (`?`) may be

used instead of an asterisk.

Subexpressions:

1. Seconds
2. Minutes
3. Hours
4. Day-of-Month
5. Month
6. Day-of-Week
7. Year (optional field)

An example of a complete cron-expression is the string `0 0 12 ? * FRI` - which means **every Friday at 12:00:00 PM**.

More details:

» [Scheduling cron expressions](#)

Cron Expressions are used in the following example:

- **Process definition - Expiry time** - a user can set up a `expiryTime` function on a process, for example, a delay of 30s will be set up like:

siivu_main_process (DRAFT)

Process settings

General Sensitive data Swimlanes Task management

General data

1 { }

Expiry time

30 16 11 4 7 1

For more details about Expiry time formatting and examples of valid Cron Expressions, click [here](#).

Cancel Save settings

The screenshot shows the 'Process settings' dialog for a specific process. The 'General' tab is active. In the 'General data' section, there is one entry with the value '{}'. Below this is an 'Expiry time' field containing '30 16 11 4 7 1'. A note below the field provides information about Cron expressions. At the bottom right are 'Cancel' and 'Save settings' buttons.

ISO 8601

ISO 8601 is an international standard covering the worldwide exchange and communication of date and time-related data. It can be used to standardize the following: dates, time of delay, time intervals, recurring time intervals, etc.

More details:

» ISO 8601

ISO 8601 format is used in the following examples:

- **Node config - Response Timeout** - can be triggered if, for example, a topic that you define and add in the **Data stream topics** tab does not respect the pattern

ISO 8601 dates and times:

Format accepted	Value ranges
Year (Y)	YYYY, four-digit, abbreviated to two-digit
Month (M)	MM, 01 to 12
Week (W)	WW, 01 to 53
Day (D)	D, day of the week, 1 to 7
Hour (h)	hh, 00 to 23, 24:00:00 as the end time
Minute (m)	mm, 00 to 59
Second (s)	ss, 00 to 59
Decimal fraction (f)	Fractions of seconds, any degree of accuracy

Node: **test test** (ID: 552202)

Node Config

Actions

Response Timeout

Response Timeout (PT30S)

Data stream topics

Topic Name

Key Name



Add stream

- **Actions - Timer expression** - it can be used if a delay is required on that action

Node: **test test** (ID: 552202)

[^](#) [X](#)

Node Config

Actions

Actions +

b333f699-ad08... +

13c58d4d-1cd... +

ab3de51d-5bf... +

Action Edit

ID: 555551

Name

b333f699-ad08-4f0c-954e-f37aaafcca512

Order

1

Timer Expression

Save

Save Data



Was this page helpful?

PLATFORM OVERVIEW / FLOWX.AI architecture

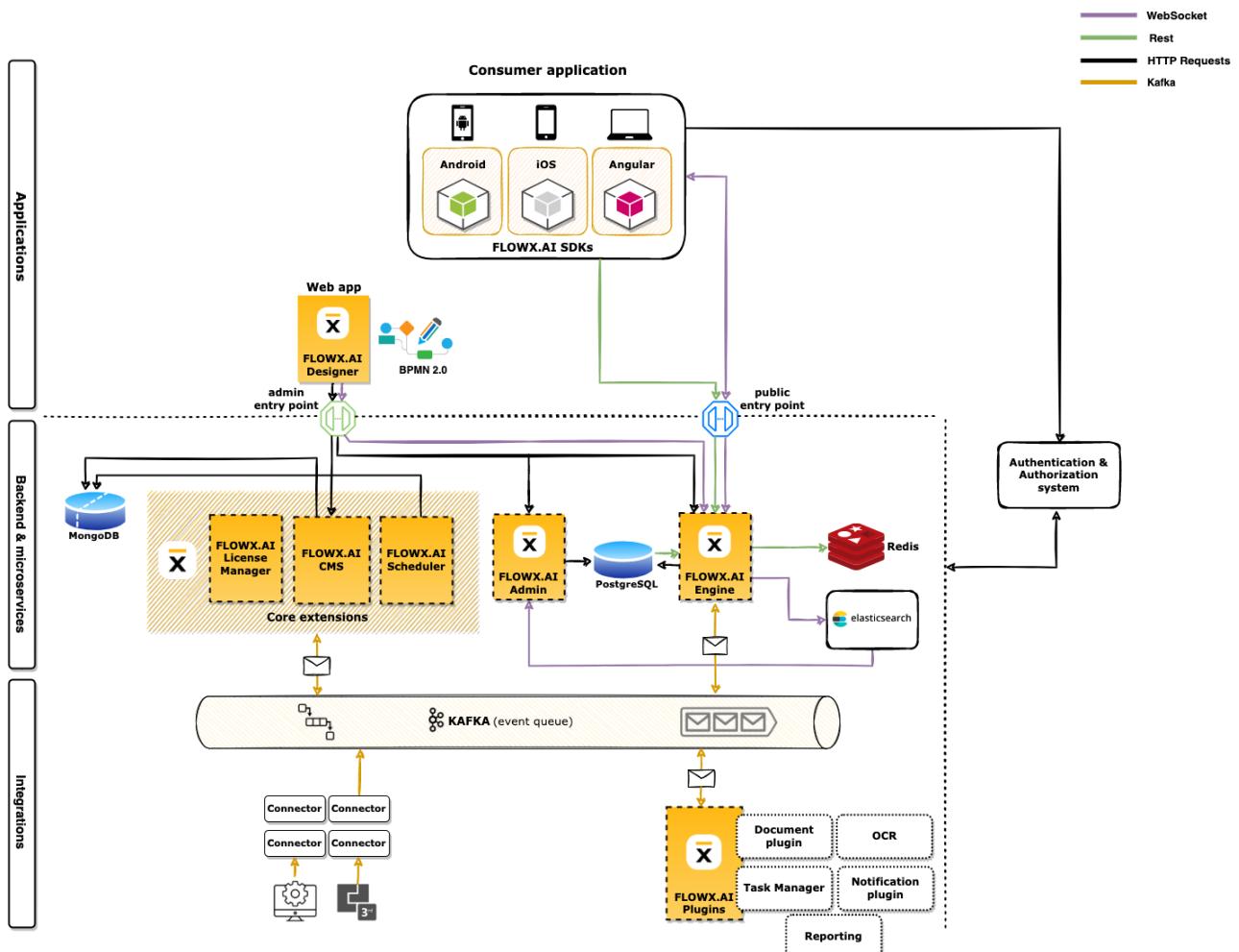
Let's go through the main components of the FLOWX.AI platform:

- **FLOWX.AI SDKs** - used in the [Web \(Angular\)](#), [iOS](#), and [Android](#) applications to render the process screens and orchestrate the [custom components](#)
- **FLOWX.AI Designer** - is a collaborative, no-code, web-based application development environment that enables users to create web and mobile applications without having to know how to code:
 - Develop processes based on [BPMN 2.0](#)
 - Configure user interfaces for the processes for both generated and custom screens
 - Define business rules and validations via [DMN](#) files or via the [MVEL](#), or other supported [scripting languages](#)
 - Create [integration connectors](#) in a visual manner
 - Create data models for your applications
 - Adding new capabilities by using [plugins](#)
 - Manage [users access](#)

Microservices

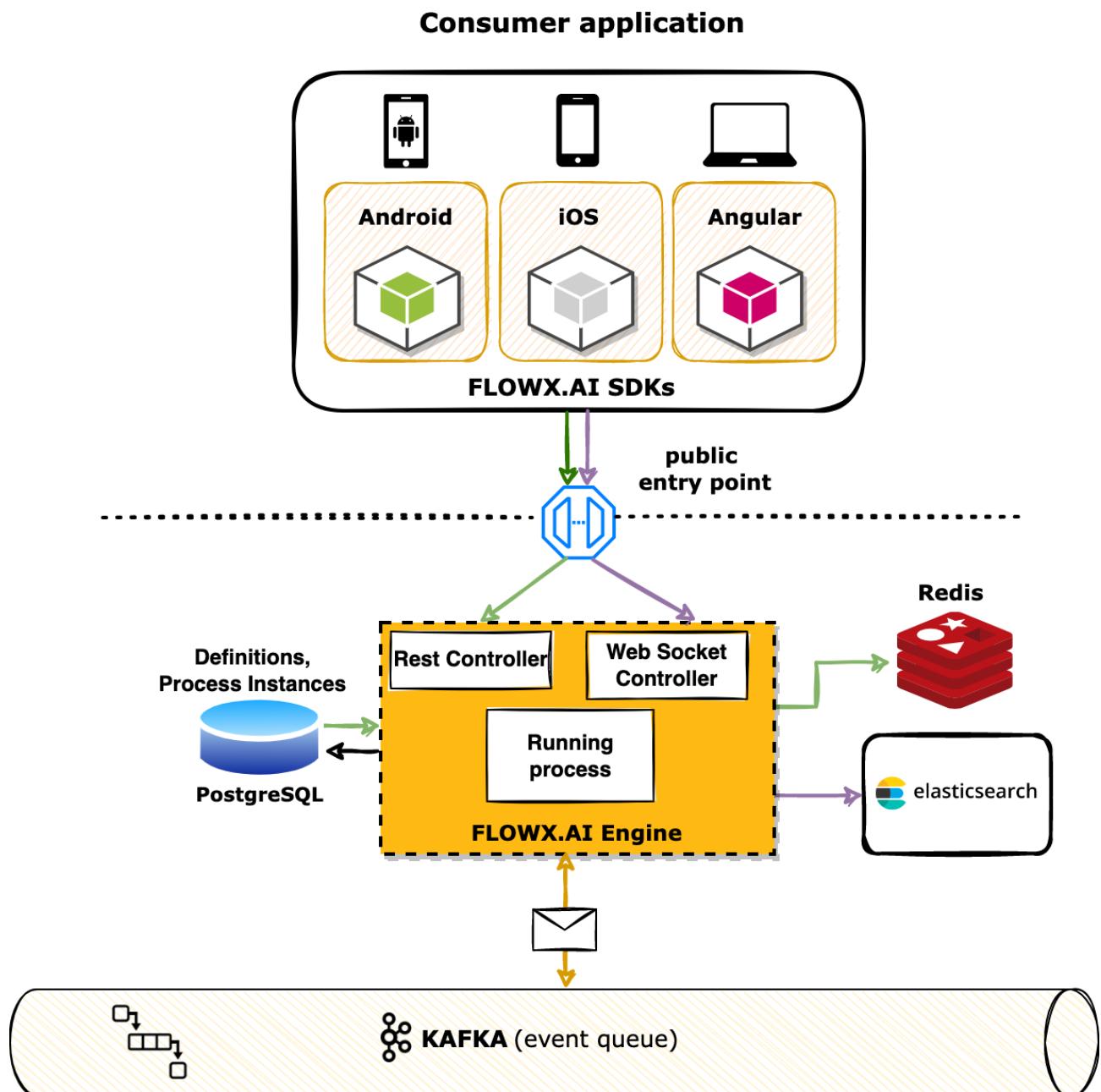
- **FLOWX.AI Engine** - is the core of the platform. It runs the business processes, coordinating integrations and the UI
- **FLOWX.AI Admin** - used to store/edit process definitions (FLOWX.AI Admin Microservice connects to the same Postgres / Oracle database as the FLOWX.AI Engine)

- **FLOWX.AI Scheduler** (part of the core components) - used to store/edit process definitions
- **FLOWX.AI Content Management** (part of the core components) - can be quickly deployed on the chosen infrastructure, preloaded with the needed taxonomies or contents, and then connected to the FLOWX Engine through Kafka events
- **FLOWX.AI License Manager** (part of the core components) - is used for displaying reports regarding the usage of the platform in the FLOWX.AI Designer
- **FLOWX.AI Plugins** - the platform comes with some ready-made integrations, such as a [document management] solution, a plugin for sending various types of **notifications**, an **OCR** plugin, and a task management plugin



FLOWX.AI Engine

We call it the engine because it's a nice analogy, once deployed on an existing stack, FLOWX.AI becomes the core of your digital operating model.



You can use FLOWX Engine to do the following:

- create any type of external or internal facing application
- redesign business processes from analog, paper-based ones to fully digital and automated processes

- manage integrations, so you can hook it up to existing CRMs, ERPs, KYC, transaction data and many more
- to read process definitions (if it is connected to the same DB as FLOWX.AI Admin)

FLOWX Engine runs the business processes, coordinating integrations and the omnichannel UI. It is a Kafka-based event-driven platform, that is able to orchestrate, generate and integrate with any type of legacy system, without expensive or risky upgrades.

This is extremely important because often, digital apps used by a bank's clients, for example, are limited by the load imposed by the core banking system. And the customers see blocked screens and endlessly spinning flywheels. FLOWX.AI buffers this load, offering a 0.2s response time, thus the customer never has to wait for data to load.

» [FLOWX Engine](#)

FLOWX.AI Designer

FLOWX.AI Designer is built to administrate everything in FLOWX.AI. It is a web application that runs in the browser, meaning that it resides out of a FLOWX deployment.

The platform has **no-code/full-code capabilities**, meaning applications can be developed in a visual way, available for anyone with a powerful business idea. So we're talking about business analysts, product managers - people without advanced programming skills, and also experienced developers.

The process visual designer works on **BPMN 2.0 standard** - meaning that the learning curve for business analysts or product managers is quite fast. Thus, creating new applications (e.g. onboarding an SME client for banks) or adding new functionality (allow personal data changes in an app) takes only 10 days, instead of 6 to 8 months.

However, we do support custom CSS or custom screens. Because we're aware each brand is different and each has its own CI, so you need to have the ability to create UIs that respect your brand guidelines.

» [FLOWX.AI Designer](#)

FLOWX.AI SDKs

Also, we provide web and native mobile SDKs, so that every app you create is automatically an omnichannel one: it can be displayed in a browser, embedded in an internet banking interface, or in a mobile banking app. Or even deployed as a standalone app in Google Play or AppStore.

Unlike other no-code/full-code platforms which provide templates or building blocks for the UI, ours is generated on the fly, as a business analyst creates the process and the data points. This feature reduces the need to use UX/UI expertise, the UI being generated respecting state-of-the-art UI frameworks.

» [Renderer SDKs](#)

FLOWX.AI Content management

This is another Java microservice that enables you to store and manage content.

The go-to place for all taxonomies. The extension offers a convenient way of managing various content pieces such as lists or content translations. Anything that is under content management is managed by the [CMS backend service](#). To store content, the service will use a MongoDB database (unstructured database). For example, each time you edit an [enumeration](#), the FLOWX.AI Designer will send an HTTP request to the microservice.

» [Content Management](#)

FLOWX.AI Scheduler

If you need to **set a timer on** a process that needs to end after X days, you can use the FLOWX.AI Scheduler microservice. It is a service that is able to receive requests (like a reminder application) to remind you in X amount of time to do something.

!(INFO)

When you start a process, the process must have an expiry date.

Scheduler microservice communicates with the FLOWX.AI Engine through Kafka Event Queue → it creates a new message (write some data) then will send that message to Kafka (with the scheduler address) → when the reminder time comes up, the scheduler will put back a new message in the Kafka layer with engine's destination (time + ID of the process).

» [Scheduler](#)

Authorization & session manager

We recommend Keycloak, a component that allows you to create users and store credentials. It can be also used for authorization - defining groups, and assigning roles to users.

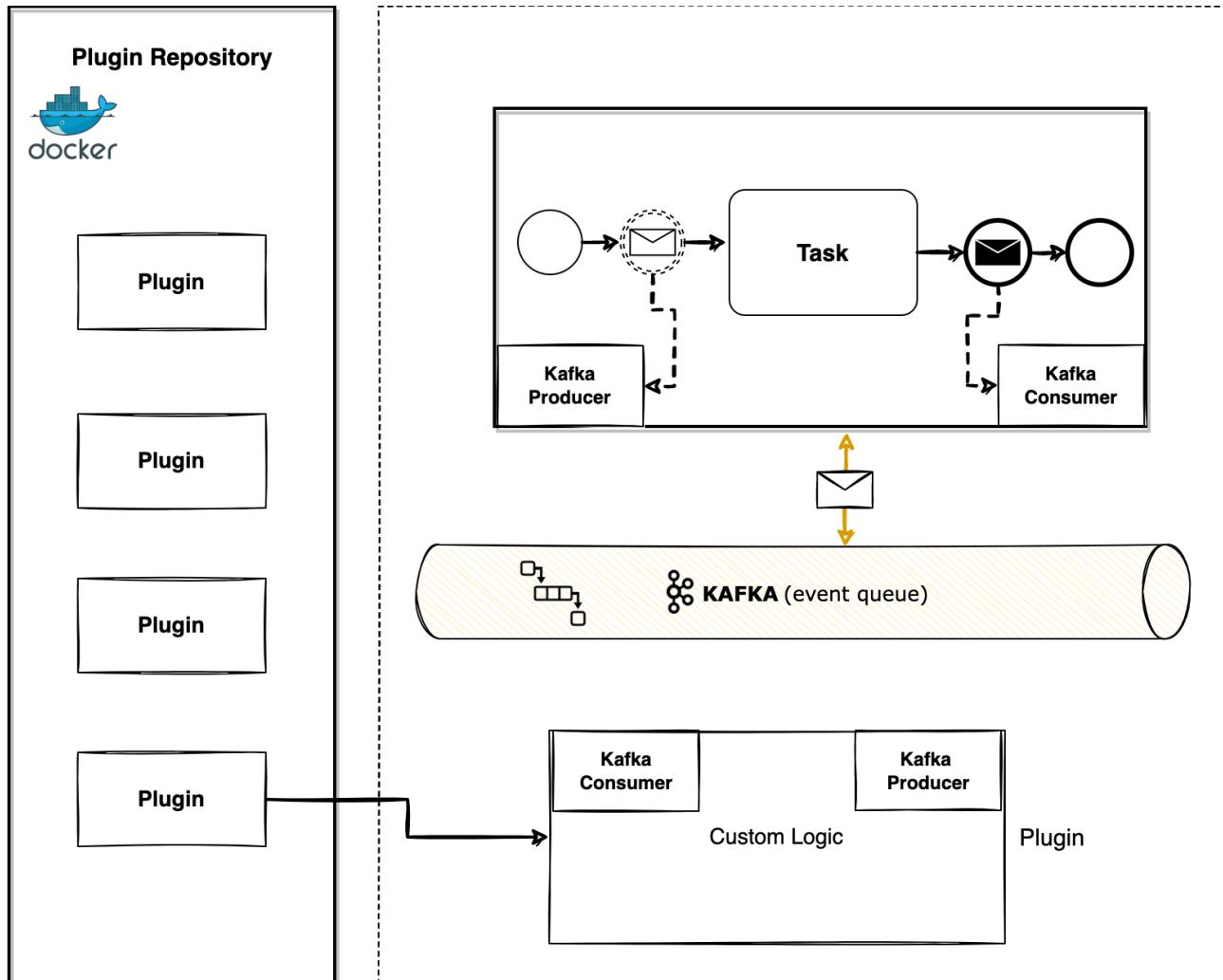
Every communication that comes from a consumer application, goes through a public entry point (API Gateway). To communicate with this component, the consumer application tries to start a process and the public entry point will check for authentication (Keycloak will send you a token) and the entry point validates it.

» [Keycloak Documentation](#)

Plugins

Plugins are bits of functionality that allow you to expand the functionality of the platform - for example, we have the following custom plugins:

- [FLOWX Notifications Plugin](#)
- [FLOWX Documents Plugin](#)
- [FLOWX OCR Plugin](#)
- [FLOWX Task Management Plugin](#)



» Plugins

Integrations

Connecting your legacy systems or third-party apps to the FLOWX.AI Engine is easily done through [custom integrations](#). These can be developed using your preferred tech stack, the only requirement is that they connect to Kafka. These could include legacy APIs, custom file exchange solutions, or RPA.

» Integrations

Was this page helpful?

BUILDING BLOCKS / Process Designer / Process definition

The core of the platform is the process definition, which is the blueprint of the business process made up of **nodes** that are linked by sequences.

Process Definitions

⋮

Drafts / In progress

Name	Version	Edited at	Edited by	⋮
Amazing Process	1	05 Oct 2022, 4:21 PM	John Doe	▶ ⚒ ⋮
Awesome Process	1	05 Oct 2022, 4:07 PM	John Doe	▶ ⚒ ⋮
Exquisite Process	2	05 Oct 2022, 2:49 PM	Jane Doe	▶ ⚒ ⋮

Published

Name	Version	Published at	Published by	⋮
Incredible Process	1	05 Oct 2022, 2:49 PM	John Doe	▶ ⚒ ⋮
Breathtaking Process	1	05 Oct 2022, 8:11 AM	Bess Twishes	▶ ⚒ ⋮
Stunning Process	1	04 Oct 2022, 12:30 PM	Silviu Grigore	▶ ⚒ ⋮

Once a process is defined and set as published on the platform, it can be executed, monitored, and optimized. When a business process is started, a new instance of the definition is created.

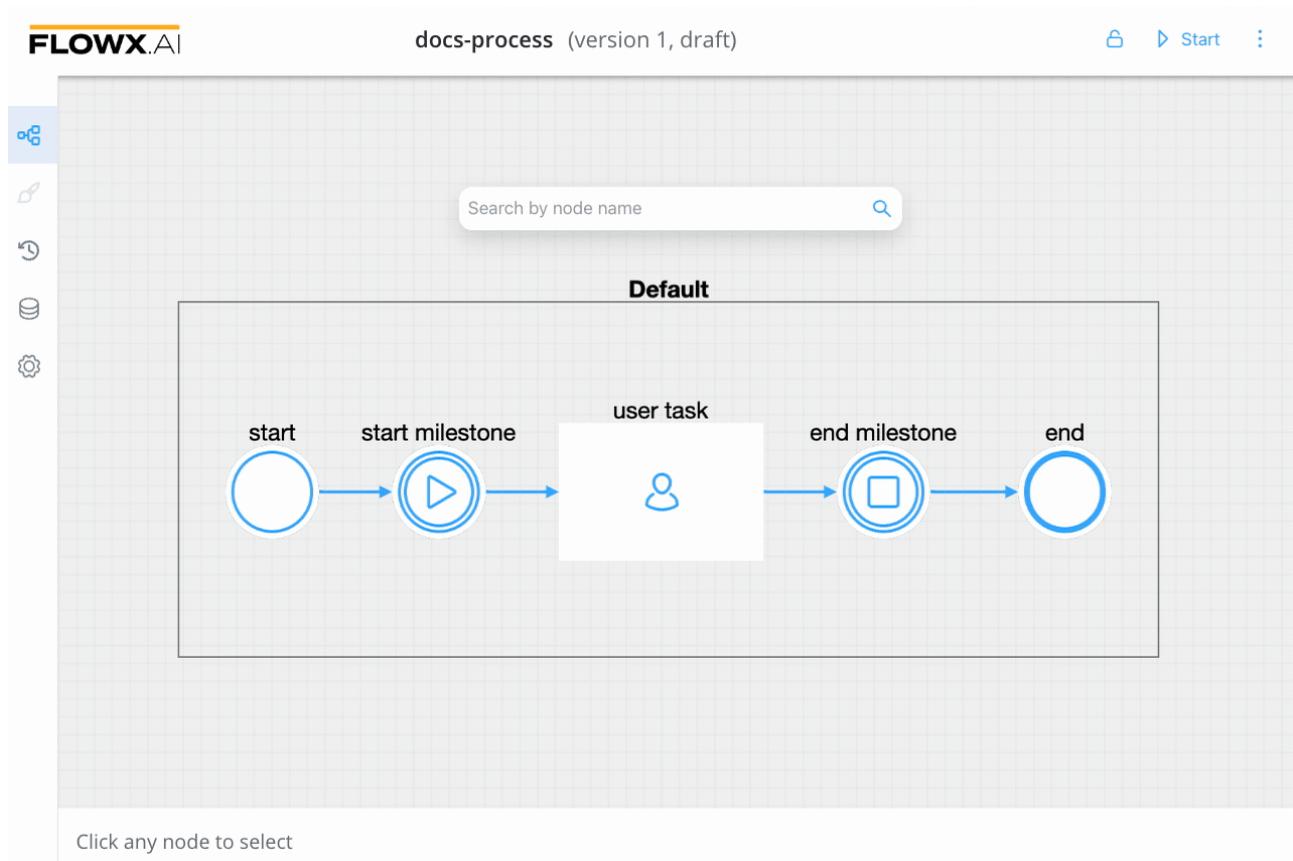
» [Process instance](#)

» [Failed process start](#)

History

In the **History** tab, you will find a record of all the modifications and events that have occurred in the process.

- **Versions** - provides information on who edited the process, when it was modified, and the version number and status
- **Audit log** - provides a detailed record of events and changes



Versions

In the **Versions** tab you will find the following details:

- Last edited on - the last time when the process was modified
- Last edited by - the last person who modified a process

- Version - version number
- Status - can be either **Published** or **Draft**

! HINT

! Published processes cannot be modified (they must be deprecated to be set as **Draft** before editing them).

- View process - clicking on the eye icon will redirect you to the process definition

Audit log

In the **Audit log** tab you will find the following items:

- Timestamp
- User
- Subject
- Event
- Subject Identifier
- Version
- Status

! INFO

Some items in the Audit log are filterable, making it easy to track changes in the process.

» Audit

Data model

In the Data Model, you can add new key-pair values, which enables you to use shortcuts when adding new keys using the UI Designer, without having to switch back and forth between menus.

X generate_data_model | Version 1 | draft

+ New item ⏷ ⏸ Start ⋮

Data Model				
	Name	Type	Used in reporting	Description
⌚	application	OBJECT	-	+ 🖍️ 🗑️
⌚	client	OBJECT	-	+ 🖍️ 🗑️
⌚	otherDeclarations	OBJECT	-	+ 🖍️ 🗑️
⌚	identificationData	OBJECT	-	+ 🖍️ 🗑️
⌚	legalAddress	OBJECT	-	+ 🖍️ 🗑️
⌚	residenceAddress	OBJECT	-	+ 🖍️ 🗑️
⌚	mailingAddress	OBJECT	-	+ 🖍️ 🗑️
	lastName	STRING	yes	no
	firstName	STRING	yes	no

Attributes type

Add attribute

Type: STRING

Attribute name: ^

! Name is required

Description:

Example value:

Use in reporting

Sensitive Data

i Sensitive data will be hidden for unauthorised personnel. More info [here](#)

Close Save

The Data Model supports the following attribute types:

- STRING
- NUMBER
- BOOLEAN
- OBJECT

- ARRAY
 - ARRAY OF STRINGS
 - ARRAY OF NUMBERS
 - ARRAY OF BOOLEANS
 - ARRAY OF OBJECTS
 - ARRAY OF ENUMS
- ENUM

 **INFO**

When you export or import a **process definition**, the data model will be included.

Data model reference

You can use data model reference feature to view attribute usage within the data model. You can now easily see where a specific attribute is being used by accessing the "View References" feature. This feature provides a list of process keys associated with each attribute and displays possible references, such as UI Elements.

For UI Elements, the references include the element label, node name, and UI Element key. Additionally, the context of the reference is provided, showing the node name and the UI element type along with its label. Users can conveniently navigate to the context by clicking the provided link to the node's UI page.

The screenshot shows the FLOWX.AI Data Model interface. On the left, there's a sidebar with icons for Home, Create, Recent, and Settings. The main area has a title bar with 'data_model | Version 1 | draft' and buttons for '+ New item', 'Start', and a three-dot menu. The main content is a table titled 'Data Model' with the following columns: Name, Type, Used in reporting, Sensitive data, and Description. The table contains the following data:

Name	Type	Used in reporting	Sensitive data	Description
firstName	STRING	no	no	
lastName	STRING	no	no	
dateOfBirth	STRING	no	no	
segmentedKey	STRING	no	no	
switch	BOOLEAN	no	no	
> test_object	OBJECT	no	no	

Sensitive data

To protect your data and your customer's data, you can hide data that could be visible in the process details or in the browser's console. You can now also secret data for a specific key.

X

Add Attribute

Attribute name**Type**

street

Placeholder

**Enumeration (Opt)**

Placeholder

**Example Value**

Placeholder

Description (Opt)

first steps of the process without the br

**Use in reporting****Sensitive data**

Sensitive data will be hidden for unauthorised personnel. More info [here](#)

[Cancel](#)[Save](#)

Reporting

The **Use in Reporting** tag is used for keys that will be used further in the reporting plugin.

» Reporting

Generating data model

A data model can be generated using data values from a **process instance**. This can be done by either merging the data model with an existing one or replacing it entirely.

To generate a data model, follow these steps:

1. Open **FLOWX.AI Designer**.
2. Go to the **Definitions** tab and select the desired **process definition**.
3. Select the **Data Model** tab and then click **Generate data model** button.
4. Add the **process instance** of the process from which you want to generate the data model.
5. Choose whether to **replace** the existing data model or **merge** it with the new one.
6. Click the **Load Data** button to display the data model body.
7. Finally, click **Save** button to save the generated data model.

The screenshot shows the FLOWX.AI interface with a sidebar on the left containing icons for Home, New item, Start, and Help. The main area displays a "Data Model" section titled "generate_data_model | Version 1 | draft". The table has columns: Name, Type, Used in reporting, Sensitive data, and Description. A message "No attributes found" is displayed below the table.

By generating a data model, you can ensure that your data is structured and organized in a way that is appropriate for your business needs. It can also help you to identify any inconsistencies or errors in the data, allowing you to correct them before they cause problems down the line.

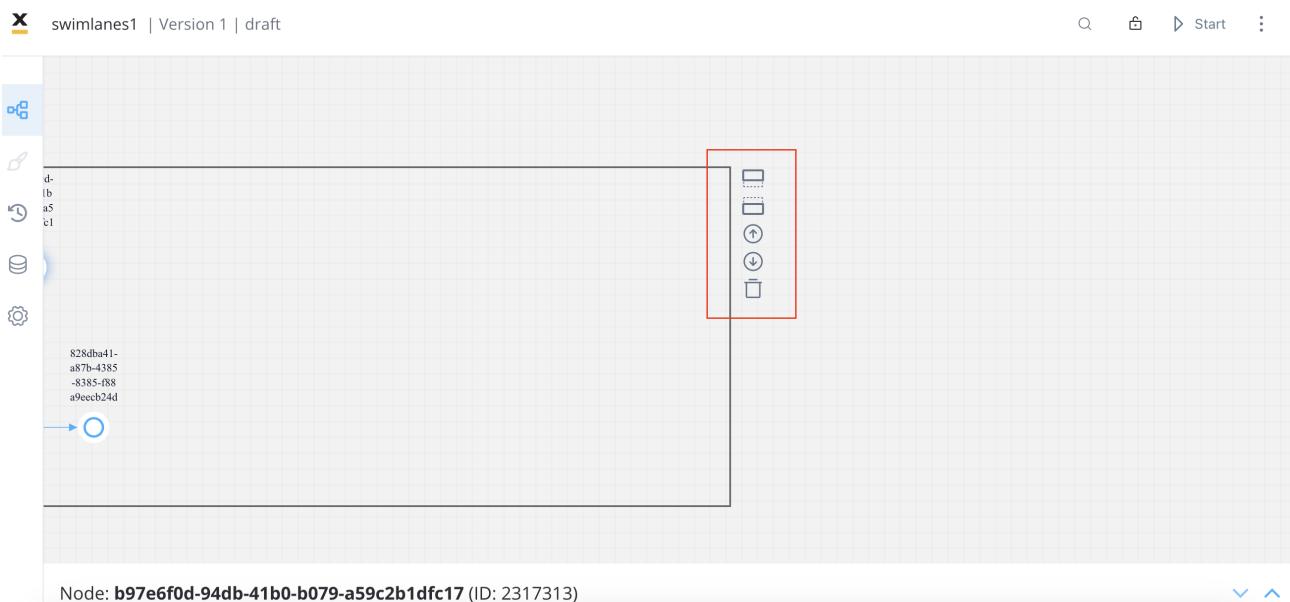
Swimlanes

Swimlanes offer a useful method of organizing process nodes based on process participants. By utilizing swimlanes, you can establish controlled access to specific process nodes for particular user roles.

Adding new swimlanes

To add new swimlanes, please follow these steps:

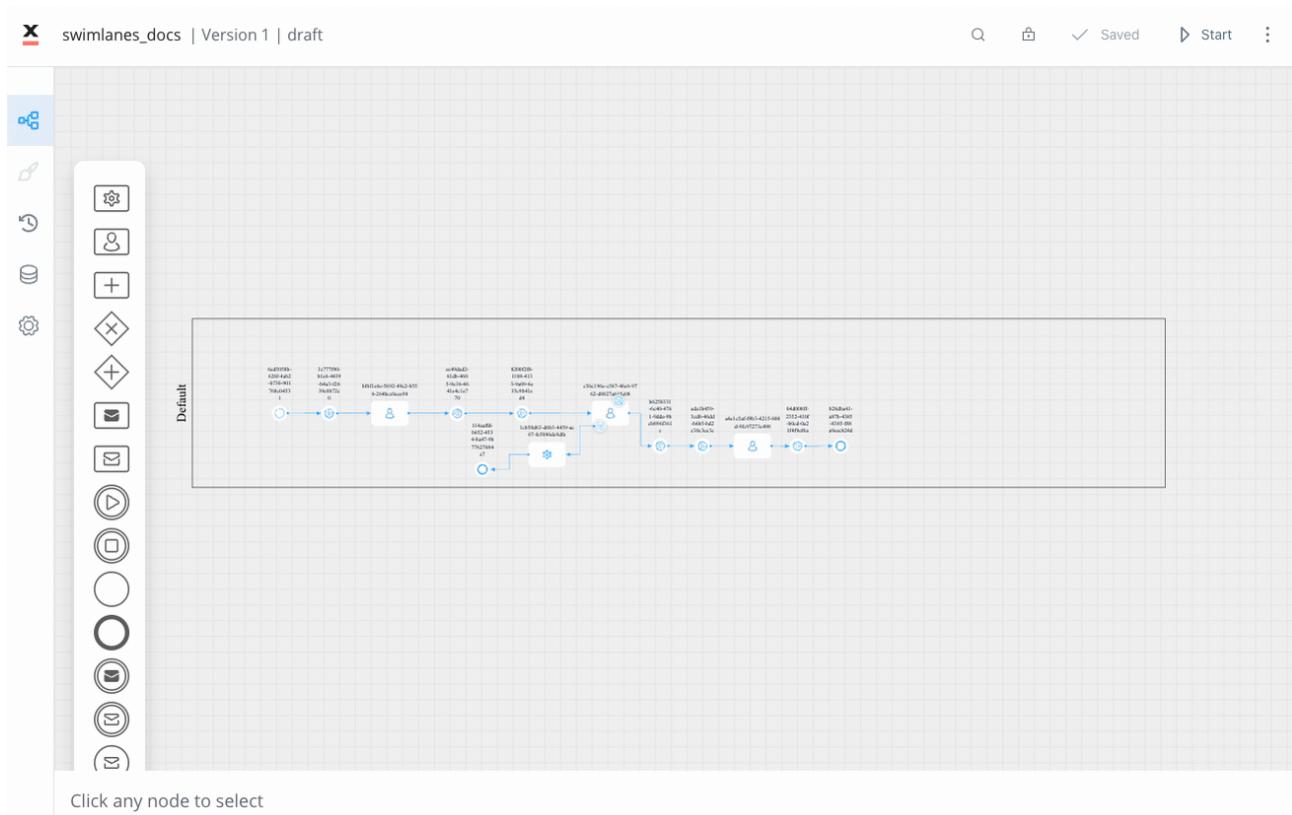
1. Access the **FLOWX.AI Designer**.
2. Open an existing process definition or create a new one.
3. Identify the default swimlane and select it to display the contextual menu.



INFO

With the contextual menu, you can easily perform various actions related to swimlanes, such as adding or removing swimlanes or reordering them.

4. Choose the desired location for the new swimlane, either below or above the default swimlane.
5. Locate and click the **add swimlane icon** to create the new swimlane.



For more details about user roles management, check the following section:

» [User roles management - Swimlanes](#)

For more details about setting up user role-based access on process definitions, check the following section:

» [Configuring access roles for processes](#)

Settings

General

In the General settings, you can edit the process definition name, include the process in reporting, set general data, and configure expiry time using Cron Expressions and ISO 8601 formatting.

- **Process definition name** - edit process definition name
- **Use process in reporting** - if switched on, the process will be included in reporting
- **Use process in task management** - if switched on, tasks will be created and displayed in the Task manager plugin, more information [here](#)
- **General data** - data that you can set and receive on a response
- **Expiry time** - a user can set up a `expiryTime` function on a process, for example, a delay of 30s will be set up like: `30 16 11 4 7 1`

For more information about **Cron Expressions** and **ISO 8601** formatting, check the following section;

» [Timer Expressions](#)

FLOWX.AI test-process (version 1, draft) [Start](#) [⋮](#)

The screenshot shows the FLOWX.AI process configuration interface. On the left is a sidebar with icons for Home, Create, History, and Settings (selected). The main area has tabs: General*, Sensitive data, Swimlanes, Permissions, and Task management. The General tab is selected. It contains fields for Process definition name (set to 'test-process'), a toggle for Use process in reporting (unchecked), and a large General data section which is currently empty. Below this is an Expiry time field containing 'ex: PT3M22S or 0 0 9-17 * * MON-FRI' with a note about Cron Expressions. At the bottom right is a blue 'Save settings' button.

General*

Sensitive data Swimlanes Permissions Task management

Process definition name

test-process

Use process in reporting

General data

Expiry time

ex: PT3M22S or 0 0 9-17 * * MON-FRI

For more details about Expiry time formatting and examples of valid Cron Expressions, click [here](#).

Save settings

Permissions

After defining roles in the identity provider solution, they will be available to be used in the process definition settings panel for configuring swimlane access.

When you create a new swimlane, it comes with two default permissions assigned based on a specific role: execute and self-assign. Other permissions can be added manually, depending on the needs of the user.

The screenshot shows the 'Permissions' tab selected in a top navigation bar. On the left, a sidebar lists 'Choose swimlane' and 'docs 3.0' with roles 'SWIMLANE_USER' and 'SWIMLANE_SUPERVISOR'. The main panel displays a 'Default' role configuration for 'docs 3.0'. It shows two roles assigned: 'FLOWX_ROLE' with permissions 'Execute, Self Assign' and 'ROLE_ADMIN' with permissions 'View, Execute, Self Assign, Unassign, Assign'. There are 'Add Role' and 'Save permissions' buttons at the bottom.

» [Configuring access rights for processes](#)

Task management

The Task Management plugin offers a business-oriented view of the process you defined in the Designer and allows for interactions at the assignment level. It also

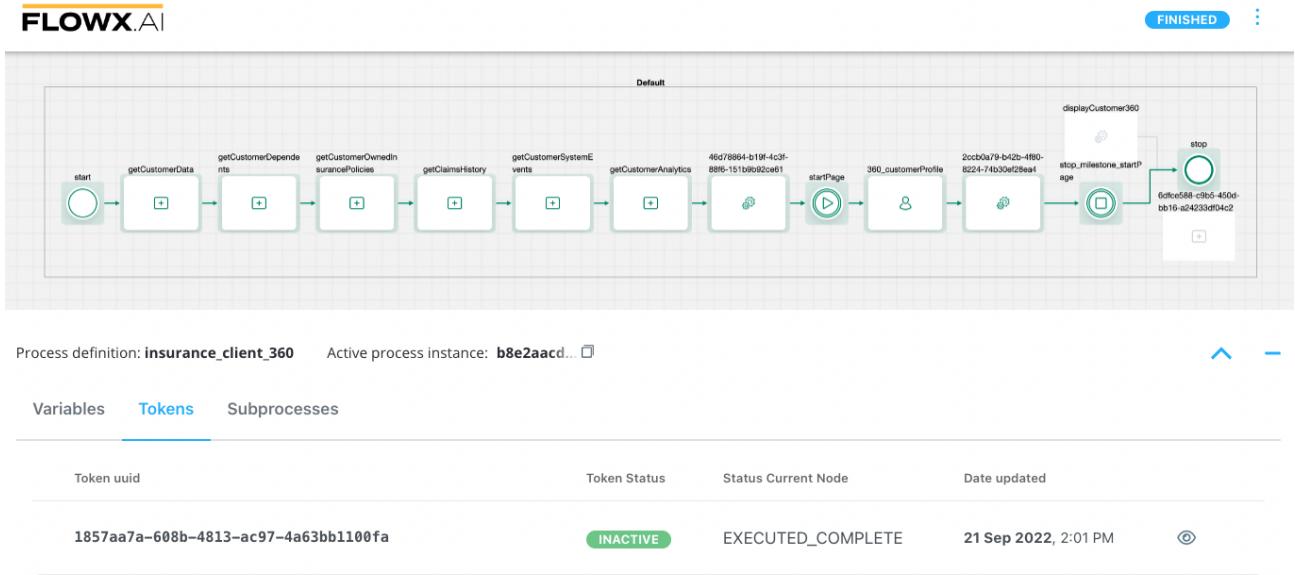
includes a generic parameter pointing to the application URL where the Flowx process is loaded and uses process keys to search data stored in the process.

The screenshot shows a sidebar with icons for General, Swimlanes, Permissions, and Task management. The Task management icon is highlighted with a blue background. The main area displays the 'Task management' tab, which includes a section for 'Application url' with a descriptive note: 'Generic parameter pointing to the application url where the flowx process is loaded.' Below this is a 'Search indexing' section with a note: 'Process keys will be used to search data stored in the process.' A blue 'Add Key' button is visible at the bottom of this section.

Was this page helpful?

BUILDING BLOCKS / Process Designer / Active process / Process instance

A process instance is a specific execution of a business process that is defined on the FLOWX.AI platform. Once a process definition is added to the platform, it can be executed, monitored, and optimized by creating an instance of the definition.



Overview

Once the desired processes are defined in the platform, they are ready to be used. Each time a process needs to be used, for example each time a customer wants to request, for example, a new credit card, a new instance of the specified process definition is started in the platform. Think of the process definition as a blueprint for a house, and of the process instance as each house of that type being built.

The **FLOWX Engine** is responsible for executing the steps in the process definition and handling all the business logic. The token represents the current position in the process and moves from one node to the next based on the sequences and rules defined in the exclusive gateways. In the case of parallel gateways, child tokens are created and eventually merged back into the parent token.

Kafka events are used for communication between FLOWX.AI components such as the engine and integrations/plugins. Each event type is associated with a Kafka

topic to track and orchestrate the messages sent on Kafka. The engine updates the UI by sending messages through sockets.

» More about Kafka

Checking the Process Status

To check the status of a process or troubleshoot a failed process, follow these steps:

1. Open **FLOWX Designer**.
2. Go to **Processes** → **Active Process** → **Process instances**.
3. Click **Process status** button.

Active processes					
Process uuid	Definition name	Exceptions	Status	Current Node Name	Start date
cb19bfc0-cacb-4cbe-96bc-73c5771c0699	Amazing Process	-	CREATED	bddd53ae-125e-4f46-8288-6beae4424219	09 Aug 2022, 9:36 PM
abc17f2b-f3fb-48e1-a024-131692ade558	Super Process	-	STARTED	node_name	09 Aug 2022, 8:35 PM
3a4849fb-85ec-4383-9402-6f14d37d840e	Awesome Process	-	STARTED	user_task	09 Aug 2022, 8:32 PM
6989a30a-370a-4dc8-9aa8-e605bbe9772e	Incredible Process	-	STARTED	user_task	09 Aug 2022, 8:30 PM
4a69ff3c-6a48-46c6-9292-877f9bd7cf69	Nice Process	-	EXPIRED	page_1	09 Aug 2022, 7:47 PM

Understanding the Process Status Data

The process status data includes the following:

The screenshot shows a process instance details page for an 'Amazing Process'. At the top, there's a process diagram with four states: a red circle (initial state), a play button (active state), a square (dismissible state), and a green circle (finished state). Below the diagram, the process definition name is 'Amazing Process' and the active process instance ID is 'cb19bfc0...'. A modal window on the right provides detailed status information: 'CREATED', 'Version: 4', 'Started: 09 Aug 2022 at 9:36 PM', and 'Ended: -'. Below the modal, there are tabs for 'Variables', 'Tokens', and 'Subprocesses', and a code block showing the current token variables:

```

processInstanceId: 620353
tokenId: 620403
tokenUuid: "f3e76161-3ebc-4dd5-8baf-a921bc499126"
webSocketPath: "/ws/updates/process"
processInstanceUid: "cb19bfc0-cacb-4cbe-96bc-73c5771c0699"
webSocketAddress: "wss://public.qa.flowxai.dev/cb19bfc0-cacb-4cbe-96bc-73c5771c0699"

```

- **Status** - status of the process instance, possible values:
 - CREATED - the status is visible if there is an error in the process creation. If there is no error, the "Started" status is displayed.
 - STARTED - indicates that the process is currently running
 - DISMISSED - the status is available for processes with subprocesses, it is displayed when a user stops a subprocess
 - EXPIRED - the status is displayed when the "expiryTime" field is defined in the process definition and the defined time has passed.
 - FINISHED - the process has successfully completed its execution
- **Process definition** - the name of the process definition
- **Active process instance** - the UUID of the process instance, with a copy action available

- **Variables** - displayed as an expanded JSON

Process definition: **Awesome Process** Active process instance: **12156183...** 

[Variables](#) [Tokens](#) [Subprocesses](#) [Exceptions](#) 10

```
processInstanceId: 619702
tokenId: 619752
tokenUuid: "147aa0c4-d961-4154-8af6-3422e1d34fb6"
webSocketPath: "/ws/updates/process"
processInstanceUuid: "12156183-b40d-4cee-b0e0-e296371cedbf"
webSocketAddress: "wss://public.qa.flowxai.dev/12156183-b40d-4cee-b0e0-e296371cedbf"
```

- **Tokens** - a token represents the state within the process instance and describe the current position in the process flow

Process definition: **Awesome Process** Active process instance: **12156183...** 

[Variables](#) [Tokens](#) [Subprocesses](#) [Exceptions](#) 10

Token uuid	Token Status	Status Current Node	Date updated	
147aa0c4-d961-4154-8af6-3422e1d34fb6	ACTIVE	EXECUTED_COMPLETE	09 Aug 2022, 4:06 PM	 

(!) INFO

For more information about token status details, here.

- **Subprocesses** - **!** displayed only if the current The fallback content to display on prerendering generated a **subprocess** instance

- **Exceptions** - errors that let you know where the process is blocked, with a direct link to the node where the process is breaking for easy editing

Process definition: **Awesome Process** Active process instance: **12156183...**  

Variables	Tokens	Subprocesses	Exceptions 10
Source	Message	Type	Timestamp
NODE (start Node)	Outgoing node for process instance not fo...	Node Definition	09 Aug 2022 at 4:06 F  
NODE (start Node)	Outgoing node for process instance not fo...	Node Definition	09 Aug 2022 at 4:06 F  
NODE (start Node)	Outgoing node for process instance not fo...	Node Definition	09 Aug 2022 at 4:06 F  

INFO

For more information on token status details and exceptions, check the following section:

» Failed process start

- **Audit Log** - the audit log displays events registered for process instances, tokens, tasks, and exceptions in reverse chronological order by timestamp

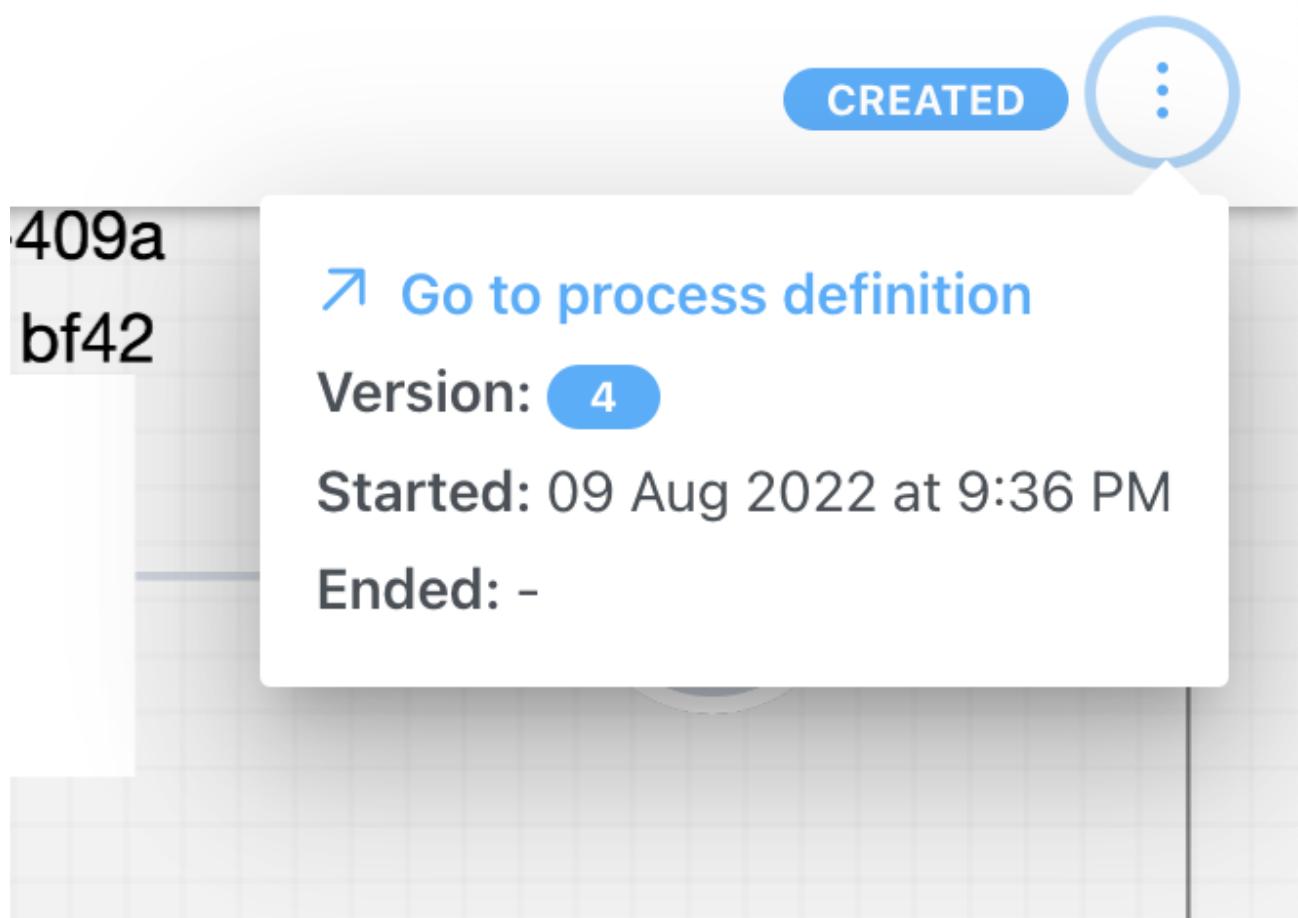
Variables	Tokens	Subprocesses	Audit Log	Exceptions 1
Timestamp	User	Subject	Event	Subject Identifier
06 Oct 2022, 1:54 PM	john.doe@email.com	Exception	View	-
06 Oct 2022, 1:54 PM	system	Exception	Created	714951
06 Oct 2022, 1:54 PM	john.doe@email.com	Process Instance	Start	a310c7b5-fb1c-4cf8-9b61-6badd174 success

» Audit

Process menu

In the breadcrumb menu (top-right corner), you can access the following:

- **Go to process definition** - opens the process for editing
- **Version** - version of the process definition
- **Started** - timestamp for when the process instance started
- **Ended** - timestamp for when the process instance ended



Color coding

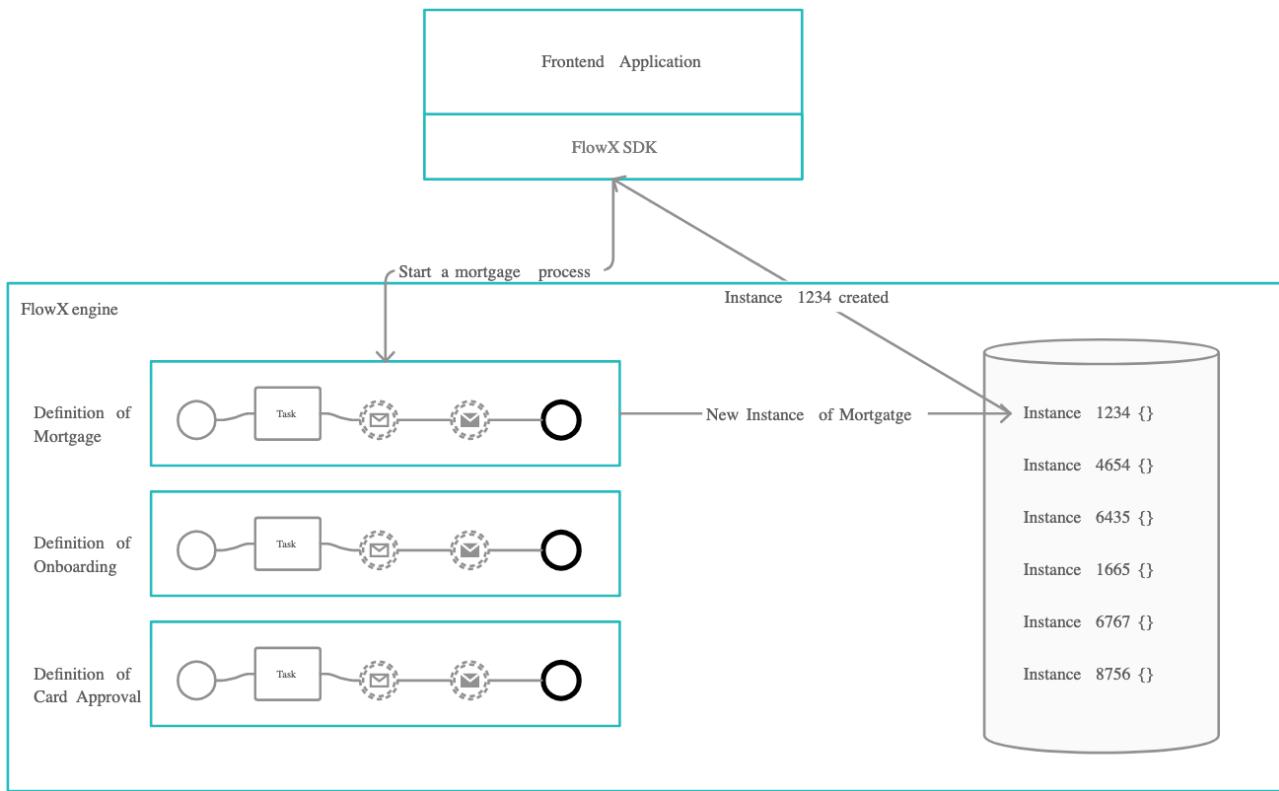
In the **Process Status** view, some nodes are highlighted with different colors to easily identify any failures:

- **Green** - nodes highlighted with green mark the nodes passed by the token
- **Red** - the node highlighted with red marks the node where the token is stuck (process failure)

Process uid	Definition name	Status	Current Node Name	Start date
a2f7ff1ef-ad39-4c00-8e58-492059f6e670	test_render_proc_instance	STARTED	Step	22 Jul 2022, 4:41 PM
e14bcfd3-a10e-4599-b0b4-e76669685fed	test_render_proc_instance	CREATED	Start	22 Jul 2022, 4:37 PM
037a6efe-0949-4f2b-ac8e-a94afe88d287	test_render_proc_instance	FINISHED	End	22 Jul 2022, 4:34 PM
0df92b59-6195-472a-bb7e-cb9cd98e0ad0	test_render_proc_instance	STARTED	Step	22 Jul 2022, 4:33 PM
2d403707-c480-48d0-a7ce-1d1603fa0ebc	test_render_proc_instance	STARTED	Step	22 Jul 2022, 4:31 PM
4fbe4e81-09a6-49d0-9f0b-aba566b896d8	test_render_proc_instance	CREATED	Step	22 Jul 2022, 4:04 PM

Starting a new process instance

To start a new process instance, a request must be made to the **FLOWX Engine**. This is handled by the web/mobile application. The current user must have the appropriate role/permission to start a new process instance.



To be able to start a new process instance, the current user needs to have the appropriate role/permissions:

» [Configuring access roles for processes](#)

When starting a new process instance, we can also set it to [inherit some values](#) from a previous process instance.

Troubleshooting possible errors

If everything is configured correctly, the new process instance should be visible in the UI and added to the database. However, if you encounter issues, here are

some common error messages and their possible solutions: Possible errors include:

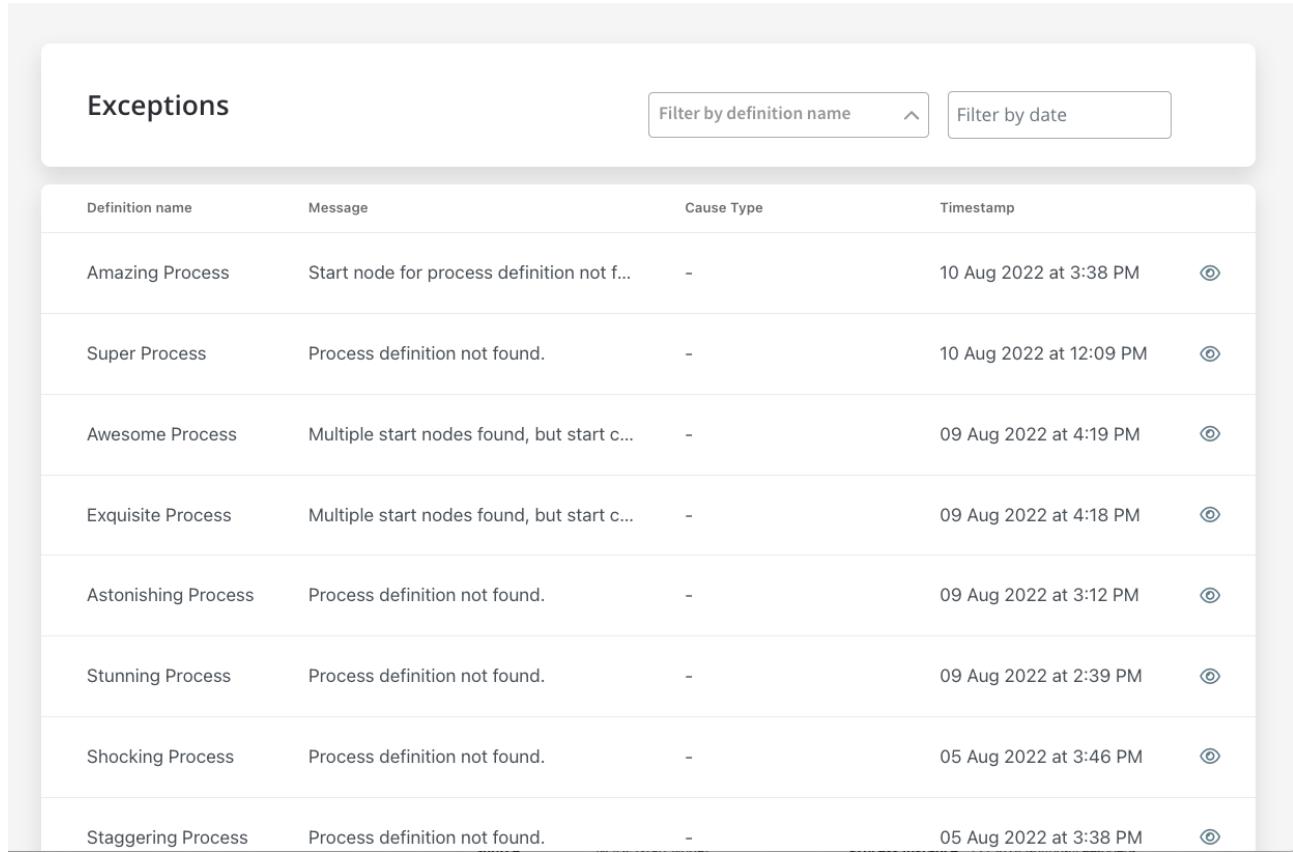
Error Message	Description
<i>"Process definition not found."</i>	The process definition with the requested name was not set as published.
<i>"Start node for process definition not found."</i>	The start node was not properly configured.
<i>"Multiple start nodes found, but start condition not specified."</i>	Multiple start nodes were defined, but the start condition to choose the start node was not set.
<i>"Some mandatory params are missing."</i>	Some parameters set as mandatory were not included in the start request.
HTTP code 403 – Forbidden	The current user does not have the process access role for starting that process.
HTTP code 401 – Unauthorized	The current user is not logged in.

Was this page helpful?

BUILDING BLOCKS / Process Designer / Active process / Failed process start

Exceptions

Exceptions are types of errors meant to help you debug a failure in the execution of a process.



The screenshot shows a table titled 'Exceptions' with columns for Definition name, Message, Cause Type, and Timestamp. There are eight rows of data, each with a small circular icon on the right. The table is part of a larger interface with search and filter options at the top.

Definition name	Message	Cause Type	Timestamp	
Amazing Process	Start node for process definition not f...	-	10 Aug 2022 at 3:38 PM	🔗
Super Process	Process definition not found.	-	10 Aug 2022 at 12:09 PM	🔗
Awesome Process	Multiple start nodes found, but start c...	-	09 Aug 2022 at 4:19 PM	🔗
Exquisite Process	Multiple start nodes found, but start c...	-	09 Aug 2022 at 4:18 PM	🔗
Astonishing Process	Process definition not found.	-	09 Aug 2022 at 3:12 PM	🔗
Stunning Process	Process definition not found.	-	09 Aug 2022 at 2:39 PM	🔗
Shocking Process	Process definition not found.	-	05 Aug 2022 at 3:46 PM	🔗
Staggering Process	Process definition not found.	-	05 Aug 2022 at 3:38 PM	🔗

Exceptions can be accessed from multiple places:

- **Failed process start** tab from **Active process** menu in FLOWX Designer

- **Process Status** view, accessible from **Process instances** list in FLOWX Designer

Process definition: **Awesome Process** Active process instance: **12156183...** □

Variables Tokens Subprocesses Exceptions 10

Source	Message	Type	Timestamp	Actions
NODE (start Node)	Outgoing node for process instance not fo...	Node Definition	09 Aug 2022 at 4:06 F	ⓘ ⓧ
NODE (start Node)	Outgoing node for process instance not fo...	Node Definition	09 Aug 2022 at 4:06 F	ⓘ ⓧ
NODE (start Node)	Outgoing node for process instance not fo...	Node Definition	09 Aug 2022 at 4:06 F	ⓘ ⓧ

⚠ CAUTION

If you open a process instance and it does not contain exceptions, the **Exceptions** tab will not be displayed.

Exceptions data

When you click **view** button, a detailed exception will be displayed.

Exceptions: 619820

Process Definition:	Awesome Process	Cause Type:	START_EVENT
Source:	NODE (start Node)	Process Instance UUID:	12156183-b40d-4cee-b0e0-e296371cedbf
Message:	Outgoing node for process instance not found.	Token UUID:	147aa0c4-d961-4154-8af6-3422e1d34fb6
Type:	Node Definition	Timestamp:	09 Aug 2022 at 4:06 PM

Details

```

1 [ai.flowx.enginedefinitions.dto.NodeDTOWrapper.lambda$getOutgoingNodeMandatory$4(NodeDTOWrapper.java:45),
java.base/java.util.Optional.orElseThrow(Optional.java:408), ai.flowx.enginedefinitions.dto.NodeDTOWrapper.
getOutgoingNodeMandatory(NodeDTOWrapper.java:45), ai.flowx.engine.instance.service.impl.
NodeProcessorServiceImpl.getNextNodeId(NodeProcessorServiceImpl.java:249), ai.flowx.engine.instance.service.
impl.NodeProcessorServiceImpl.process(NodeProcessorServiceImpl.java:155), ai.flowx.engine.instance.service.
impl.NodeProcessorServiceImpl$$FastClassBySpringCGLIB$$badf4a43.invoke(<generated>), org.springframework.
cglib.proxy.MethodProxy.invoke(MethodProxy.java:218), org.springframework.aop.framework.
CglibAopProxy$CglibMethodInvocation.invokeJoinpoint(CglibAopProxy.java:779), org.springframework.aop.
framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:163), org.springframework.aop.
framework.CglibAopProxy$CglibMethodInvocation.proceed(CglibAopProxy.java:750), org.springframework.aop.
aspectj.MethodInvocationProceedingJoinPoint.proceed(MethodInvocationProceedingJoinPoint.java:89), ai.flowx.
commons.trace.aop.JaegerTraceAspect.around(JaegerTraceAspect.java:52), jdk.internal.reflect.
GeneratedMethodAccessor582.invoke(Unknown Source), java.base/jdk.internal.reflect.
DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43), java.base/java.lang.reflect.Method.
invoke(Method.java:566), org.springframework.aop.aspectj.AbstractAspectJAdvice.invokeAdviceMethodWithGivenArgs
(AbstractAspectJAdvice.java:634), org.springframework.aop.aspectj.AbstractAspectJAdvice.invokeAdviceMethod
(AbstractAspectJAdvice.java:624), org.springframework.aop.aspectj.AspectJAroundAdvice.invoke
(AspectJAroundAdvice.java:72), org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
(ReflectiveMethodInvocation.java:175), org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.
proceed(CglibAopProxy.java:750)]

```

- **Process Definition** - the process where the exception was thrown
- **Source** - the source of the exception (see the possible type of **sources** below)
- **Message** - a hint type of message to help you understand what's wrong with your process

- **Type** - exception type
- **Cause Type** - cause type (or the name of the node)
- **Process Instance UUID** - process instance unique identifier
- **Token UUID** - token unique identifier
- **Timestamp** - default format: yyyy-MM-dd 'T'HH:mm:ss.SSSZ
- **Details** - stack trace (a **stack trace** is a list of the method calls that the process was in the middle of when an **Exception** was thrown)

Possible sources:

- Action
- Node
- Subprocess
- Process Definition

Exceptions type

Based on the exception type, there are multiple causes that could make a process fail. Here are some examples:

Type	Cause
Business Rule Evaluation	when executing action rules fails for any reason
Condition Evaluation	when executing action conditions

Type	Cause
Engine	<p>when the connection with the database fails</p> <p>when the connection with Redis fails</p>
Definition	misconfigurations: process def name, subprocess parent process id value, start node condition missing
Node	when an outgoing node can't be found (missing sequence etc)
Gateway Evaluation	<p>when the token can't pass a gateway for any reason, possible causes:</p> <ul style="list-style-type: none"> • missing sequence/node • failed node rule
Subprocess	exceptions will be saved for them just like for any other process, parent process ID will also be saved (we can use this to link them when displaying exceptions)

Was this page helpful?

BUILDING BLOCKS / Process Designer / Subprocess

Sub-processes are smaller process flows that can be triggered by actions in the main process. They can also inherit some process parameter values from the parent process and send their results back to the parent process when they are completed. The subprocesses will communicate with the front-end apps using the same connection details as their parent process.

They can be started in two ways:

- **asynchronous** - they will execute alongside the parent process since the parent process does not need to wait for the sub-process to end
- **synchronous** - the parent process will wait until the sub-processes are finished before advancing

Configuring & starting subprocesses

The sub-processes will be designed in the same way as the main process, by using the FLOWX Designer.

They can be started by a parent process in one of two ways:

- by using a StartSubprocess action inside any of the task nodes in the process
- by adding a custom Subprocess Run node type in the process

In both cases, by default, the sub-process will inherit all the parent process parameter values. It can be configured to inherit only some parameter values from

its parent. The available action parameters for this are:

- *paramsToCopy* - choose which of the keys from the parent process parameters to be copied to the sub-process
- *withoutParams* - choose which of the keys from the parent process parameters are to be ignored when copying parameter values from the parent process to the sub-process

Sub-processes can have an action configured on them which will append their results to the parent process parameter values.

Executing subprocesses

The sub-processes can be started in async or sync mode, by setting a specific action parameter, named *startedAsync*, on the action that triggers the subprocess.

If the subprocesses are started in sync mode, they will notify the parent process when they are completed and the parent process will handle receiving the process data from the child and resuming its flow.

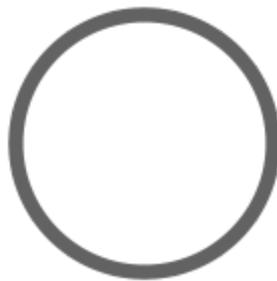
[Was this page helpful?](#)

BUILDING BLOCKS / Node / Start/End nodes

Let's go through all the options for configuring start and end nodes for a process definition.

Start node

The start node represents the beginning of a process and it is mandatory to add one when creating a process.



A process can have one or more start nodes. If you defined multiple start nodes, each should have a start condition value configured. When starting a new process instance the desired start condition should be used.

The screenshot shows the FLOWX.AI process editor interface. At the top, there is a toolbar with icons for node types (Start, User Task, Parallel, Exclusive OR), a search bar labeled "Search by node name", and a "Start Node Example" label next to a small blue circle icon representing the start node. Below the toolbar, the main workspace shows a single start node. The bottom half of the screen is a configuration panel for the selected node:

- Node Config** tab is active.
- General Config** section:
 - Node name**: Start Node Example
 - Can go back?**: A toggle switch is set to **On**.
- Swimlane** dropdown: Default
- Stage** dropdown: Stage
- Save** button: A blue button at the bottom right.

Configuring a start node

Node configuration is done by accessing the **Node Config** tab. You have the following configuration options for a **start node**:

- **General Config**
- **Start condition**

General Config

- **Node name** - the name of the node
- **Can go back** - switching this option to true will allow users to return to this step after completing it

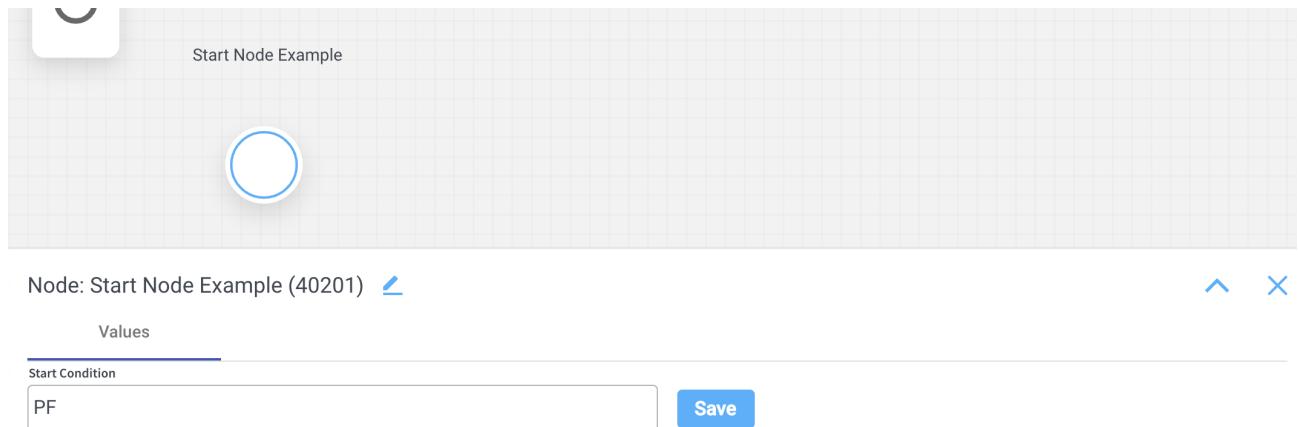
(!) INFO

When encountering a step with `canGoBack` switched to false, all steps found behind it will become unavailable.

- **Swimlane** - choose a swimlane (if there are multiple swimlanes on the process) to make sure only certain user roles have access only for certain process nodes- if there are no multiple swimlanes, the value is **Default**
- **Stage** - assign a stage to the node

Start condition

The start condition should be set as a string value. This string value will need to be set on the payload for the start process request on the `startCondition` key.



To test the start condition, we can send a start request via REST:

```
POST {{processUrl}}/api/process/{{processName}}/start
{
    "startCondition": "PF"
}
```

Error handling on start condition

If a request is made to start a process with a start condition that does not match any start node, an error will be generated. Let's take the previous example and assume we send an incorrect value for the start condition:

```
POST {{processUrl}}/api/process/{{processName}}/start
{
    "startCondition": "PJ"
}
```

A response with the error code `bad request` and title `Start node for process definition not found` will be sent in this case:

```
{  
    "entityName": "ai.flowx.process.definition.domain.NodeDefinition",  
    "defaultMessage": "Start node for process definition not found",  
    "errorKey": "error.validation.process_instance.start_node_for_process_definition_not_found",  
    "type": "https://www.jhipster.tech/problem/problem-with-message",  
    "title": "Start node for process definition not found.",  
    "status": 400,  
    "message": "error.validation.process_instance.start_node_for_process_definition_not_found",  
    "params": "ai.flowx.process.definition.domain.NodeDefinition"}  
}
```

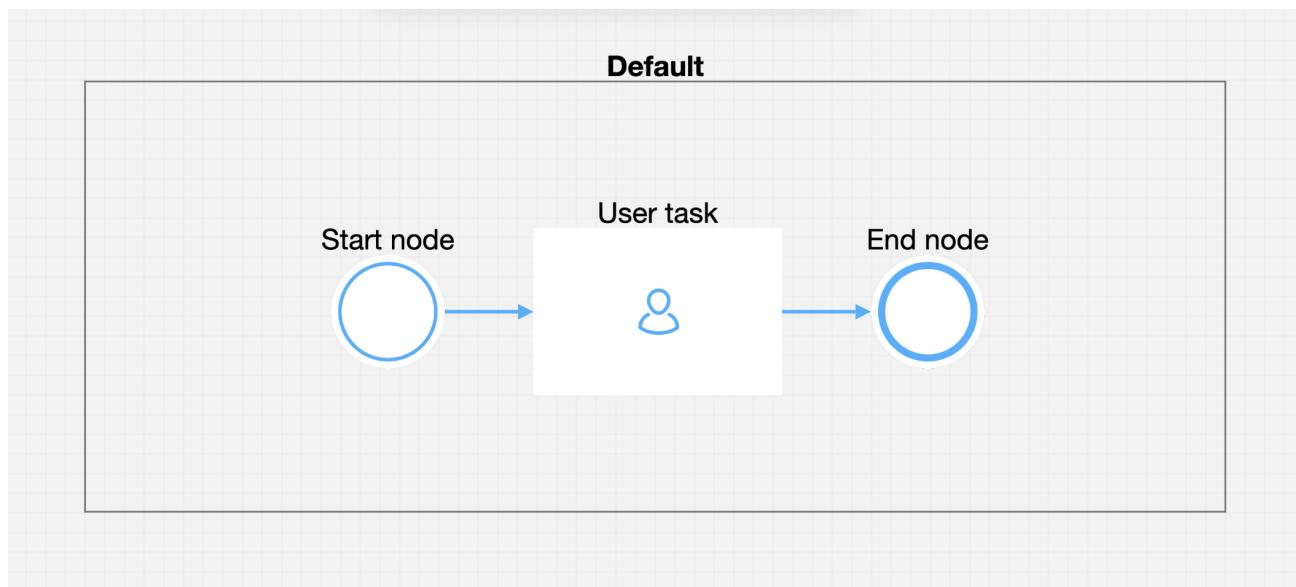
End node



An end node is used to mark where the process finishes. When the process reaches this node, the process is considered completed and its status will be set to `Finished`.

Configuring an end node

Multiple end nodes can be used to show different end states. The configuration is similar to the start node.



Was this page helpful?

BUILDING BLOCKS / Node / Message send/Message received task nodes

Message send task and message received

The fallback content to display on prerendering are used to handle the interaction between a running process and any external systems.

Message send task

This node is used to configure messages that should be sent to external systems.



Configuring a message send task node

Node configuration is done by accessing the **Node Config** tab. You have the following configuration options for a message send task node:

General Config

Inside the General Config you have the following properties:

- **Node name** - the name of the node
- **Can Go Back** - switching this option to true will allow users to return to this step after completing it

(!) INFO

When encountering a step with `canGoBack` switched to false, all steps found behind it will become unavailable.

- **Swimlane** - choose a swimlane (if there are multiple swimlanes on the process) to make sure only certain user roles have access only for certain process nodes - if there are no multiple swimlanes, the value is **Default**
- **Stage** - assign a stage to the node

Node: **Message send task** (ID: 546054)

Node Config Actions

General Config

Node name

Message send task

Can go back?

Swimlane

Default ▼

Stage

▼

To configure a message send task node, we first need to add a new node and then configure an

The fallback content to display on prerendering

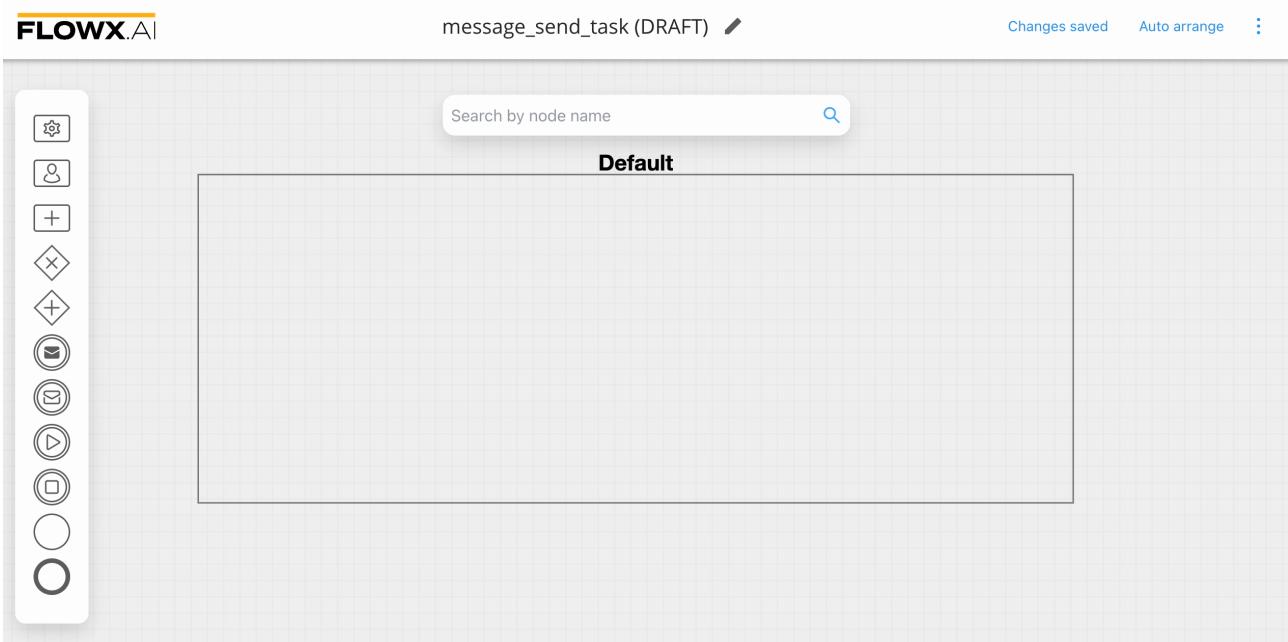
(**Kafka Send Action** type):

1. Open
The fallback content to display on prerendering
and start configuring a process.
2. Add a **message send task** node.
3. Select the **message send task** node and open **node configuration**.

4. Add an

The fallback content to display on prerendering
, the type of the action set to **Kafka send**.

5. ! A few action parameters will need to be filled in depending on the selected action type.



Multiple options are available for this type of action and can be configured via the FLOWX.AI Designer. To configure and [add an action to a node](#), use the

The fallback content to display on prerendering tab at the node level, which has the following configuration options:

- [Action Edit](#)
- [Back in steps \(for Manual actions\)](#)
- [Parameters](#)
- [Data to send \(for Manual actions\)](#)

Action Edit

- **Name** - used internally to make a distinction between different **actions** on nodes in the process. We recommend defining an action naming standard to be able to easily find the process actions
- **Order** - if multiple actions are defined on the same node, the running order should be set using this option
- **Timer expression** - it can be used if a delay is required on that action. The format used for this is **ISO 8601 duration format** (for example, a delay of 30 seconds will be set up as `PT30S`)
- **Action type** - should be set to **Kafka Send Action** for actions used to send messages to external systems
- **Trigger type** (options are Automatic/Manual) - choose if this action should be triggered automatically (when the process flow reaches this step) or manually (triggered by the user); in most use cases, this will be set to automatic
- **Required type** (options are Mandatory/Optional) - automatic actions can only be defined as mandatory. Manual actions can be defined as mandatory or optional.
- **Repeatable** - 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

Back in steps

- **Allow BACK on this action** - back in process is a functionality that allows you to go back in a business process and redo a series of previous actions in the process, or more details, check **Moving a token backwards in a process** section

Action Edit

ID: 540663

Name

f29bcac4-4e42-4e0d-9b66-953d67f272c4

Order

1

Timer Expression

Kafka Send Action



Automatic Manual

Mandatory Optional

Repeatable

Autorun Children?

Data to send

- **Keys** - are 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](#))



Data to send option is configurable only when the action **trigger type** is **Manual**.

Parameters

Address

Replace Values

Message

1

Save

For more information about what Kafka is, check the following sections:

» [Intro to Kafka](#)

» [Kafka documentation](#)

Example of a message send event

Send a message to a CRM integration to request a search in the local database:

Action Edit

- **Name** - pick a name that makes it easy to figure out what this action does, for example, `sendRequestToSearchClient`

- **Order** - 1
- **Timer Expression** - this remains empty if we want to action to be triggered as soon as the token reaches this node
- **Action type** - Kafka Send Action
- **Trigger type** - *Automatic* - to trigger this action automatically
- **Required type** - *Mandatory* - to make sure this action will be run before advancing to the next node
- **Repeatable** - false, it only needs to run once

Parameters

ⓘ INFO

Parameters can be added either using **Custom** option (where you configure everything on the spot), or by using **From integration** and import parameters already defined in an integration.

More details about **Integrations management** you can find [here](#).

Custom

- **Topics** - `ai.flowx.in.crm.search.v1` the Kafka topic on which the CRM listens for requests
- **Message** - `{ "clientType": "${application.client.clientType}", "personalNumber": "${personalNumber.client.personalNumber}" }` - the message payload will have two keys, `clientType` and `personalNumber`, both with values from the process instance
- **Headers** - `{"processInstanceId": ${processInstanceId}}`

Action Edit

ID: 540663

Name

sendRequestToSearchClient

Order

1

Timer Expression

Kafka Send Action



Automatic Manual

Mandatory Optional

Repeatable

Parameters

Custom

From integration

Topics

```
ai.flowx.in.crm.search.v1
```

Message

```
1 {  
2   "clientType": "${application.client.clientType}",  
3   "personalNumber": "${personalNumber.client.personalNumber}"  
4 }
```

Advanced configuration

Show Headers

Save

Advanced configuration

Show Headers

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

Replace Values

Save

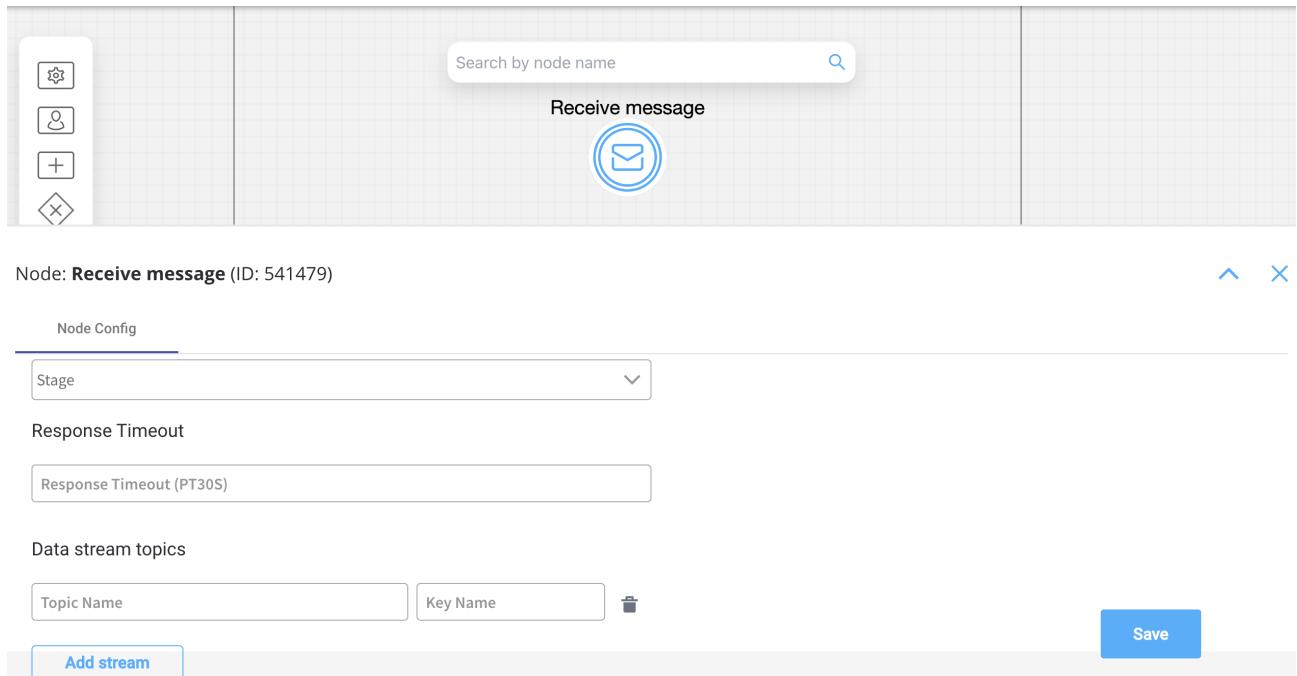
Message receive task

This type of node is used when we need to wait for a reply from an external system.



The reply from the external system will be saved in the process instance values, on a specified key. If the message needs to be processed at a later time, a timeout can be set using the [ISO 8601](#) format.

For example, let's think about a CRM microservice that waits to receive requests to look for a user in a database. It will send back the response when a topic is configured to listen for the response.



Configuring a message receive task node

The values you need to configure for this node are the following:

- **Topic name** - the topic name where the **process engine** listens for the response (this should be added to the platform and match the topic naming rule for the engine to listen to it) - `ai.flowx.out.crm.search.v1`

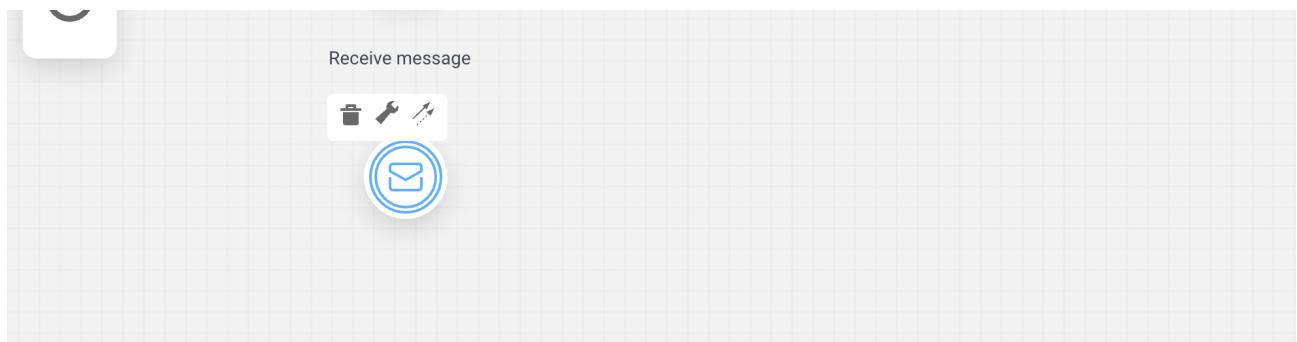
🔥 DANGER

A naming pattern must be defined on the process engine to use the defined topics. It is important to know that all the events that start with a configured pattern will be consumed by the Engine. For example,

`KAFKA_TOPIC_PATTERN` is the topic name pattern that the Engine listens to for incoming Kafka events.

- **Key Name** - will hold the result received from the external system, if the key already exists in the process values, it will be overwritten - `crmResponse`

For more information about Kafka configuration, click [here](#).



Node: Receive message (40221) [🔗](#)

Values

Response Timeout (PT30S)		Save
Topic Name	Key Name	
ai.flowx.out.crm.search.v1	crmResponse	Delete
Topic Name	Key Name	
		Add

From integration

After defining one integration (inside [Integration management](#)) you can open a compatible node and start using already defined integrations.

- **Topics** - topics defined in your integration
- **Message** - the **Message data model** from your integration
- **Headers** - all integrations have `processInstanceId` as a default header parameter, add any other relevant parameters

1

Timer Expression

Kafka Send Action ▾

Automatic Manual

Mandatory Optional

Repeatable

Autorun Children?

Parameters 

Custom From integration

Select a scenario ▾

Save

Was this page helpful?

BUILDING BLOCKS / Node / Task node

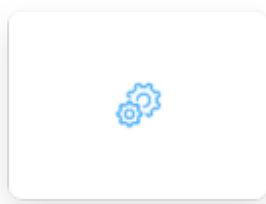
A task

The fallback content to display on prerendering refers to a task that utilizes various services, such as Web services, automated applications, or other similar services, to accomplish a particular task.

This type of node finds application in multiple scenarios, including:

- Executing a
The fallback content to display on prerendering on the process instance data.
- Initiating a
The fallback content to display on prerendering
- Transferring data from a
The fallback content to display on prerendering to the parent process.
- Transmitting data to
The fallback content to display on prerendering

Configuring task nodes



One or more actions can be configured on a task node. The actions are executed in the configured order.

Node configuration is done by accessing the **Node Config** tab. You have the following configuration options for a task node:

General Config

- **Node name** - the name of the node
- **Can go back** - switching this option to true will allow users to return to this step after completing it

Node: **Task node** (ID: 546052)

Node Config	Actions
<p>General Config</p> <p>Node name</p> <div style="border: 1px solid #ccc; padding: 5px; width: fit-content;">Task node</div> <p>Can go back? <input checked="" type="checkbox"/></p>	

! **INFO**

When encountering a step with `canGoBack` switched to false, all steps found behind it will become unavailable.

- **Swimlane** - choose a swimlane (if there are multiple swimlanes on the process) to make sure only certain user roles have access only for certain

process nodes- if there are no multiple swimlanes, the value is **Default**

- **Stage** - assign a stage to the node

Response Timeout

- **Response timeout** - can be triggered if, for example, a topic that you define and add in the **Data stream topics** tab does not respect the pattern, the format used for this is **ISO 8601 duration format**(for example, a delay of 30s will be set up like **PT30S**)

Node: **Task node** (ID: 546052)

Node Config	Actions
Swimlane	<input type="text" value="Default"/> 
Stage	<input type="text" value=""/> 
Response Timeout	<input type="text" value="Response Timeout (PT30S)"/>

Data stream topics

- **Topic Name** - the topic name where the **process engine** listens for the response (this should be added to the platform and match the topic naming rule for the engine to listen to it) - available for UPDATES topics (Kafka receive events)

🔥 DANGER

A naming pattern must be defined on the process engine configuration to use the defined topics. It is important to know that all the events that start with a configured pattern will be consumed by the Engine. For example,

`KAFKA_TOPIC_PATTERN` is the topic name pattern where the Engine listens for incoming Kafka events.

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

Task Management

- **Update task management** - force [Task Manager Plugin](#) to update information about this process after this node

Node: **Task node** (ID: 546052)

Node Config

Actions

Data stream topics

Topic Name

Key Name



Add stream

Task Management

Update task management?

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

Configuring task nodes actions

Multiple options are available when configuring an action on a task node. To configure and add an action to a node, use the **Actions** tab at the node level, which has the following configuration options:

- Action Edit
- Parameters

Action Edit

! INFO

Depending on the type of the **action**, different properties are available, let's take a **Business rule** as an example.

1. **Name** - used internally to differentiate between different actions on nodes in the process. We recommend defining an action naming standard to be able to quickly find the process actions.
2. **Order** - if multiple actions are defined on the same node, their running order should be set using this option
3. **Timer Expression** - can be used if a delay is required on that action. The format used for this is [ISO 8601 duration format](#) (for example, a delay of 30s will be set up like `PT30S`)
4. **Action type** - defines the appropriate action type
5. **Trigger type** - (options are Automatic/Manual) - choose if this action should be triggered automatically (when the process flow reaches this step) or manually (triggered by the user); In most use cases, this will be set to automatic.
6. **Required type** - (options are Mandatory/Optional) - automatic actions can only be defined as mandatory. Manual actions can be defined as mandatory or optional.
7. **Repeatable** - should be checked if the action can be triggered multiple times

Action Edit

ID: 31808

Name

action75

Order

1

Timer Expression

Business Rule



Automatic



Manual



Mandatory



Optional



Repeatable

Parameters

INFO

Depending on the type of the **action**, different properties are available. We refer to a **Business rule** as an example

1. **Business Rules** - business rules can be attached to a node by using actions with action rules on them, these can be specified using **DMN rules**, **MVEL** expressions, or scripts written in Javascript, Python, or Groovy.

» [Supported scripting languages](#)

Business Rule action

A **business rule** is a Task action that allows a script to run. For now, the following script languages are supported:

- **MVEL**
- **JavaScript**
- **Python**
- **Groovy**
- **DMN** - more details about a DMN business rule configuration can be found [here](#)

For more details on how to configure a Business Rule action, check the following section:

» [Business rule action](#)

Send data to user interface

Being an event-driven platform FLOWX uses web socket communication in order to push events from the frontend application. For more details on how to configure a Send data to user interface action, check the following section:

» [Send data to user interface](#)

Upload File action

Upload file action will be used to upload a file from the frontend application and send it via a Kafka topic to the document management system.

For more details on how to configure an Upload File action, check the following section:

» [Upload file action](#)

Start Subprocess action

In order to create reusability between business processes, as well as split complex processes into smaller, easier-to-maintain flows, the start subprocess business rule can be used to trigger the same sequence multiple times.

For more details on how to configure a Business Rule action, check the following section:

» Start subprocess action

Append Params to Parent Process

Used for copying data in the subprocess from its parent process. For more details about the configuration, check the following section:

» Append params to parent process

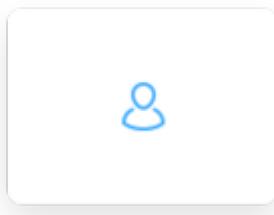
Was this page helpful?

BUILDING BLOCKS / Node / User task node

This

The fallback content to display on prerendering represents an interaction with the user. It is used to display a piece of UI (defined in the [UI Designer](#)) or a [custom Angular component](#). You can also define The fallback content to display on prerendering available for the users to interact with the process.

Configuring a user task node



User task nodes allow you to define and configure UI templates and possible actions for a certain template config node (ex: button components).

General Config

- **Node name** - the name of the node
- **Can go back** - setting this to true will allow users to return to this step after completing it. When encountering a step with `canGoBack` false, all steps found behind it will become unavailable.
- **Flow Names** - leave this field empty if the node should be included in all flows

Node: **User task** (ID: 545152)

Node Config

Actions

General Config

Node name

User task

Can go back?

Flow Names

Leave empty if this node is to be included in all flows

! INFO

When encountering a step with `canGoBack` switched to false, all steps found behind it will become unavailable.

- **Swimlane** - choose a swimlane (if there are multiple swimlanes on the process) to make sure only certain user roles have access only for certain process nodes- if there are no multiple swimlanes, the value is **Default**
- **Stage** - assign a stage to the node

Response Timeout

- **Response timeout** - can be triggered if, for example, a topic that you define and add in the **Data stream topics** tab does not respect the pattern, the format used for this is **ISO 8601 duration format** (for example, a delay of 30s will be set up like `PT30S`)

Node: **User task** (ID: 545152)

Node Config

Actions

Swimlane

Default



Stage



Response Timeout

Response Timeout (PT30S)

The fallback content to display on prerendering

- **Topic Name** - the topic name where the The fallback content to display on prerendering listens for the response (this should be added to the platform and match the topic naming rule for the engine to listen to it) - available for UPDATES topics (Kafka receive events)

DANGER

A naming pattern must be defined on the process engine configuration to use the defined topics. It is important to know that all the events that start with a configured pattern will be consumed by the Engine. For example,

KAFKA_TOPIC_PATTERN is the topic name pattern where the Engine listens for incoming Kafka events.

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

Task Management

- **Update task management** - force **Task Management** The fallback content to display on prerendering to update information about this process after this node

Node: **User task** (ID: 545152)

Node Config	Actions
Data stream topics	
Topic Name	Key Name 
Add stream	

Task Management

Update task management? 

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

Configuring the UI

The

The fallback content to display on prerendering includes an intuitive **UI Designer** (drag-and-drop editor) for creating diverse UI templates. You can use various elements from basic **buttons**, indicators, and **forms**, but also predefined **collections** or **prototypes**.

Accessing the UI Designer

To access the

The fallback content to display on prerendering , follow the next steps:

1. Open **FLOWX Designer** and from the **Processes** tab select **Definitions**.
2. Select a **process** from the process definitions list.
3. Click the **Edit process** button.
4. Select a **user task node** from the Pro dcess Designer then click the **brush** icon to open the **UI Designer**.

The screenshot shows the FLOWX.AI web application 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), and Notification templates. The main area is titled "Process Definitions" and contains two sections: "Drafts / In progress" and "Published".

Drafts / In progress:

Name	Version	Edited at	Edited by	Actions
Test Process	1	06 Jun 2022, 4:33 PM	John Doe	▶ ⚒ ⋮
Test Process 2	2	06 Jun 2022, 9:53 AM	Jane Doe	▶ ⚒ ⋮

Published:

Name	Version	Published at	Published by	Actions
Happy Process	4	26 May 2022, 12:49 PM	John Doe	▶ ⚒ ⋮
Stepper Process	1	24 May 2022, 12:13 PM	Jane Doe	▶ ⚒ ⋮

» Creating a user interface

Predefined components

UI can be defined using the available components provided by FLOWX, using the UI Designer available at node level.

Predefined components can be split in 3 categories:

1. Root components

These elements are used to group different types of components, each having a different purpose:

- **Card** - used to group and configure the layout for multiple **form elements**.

- **Container** - used to group and configure the layout for multiple **components** of any type.
- **Custom** - these are Angular components developed in the container application and passed to the SDK at runtime, identified here by the component name

More details in the following section:

» **Root components**

2. UI Components

The root component can hold a hierarchical component structure.

Available children for **Card** and **Container** are:

- **Container** - used to group and align its children
- **Form** - used to group and align form field elements (**inputs**, **radios**, **checkboxes**, etc)
- **Image** - allows you to configure an image in the document
- **Text** - a simple text can be configured via this component, basic configuration is available
- **Hint** - multiple types of hints can be configured via this component
- **Link** - used to configure a hyperlink that opens in a new tab
- **Button** - Multiple options are available for configuration, the most important part being the possibility to add actions
- **File Upload** - A specific type of button that allows you to select a file

- **Custom** - custom components

More details in the following section:

» [Component types](#)

3. Form elements

This type of elements are used to allow the user to input data, and can be added only in a **Form** Component. They have multiple properties that can be managed.

1. **Input** - FLOWX form element that allows you to generate an input form field
2. **Select** - to add a dropdown
3. **Checkbox** - the user can select zero or more input from a set of options
4. **Radio** - the user is required to select one and only one input from a set of options
5. **Datepicker** - to select a date from a calendar picker
6. **Switch** - allows the user to toggle an option on or off

More details in the following section:

» [Form elements](#)

Custom components

These are components developed in the web application and referenced here by component identifier. This will dictate where the component is displayed in the component hierarchy and what actions are available for the component.

To add a custom component in the template config tree, we need to know its unique identifier and the data it should receive from the process model.

More details in the following section:

» **Custom**

The sections that can be configured are as follows:

1. **Message** - configure what data will be pushed to the frontend application
2. **Input keys** - used to define the process model paths from which the components will receive its data
3. **UI Actions** - actions defined here will be made available to the custom component. Multiple actions can be configured on a custom component and mapped to different triggers when developing it. Naming each action suggestively is important so the frontend engineer developing the component knows what actions should be triggered by certain events.

More information about configuration, [\[here\]](#)(using ui designer).

Displaying a UI element

When a process instance is started the web application will receive all the UI elements that can be displayed in that process.

When the process instance token will reach a User Task, a web socket message will be sent informing the SDK to display the UI element associated with that user task

Example:

1. Start a process: **POST**

```
{ {{processUrl}}/api/internal/process/DemoProcess/start }
```

INFO

The provided instruction involves initiating a process by making a **POST** request to the specified URL

`({{processUrl}}/api/internal/process/DemoProcess/start)`. This API call triggers the start of a process named "DemoProcess" by sending relevant data to the server.

```
{
  "processDefinitionName" : "DemoProcess",
  "tokens" : [ {
    "id" : 759224,
    "startNodeId" : null,
    "currentNodeId" : 662807,
    "currentnodeName" : null,
    "state" : "ACTIVE",
    "statusCurrentNode" : "ARRIVED",
    "dateUpdated" : "2023-05-31T09:44:39.969634Z",
    "uuid" : "d310996d-f3b9-44e5-983d-3631c844409e"
  } ],
  "state" : "STARTED",
  "templateConfig" : [ {
    "id" : 630831,
```

```
"flowxUuid" : "80ea0a85-2b0b-442a-a123-2480c7aa2dce",
"nodeDefinitionId" : 662856,
"componentIdentifier" : "CONTAINER",
"type" : "FLOWX",
"order" : 1,
"canGoBack" : true,
"displayOptions" : {
  "flowxProps" : { },
  "style" : null,
  "flexLayout" : {
    "fxLayoutGap" : 0,
    "fxLayoutAlign" : "start stretch",
    "fxLayout" : "column"
  },
  "className" : null,
  "platform" : "DEFAULT"
},
"templateConfig" : [ {
  "id" : 630832,
  "flowxUuid" : "38e2c164-f8cd-4f6e-93c8-39b7cdd734cf",
  "nodeDefinitionId" : 662856,
  "uiTemplateParentId" : 630831,
  "componentIdentifier" : "TEXT",
  "type" : "FLOWX",
  "order" : 0,
  "key" : "",
  "canGoBack" : true,
  "displayOptions" : {
    "flowxProps" : {
      "text" : "Demo text"
    },
    "style" : null,
    "flexLayout" : null,
    "className" : null,
    "platform" : "DEFAULT"
```

```
        },
        "expressions" : {
            "hide" : ""
        },
        "templateConfig" : [ ],
        "dataSource" : {
            "processData" : {
                "parentFlowxUuid" : null
            },
            "nomenclator" : {
                "parentFlowxUuid" : null
            }
        }
    }
},
"uuid" : "44177340-5ac6-4591-89ad-04df0815fb0",
"generalData" : null,
"backCounter" : 0,
"startedByActionId" : null,
"subProcesses" : null,
"subprocessesUuids" : null,
"baseUrl" : null
}
```

2. **ProgressUpdateDto** will trigger the **SDK** to search for the UI element having the same **nodeId** with the one in the SSE event.
3. Additionally, it will ask for data and actions that are required for this component via a **GET request**

```
 {{processUrl}}/api/process/db573705-71dd-4216-9d94-  
 5ba2fb36ff2a/data/42062
```

```
...
    "nodeDefinitionId" : 662856,
    "processDefinitionId" : 662952,
    "actionParams" : [ {
        "id" : 759458,
        "key" : "headers",
        "value" : "{$processInstanceId":${processInstanceId}}",
        "replaceValues" : true,
        "actionDefinitionId" : 759403
    }, {
        "id" : 759457,
        "key" : "customId",
        "value" : "folder",
        "replaceValues" : true,
        "actionDefinitionId" : 759403
    }, {
        "id" : 759456,
        "key" : "documentType",
        "value" : "document",
        "replaceValues" : false,
        "actionDefinitionId" : 759403
    }, {
        "id" : 759455,
        "key" : "topicName",
        "value" : "test.topic",
        "replaceValues" : false,
        "actionDefinitionId" : 759403
    } ],
    "actionRuleDefinitions" : [ ],
    "callbackActions" : null,
    "timerExpression" : "",
    "order" : 1,
    "manual" : false,
```

```
"repeatable" : false,  
"optional" : false,  
"autoRunChildren" : false,  
"allowTokenReset" : false,  
"restartFromSnapshot" : false,  
"keysForRestart" : [ ],  
"keys" : [ ]  
...
```

Values

For more details, please check the following page:

» [Message send receive task](#)

[Was this page helpful?](#)

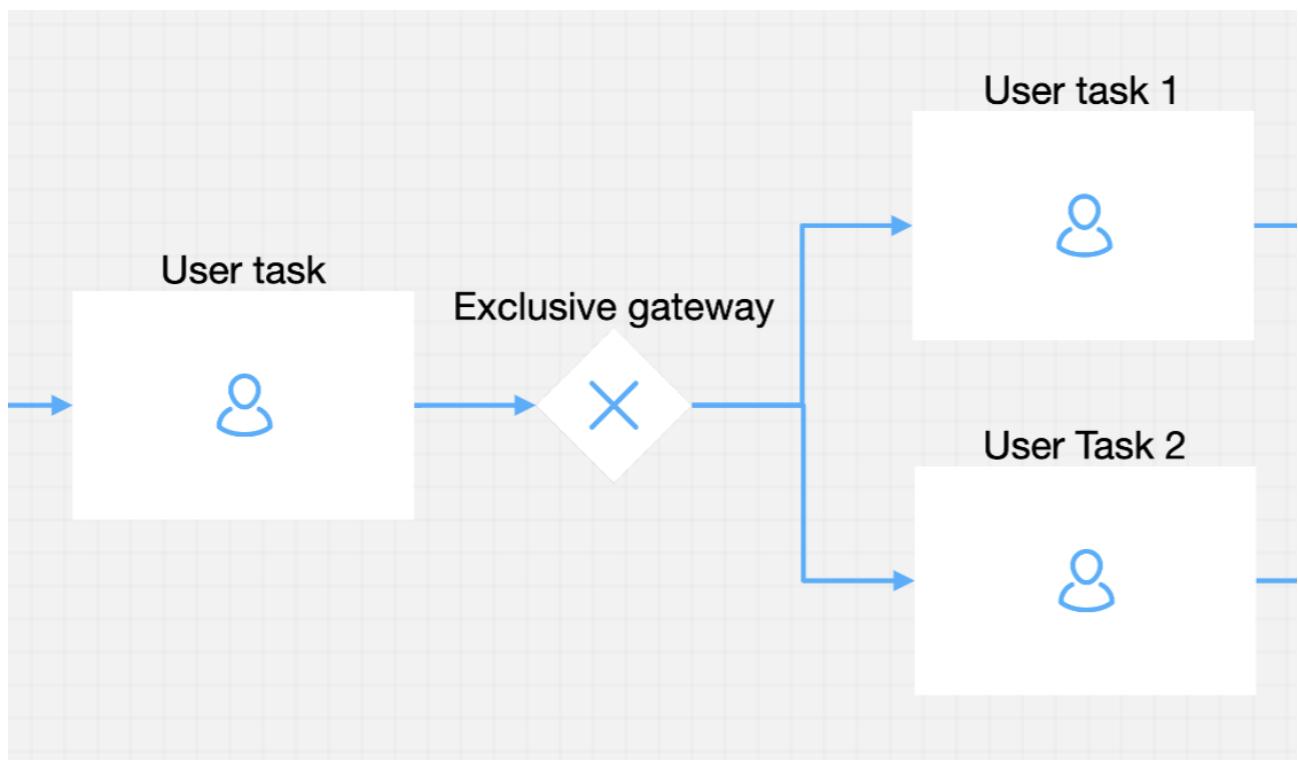
BUILDING BLOCKS / Node / Exclusive gateway

The fallback content to display on prerendering decisions can be configured using an Exclusive Gateway. Using this The fallback content to display on prerendering will make `if condition then go to this node` constructions are available.

Configuring an Exclusive gateway node



To configure this kind of node, it is useful to previously configure the **in** and **out** sequence from the gateway process.



General Config

- **Node name** - the name of the node
- **Can go back** - setting this to true will allow users to return to this step after completing it

INFO

When encountering a step with `canGoBack` switched to false, all steps found behind it will become unavailable.

- **Swimlane** - choose a swimlane (if there are multiple swimlanes on the process) to make sure only certain user roles have access only for certain process nodes- if there are no multiple swimlanes, the value is **Default**
- **Stage** - assign a stage to the node

Node: **12f5bcc0-13f3-44d8-b9c2-4e00c143fff4** (ID: 545174) ▼ X

Node Config

Gateway Decisions

General Config

Node name

exclusive gateway

Can go back?

Swimlane

Default

Stage

Save

Gateway Decisions

- **Language** - when configuring the condition, **MVEL** (or **DMN**) will be used and you should enter an expression that will be evaluated as **true** or **false**
- **Conditions** - selecting the **Gateway Decisions** tab of the gateway we can see that we can configure a list of conditions (**if, else if, else**) and **select** from a dropdown where we should go if the condition is **true**

🔥 DANGER

Expression order is important because the first **true** evaluation will stop the execution and the token will move to the selected node.

Node: **Exclusive gateway** (ID: 501853) ^

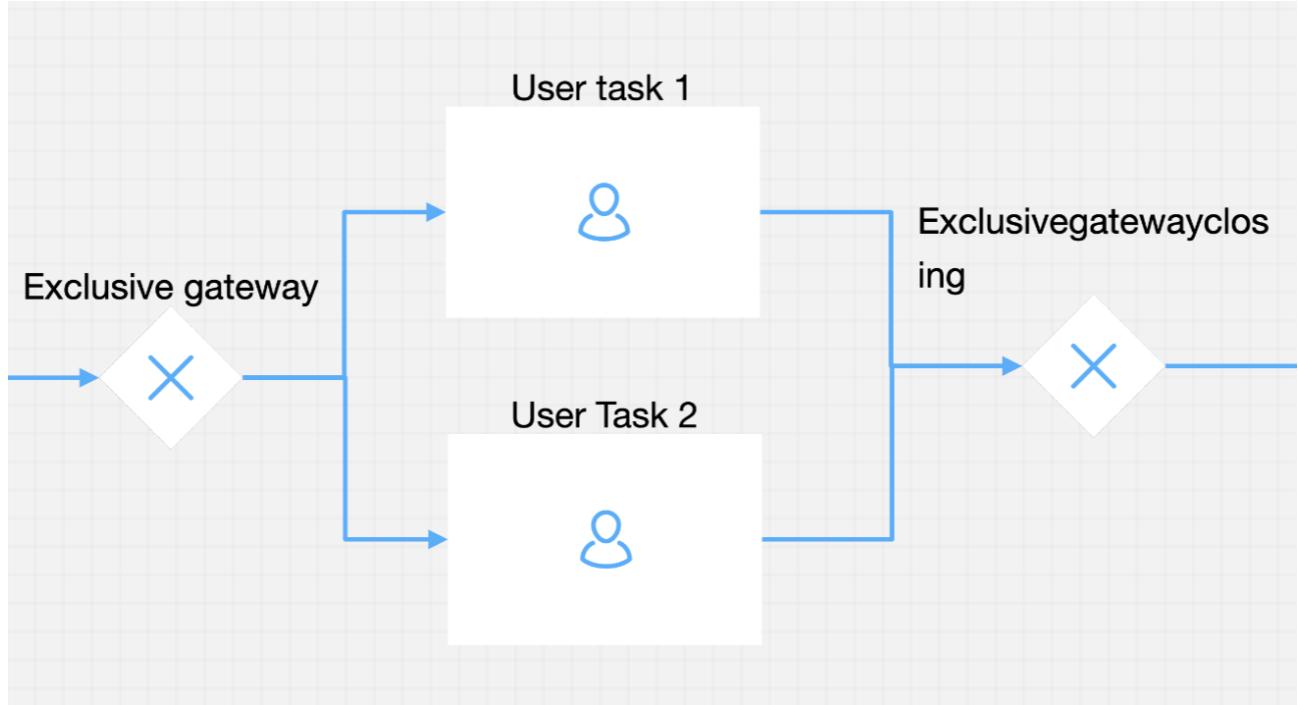
Node Config Gateway Decisions

Language: MVEL

if	↳ (input.get("application.client.age") < 18) then go to	User task 1	✖
else if	↳ (input.get("application.client.age") < 18) then go to	User task 1	✖
else if	↳ (Expression) then go to	⋮	Add

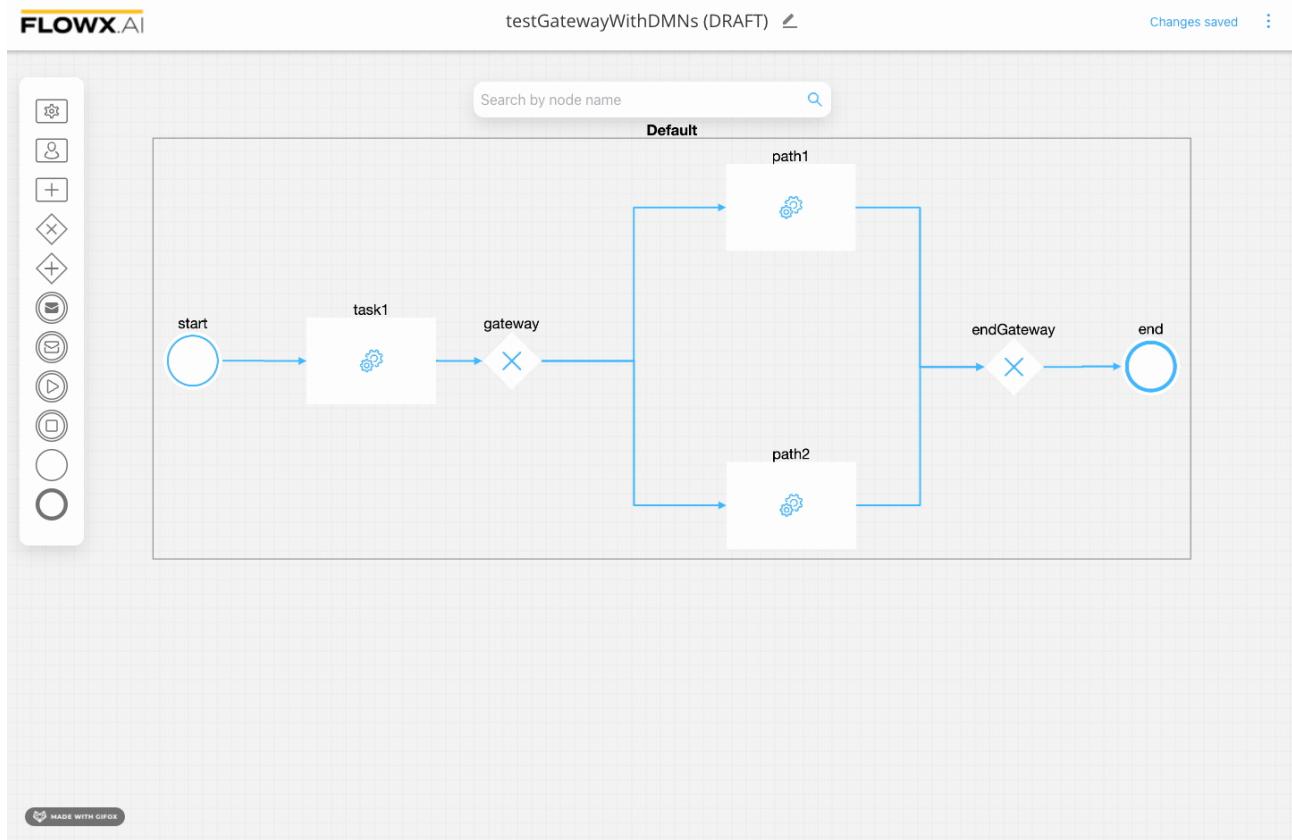
Save

After the exclusive part of the process (where a path or another will be used), you need to end each path or join back to a single process using a new exclusive gateway without any configuration on it.



Configuring a DMN Exclusive Gateway node

You can use **DMN** to define gateway decisions, using exclusive gateways.



Gateway Decision - DMN example (applicable only for exclusive gateway - XOR)

Node: **gateway** (ID: 367901)

[Node Config](#) [Gateway Decisions](#)

[Language](#)

Language

DMN

Rules

[View DRD](#)

		Hit Policy: First			
	When	Input	Then	Next Node Name	Annotations
		boolean		string	
1	true			path1	
2	false			path2	
+	-				

[Was this page helpful?](#)

BUILDING BLOCKS / Node / Parallel gateway

If multiple operations can be done in parallel a Parallel Gateway can be used. This kind of node will open a parallel section of the

The fallback content to display on prerendering , very useful for integrations that can be done in parallel, without waiting for each other. Each parallel section should be also closed by another parallel Gateway node.

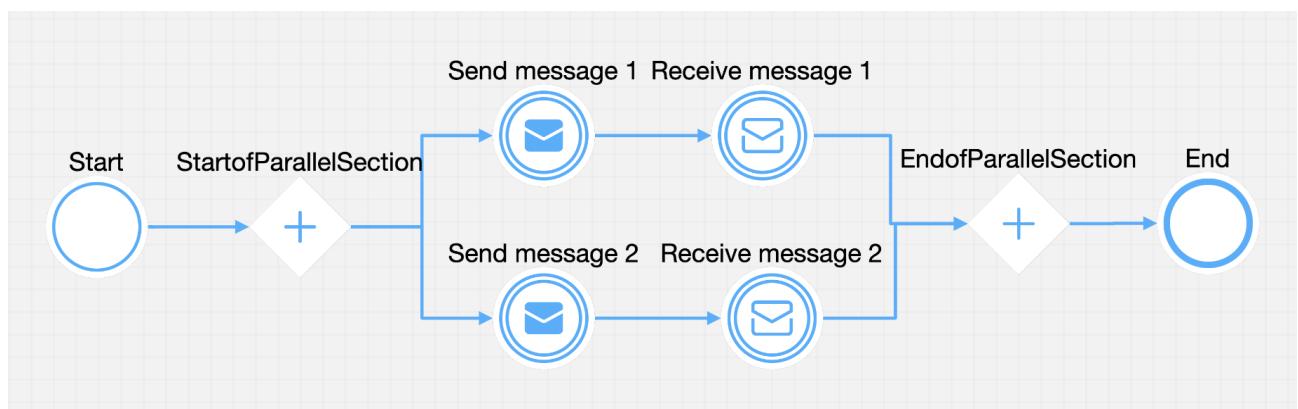
Configuring a Parallel gateway node



This kind of

The fallback content to display on prerendering

has no special configuration and can start 2 or more parallel paths. It is important to keep in mind that the close Parallel node, required to close the parallel section will wait for all branches to finish before moving to next node.



Was this page helpful?

BUILDING BLOCKS / Node / Milestone node

A **milestone node** is used to define how **user tasks** (which are placed between two milestones - **start milestone** and **end milestone**) will be displayed.



Multiple options are available for displaying the content:

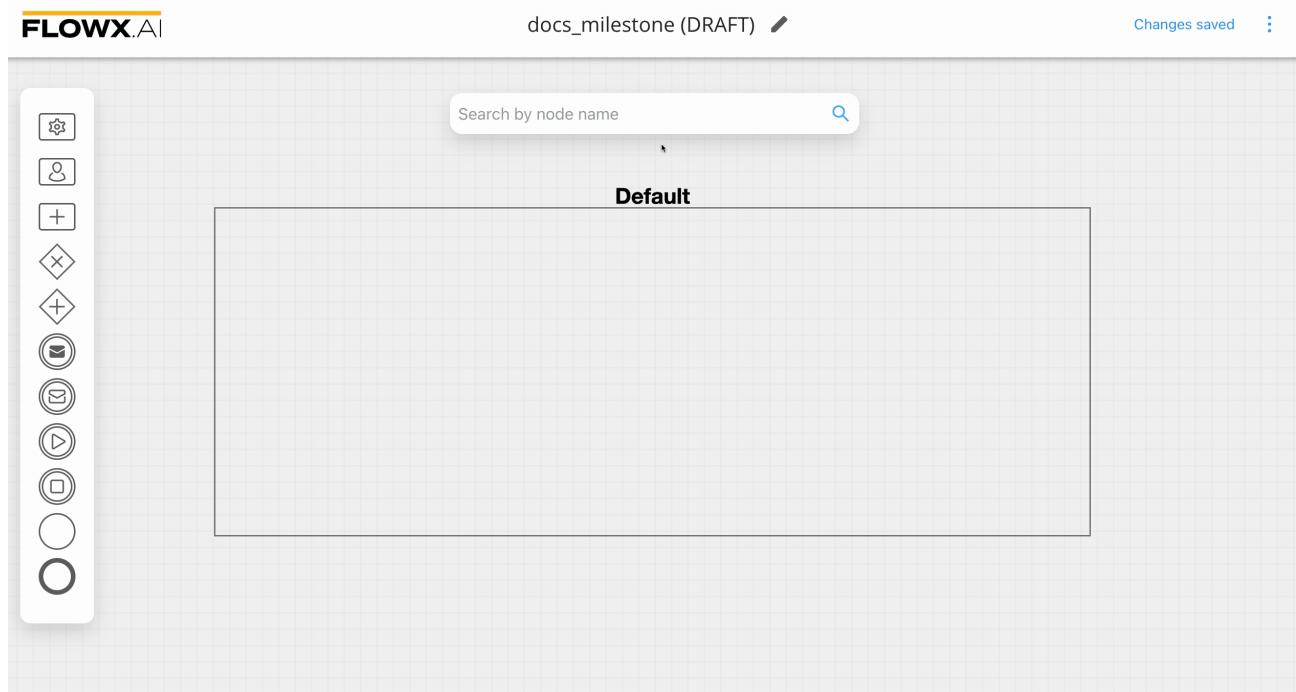
- **Modal**
- **Page**
- **Stepper + Steps**
- **Container**

Configuring a Milestone node

A combination of **start** and **end** nodes can be used to achieve all kinds of a grouping of multiple user task nodes.

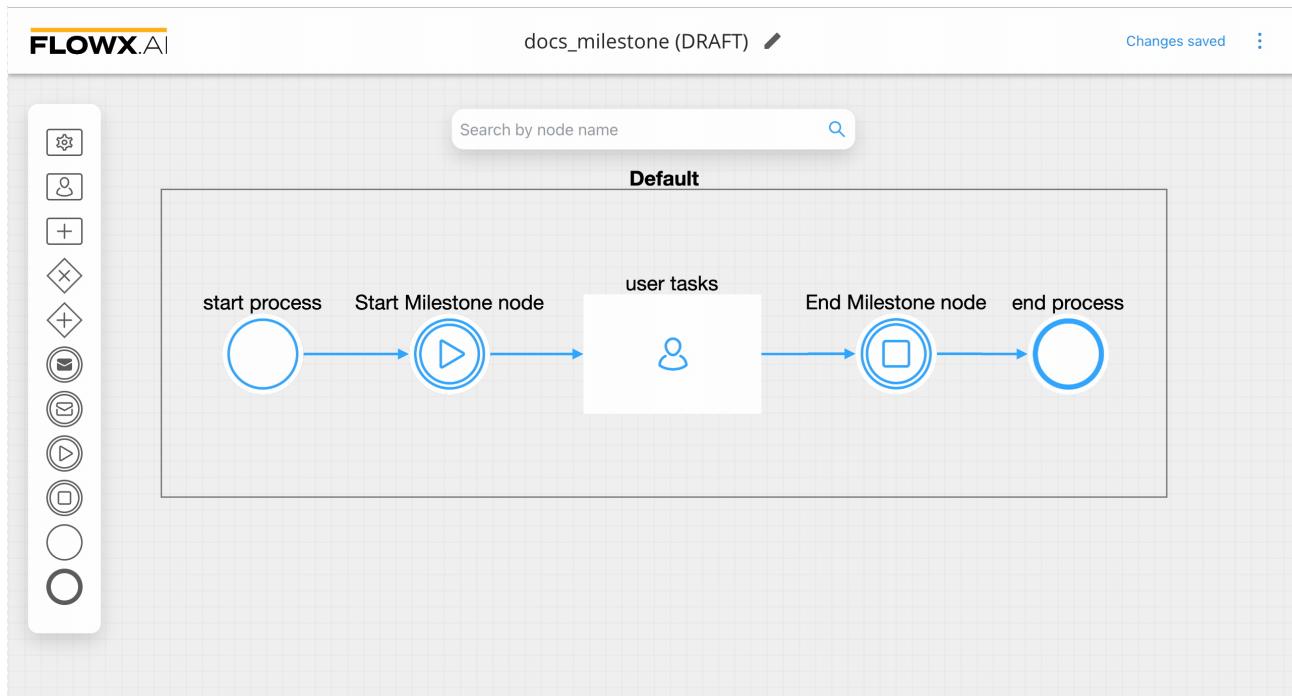
To configure a user task to be displayed in a Modal:

1. Open **Process Designer** and start configuring a process.
2. Add a **user task** that you want to display.
3. Add a **start milestone** before the user task.
4. Add an **end milestone** after the user task.



5. Select the **start milestone node** and open **UI Designer** - here you can choose from multiple templates of how to display the content.
6. For example, drag and drop the **modal** template to the canvas.

- No additional information is required for displaying a **user task** in a modal view but you can do multiple customizations via the different configurations using the UI Designer.



Available Components

Modal

You can configure a start milestone node and an end milestone node before and after a **user task**. After adding the milestones, you can add a modal template to the start milestone node to display a modal screen (like in the example above).

The screenshot shows the FLOWX.AI application 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
- Plugins**
 - Task Manager
 - All tasks

The main content area displays a page titled "This is a PAGE." with a "Continue" button. Below this, a modal dialog is open, containing the following text and buttons:

This is the 1st USER TASK in a MODAL.

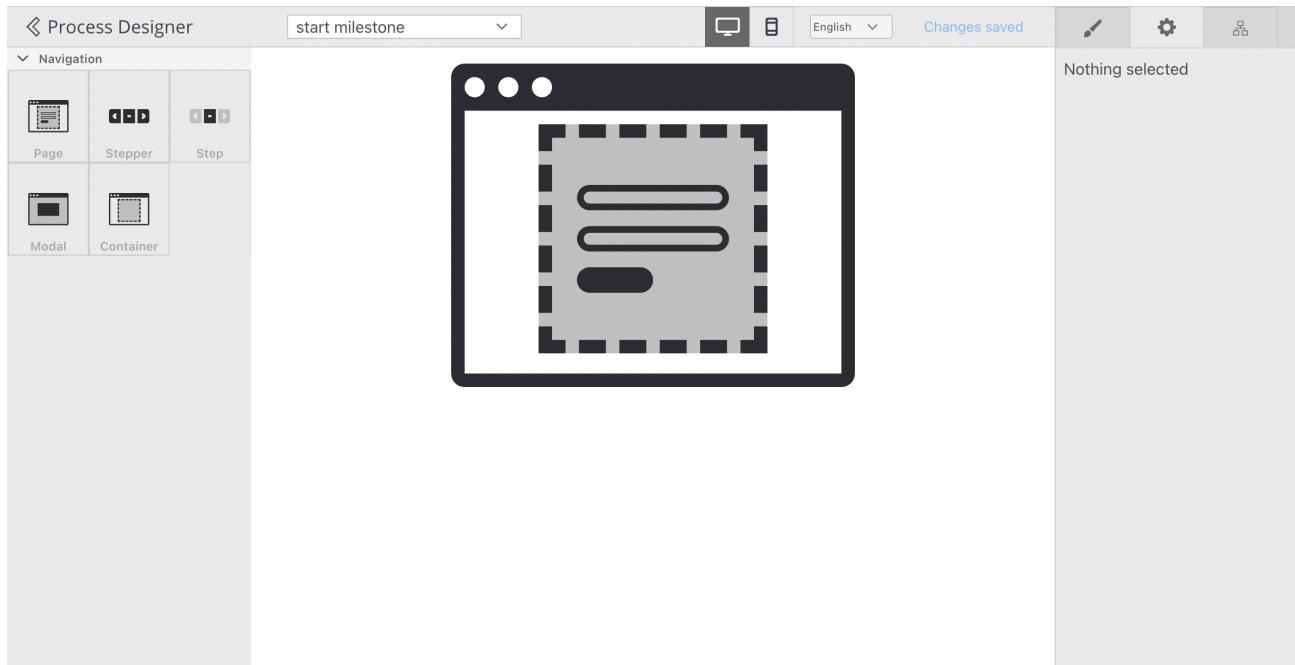
FormGroupTitle

This is the 2nd USER TASK in the same MODAL.

Next

Page

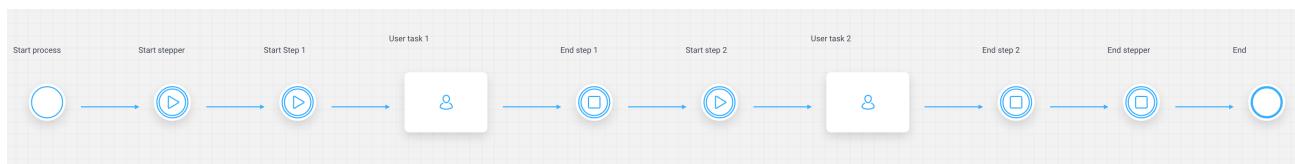
A basic full page content can be displayed using this kind of template on a milestone start.



Stepper + Steps

To create a stepper architecture:

1. First define a **milestone start node**.
2. Then add a **Stepper template** on the first node (and a **milestone end** after the **first node**).
3. In between the **stepper milestones** add for each **step** a **milestone start** and a **milestone end** node with a **Step configuration**.

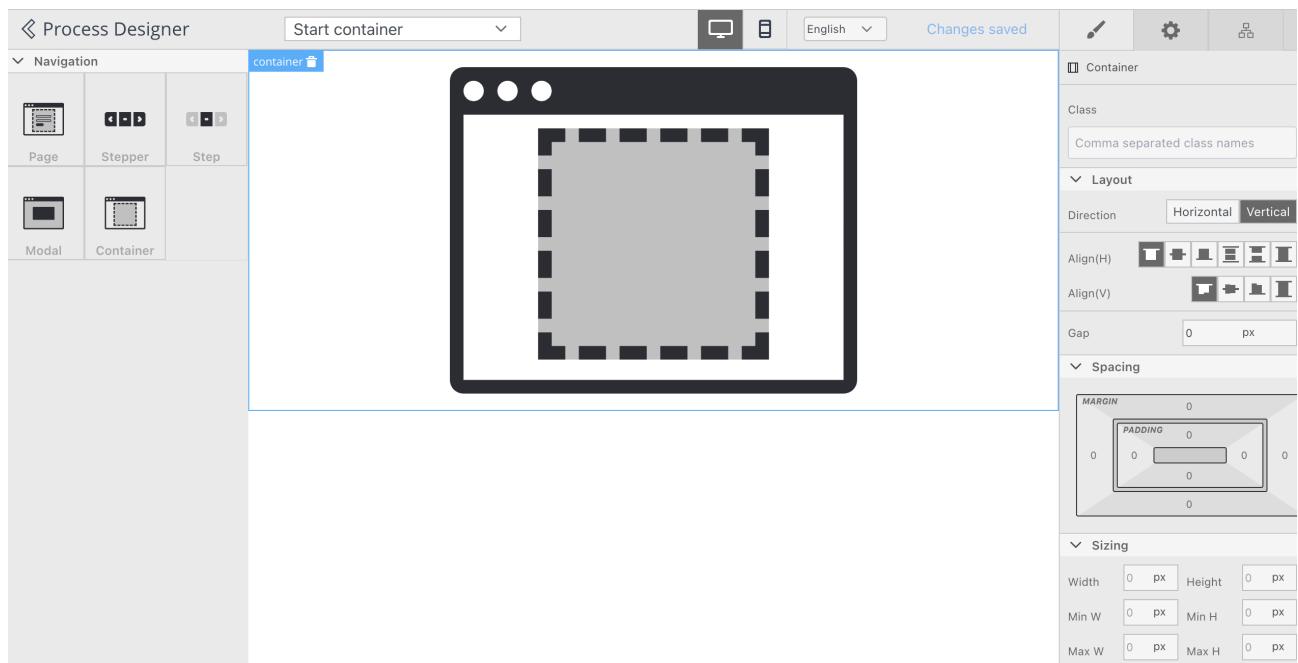


For more information about how to create a process with a Stepper + Steps structure, and how to configure the UI, check the following section:

» Create a User Interface

Container

Containers allows us to display multiple user task on the same Page/Modal/Step with a different layout, other than the basic one. You can use **Layout** tab to play with multiple alignments.



Was this page helpful?

BUILDING BLOCKS / Node / Subprocess run node

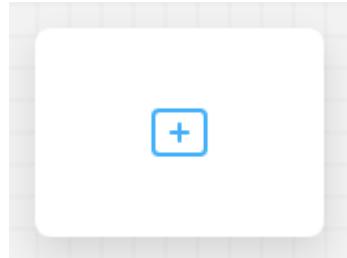
!(INFO)

There might be cases when extra functionality is needed on certain

The fallback content to display on prerendering

A node that provides advanced options for starting

The fallback content to display on prerendering



» Subprocess

It contains a default action for starting a subprocess.

A subprocess can be started in two modes:

- **async mode** - the parent

The fallback content to display on prerendering

will continue without waiting for the sub-process to finish

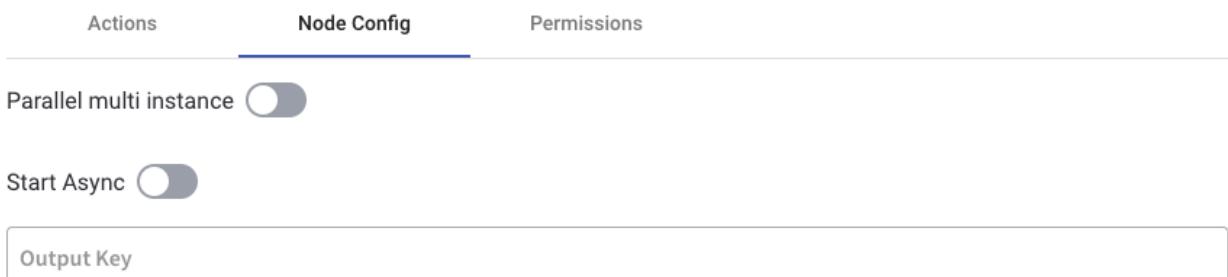
!(INFO)

Select if this task should be invoked asynchronously. Make tasks asynchronous if they cannot be executed instantaneously, for example, a task performed by an outside service.

- **sync mode** - the parent process must wait for the subprocess to finish before advancing

The start mode can be chosen when configuring the subprocess run node.

In case the parent process needs to wait for the subprocess to finish and retrieve some results from it, the parent process key that will hold the results must be defined using the *output* key node config value.



This node type can also be used for starting a set of subprocesses that will be started and run at the same time. This will prove useful in case we have an array of values in the parent process parameters and we want to start a subprocess for each of the elements in that array.



In order to do this, we need to select the parallel multi instance option. The *collection key* name from the parent process also needs to be specified.

!(INFO)

When designing such a subprocess that will be started in a loop, you need to keep in mind that the input value for the subprocess (that is, one of the values from the array in the parent process) will be stored in the subprocess parameter values under the key named *item*. This will have to be used inside the subprocess. If this subprocess produces any results, they should be stored under a key named *result* in order to be sent back to the parent process.

Was this page helpful?

BUILDING BLOCKS / Node / Message events / Message Throw Intermediate

Event

(!) QUICK INTRO

What is it? It's like throwing a message to tell someone about something. After throwing the message, the process keeps going, and other parts of the process can listen to that message.

Why it is important? The Message Throw Intermediate Event is important because it allows different parts of a process to communicate and share information with each other.

Configuring a Message Throw Intermediate Event

A Message Throw Intermediate Event is an event in a process where a message is sent to trigger a communication or action with another part of the process (can be correlated with a catch event). It represents the act of throwing a message to initiate a specific task or notification. The event creates a connection between the sending and receiving components, allowing information or instructions to be transmitted. Once the message is thrown, the process continues its flow while expecting a response or further actions from the receiving component.



General config

- **Can go back?** - setting this to true will allow users to return to this step after completing it, when encountering a step with `canGoBack` false, all steps found behind it will become unavailable
- **Correlate with catch events** - the dropdown contains all catch messages from the process definitions accessible to the user
- **Correlation key** - is a process key that uniquely identifies the instance to which the message is sent
- **The data field** - allows the user to define a JSON structure with the data to be sent along with the message
- **Stage** - assign a stage to the node

Node: Application decision (ID: 2307723) ▼ ▲

Node Config

General Config

Can go back?

Correlate with catch events

branch1



Correlation Key

app.id

```
1 {"approved": "${app.approved}"}
```

Stage

Onboarding



Save

Was this page helpful?

BUILDING BLOCKS / Node / Message events / Message Catch Boundary Events

(!) QUICK INTRO

What is it? A Message Catch Boundary Event is a special

The fallback content to display on prerendering

that can be attached to a user task in a

The fallback content to display on prerendering

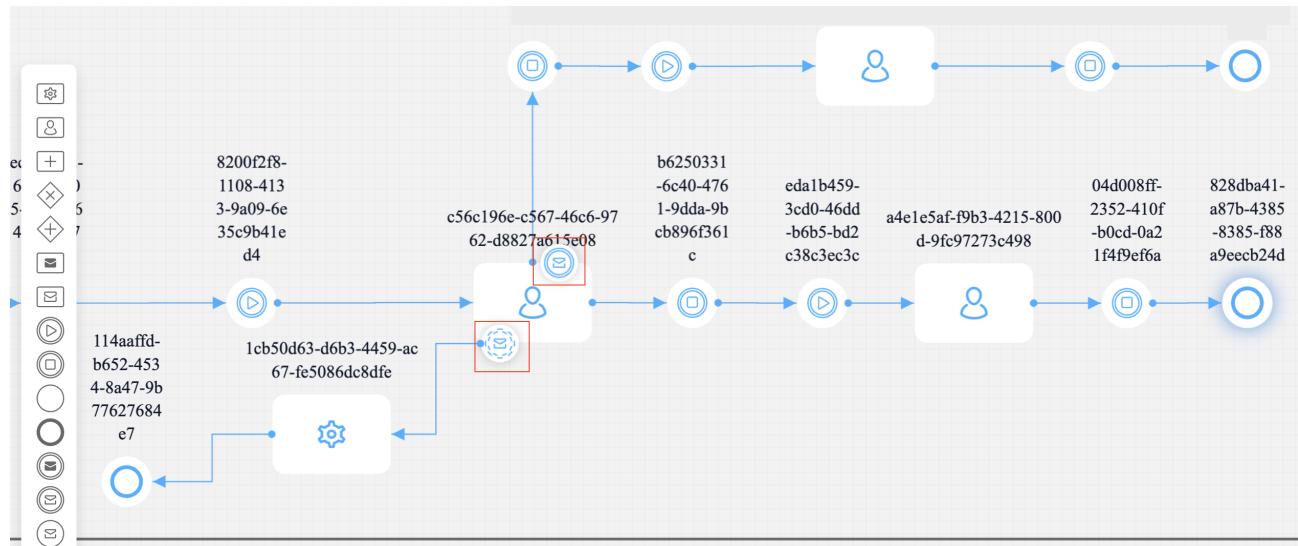
Why it is important? It allows the process to listen for and capture specific messages during the execution of the associated user task.

When used as a boundary event on a **user task**, message catch boundary event nodes behave similar to an **exclusive gateway**, but they are activated upon receiving an event. This means you can proceed in the process without receiving an event and continue through the sequence initiated from the user task.

If an event is received, it advances through the sequence from the intermediate

The fallback content to display on prerendering

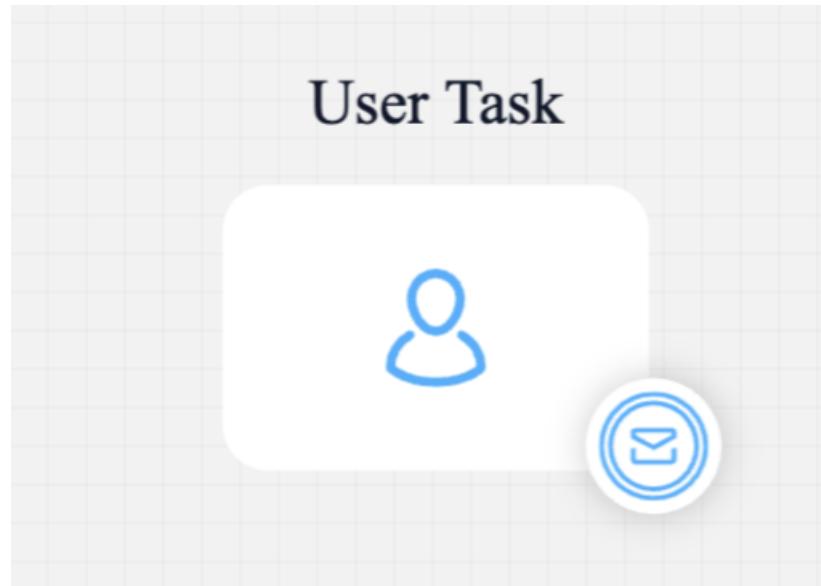
. You can have multiple intermediate boundary events on the same user task, but only one can be activated at a time.



There are two types of Message Catch Boundary Events:

- **Interrupting**
- **Non-Interrupting**

Message Catch Interrupting Event



When an Interrupting Message Catch Boundary Event is triggered by receiving a message, it interrupts the associated task that is being performed. The task is immediately finished, and the

The fallback content to display on prerendering continues to advance based on the received message.

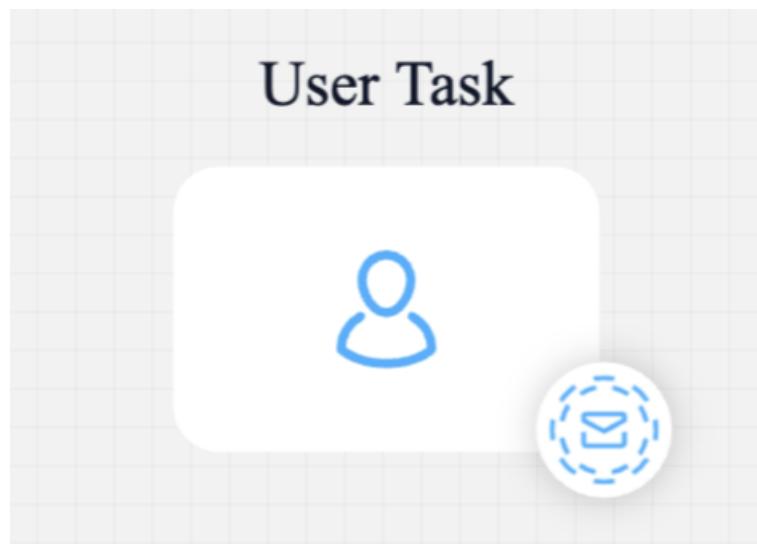
It can also be used as a standalone node, see more information on the following section:

» [Message Catch Intermediate Event](#)

Message Catch Non-Interrupting Event

It is used only as a boundary event and is placed only on a user task. If your process is in that user task and receives

The fallback content to display on prerendering , the event is activated, and a new token is created that advances independently. Sections with non-interrupting events should not contain user tasks. You can have multiple non-interrupting events on the same user task, and all of them can be activated simultaneously.



A Non-Interrupting Message Catch Boundary Event also listens for messages while the associated task is being performed. However, in this case, the task is not immediately finished when messages are received. The event captures the messages, allowing the task to continue its execution. Multiple non-interrupting events can be received while the task is still active, and the task will continue until its completion.

Configuring a Message Catch Interrupting/Non-Interrupting Event

General config

- **Correlate with throwing events** - the dropdown contains all throw events from the process definitions accessible to the user

(!) INFO

It is used to establish the correlation between the catch event and the corresponding throw event. By selecting the appropriate throw event, the catch event will be triggered when a message is thrown from that event.

- **Correlation key** - process key used to establish a correlation between the received message and a specific process instance

(!) INFO

Correlation key serves as a means to correlate the incoming message with the specific process instance it belongs to. When a message is received with a matching correlation key, the catch event will be triggered.

- **Receive data (process key)** - the catch event can receive data associated with the message and store it in a process variable with the specified process key

(!) INFO

This data can then be used within the process instance for further processing or decision-making.

[Was this page helpful?](#)

BUILDING BLOCKS / Node / Message events / Message Catch Intermediate Event

(!) QUICK INTRO

What is it? A Message Catch Intermediate Event is a type of event in a process that waits for a specific message before continuing with the process flow.

Why it is important? It enables the process to synchronize and control the flow based on the arrival of specific messages, ensuring proper coordination between process instances.

Similar to the Message Catch Boundary Event, the Message Catch Intermediate Event is important because it facilitates the communication and coordination between process instances through messages. By incorporating this event, the process can effectively synchronize and control the flow based on the arrival of specific messages.

(!) INFO

Message Catch Intermediate Event can be used as a standalone node, this means that it will block a process until it receives an event.

Configuring a Message Catch Intermediate Event

Imagine a process where multiple tasks are executed in sequence, but the execution of a particular task depends on the arrival of a certain message. By incorporating a Message Catch Intermediate Event after the preceding task, the process will pause until the expected message is received. This ensures that the subsequent task is not executed prematurely and allows for the synchronization of events within the process.



General config

- **Can go back?** - setting this to true will allow users to return to this step after completing it, when encountering a step with `canGoBack` false, all steps found behind it will become unavailable
- **Correlate with throwing events** - the dropdown contains all catch messages from the process definitions accessible to the user
- **Correlation key** - process key used to establish a correlation between the received message and a specific process instance
- **Receive data** - the process key that will be used to store the data received along with the message
- **Stage** - assign a stage to the node

Node: **ee61fcc9-aa51-4b8a-a4f1-ee2df56a23d6** (ID: 2313991)

Node Config

General Config

Can go back?

Correlate with throwing events

same_process



Correlation Key

app.id

Receive data

Process Key

test

Stage



Was this page helpful?

BUILDING BLOCKS / Node / Message events / Message Catch Start Event

(!) QUICK INTRO

What is it? It represents the starting point for a process instance based on the receipt of a specific message. When this event is triggered by receiving the designated message, it initiates the execution of the associated process.

Why it is important? The Message Catch Start Event is important because it allows a process to be triggered and initiated based on the reception of a specific message.

Configuring a Message Catch Start Event

A Message Catch Start Event is a special event in a process that initiates the start of a process instance upon receiving a specific message. It acts as the trigger for the process, waiting for the designated message to arrive. Once the message is received, the process instance is created and begins its execution, following the defined process flow from that point onwards. The Message Catch Start Event serves as the entry point for the process, enabling it to start based on the occurrence of the expected message.

⚠ CAUTION

It is mandatory that in order to use this type of node together with task management plugin, to have a service account defined in your identity

solution. For more information, check our documentation in how to create service accounts using Keycloak, [here](#)



General config

- **Can go back?** - setting this to true will allow users to return to this step after completing it, when encountering a step with `canGoBack` false, all steps found behind it will become unavailable
- **Correlate with catch events** - the dropdown contains all catch messages from the process definitions accessible to the user
- **Correlation key** - is a process key that uniquely identifies the instance to which the message is sent
- **The data field** - allows the user to define a JSON structure with the data to be sent along with the message
- **Stage** - assign a stage to the node

Node: **cc20eb60-c74f-4e37-9a09-d3e1154582f1** (ID: 2309403)

Node Config

General Config

Can go back?

Correlate with throwing events

new_event



Correlation Key

qwe

Receive data

Process Key

receivedData

Stage



Was this page helpful?

BUILDING BLOCKS / Actions / Business Rule action / DMN Business

Rule action

For a brief introduction to

The fallback content to display on prerendering
, check the following section:

» [Intro to DMN](#)

Creating a DMN Business Rule action

To create and attach a DMN

The fallback content to display on prerendering
action to a task

The fallback content to display on prerendering
, you must do the following:

1. Open

The fallback content to display on prerendering
and go to

The fallback content to display on prerendering

2. Select your process from the list and click **Edit process**.

3. Select a **task node** then click the **edit button** (the key icon) - this will open the node configuration menu.

4. In the opened menu, go to the

The fallback content to display on prerendering

tab then click the "+" button.

5. From the dropdown menu choose the action type - **Business Rule**.

6. In the **Language** dropdown menu, select **DMN**.



Using a DMN Business Rule action

We have the following scenario, a bank needs to perform client identification tasks/actions. This action can be defined as a

The fallback content to display on prerendering
inside a

The fallback content to display on prerendering
process using

The fallback content to display on prerendering

A business person or specialist can use DMN to design this business rule, without having to go deep into technical definitions.

Here is an example of an **MVEL** script - defined as a business rule action inside a **Service Task** node:

```
closedClientType = ["PF_CLOSED", "PF_SPECIAL", "PF_ABC",
"PJ_CLOSED"];
clientType =
input.get("application").get("client").get("clientType");
if (closedClientType.contains(clientType)) {
    alertTitle = "Customer no longer with the bank";
    alertDescription = "Hey! This person was a client
before. For a new account modify the CIF.";
    output.put("applications", {"client": {"alertTitle": alertTitle,
"alertDescription": alertDescription}});
}
```

The previous example could be easily transformed into a DMN Business Rule action - represented by the decision table:

Decision table		Hit Policy: Unique		
	When	And		Annotations
	application.client.clientType string	alert_title string	alert_description string	
1	IN ("PF_CLOSED", "PF_SPECIAL", "PF_ABC", "PJ_CLOSED")	Customer no longer with the bank.	Hey! This person was a client before. For a new account, modify the CIF	
2	-			
+	-			

In the example above we used FEEL expression language in order to write the rules that should be met in order for the output to happen. FEEL defines a syntax

for expressing conditions that input data should be evaluated against.

Input - In the example above we used as inputs the type of clients (inside a bank) using the `application.client` key

Output - In the example above we used as inputs the type of clients (inside a bank) using the `application.client` key

DMN also defines an XML schema that allows DMN models to be used across multiple DMN authoring platforms. The following output is the XML source of the decision table example from the previous section:

```
// Decision Table XML source
<?xml version="1.0" encoding="UTF-8"?>
<definitions
  xmlns="https://www.omg.org/spec/DMN/20191111/MODEL/"
  xmlns:biodi="http://bpmn.io/schema/dmn/biodi/2.0"
  id="Definitions_06nober" name="DRD"
  namespace="http://camunda.org/schema/1.0/dmn"
  exporter="Camunda Modeler" exporterVersion="5.0.0">
  <decision id="closed_CIF" name="Decision table">
    <decisionTable id="decisionTable_1">
      <input id="input_1"
        label="application.client.clientType" biodi:width="277">
        <inputExpression id="inputExpression_1"
          typeRef="string">
          <text></text>
        </inputExpression>
      </input>
      <output id="output_1" label="alert_title"
        typeRef="string" />
        <output id="OutputClause_043h9fw"
          label="alert_description" typeRef="string" />
```

```
<rule id="DecisionRule_10bh1zx">
    <inputEntry id="UnaryTests_0a6rf6l">
        <text>IN ("PF_CLOSED", "PF_SPECIAL", "PF_ABC",
"PJ_CLOSED")</text>
    </inputEntry>
    <outputEntry id="LiteralExpression_0xszo8x">
        <text>Customer no longer with the bank.</text>
    </outputEntry>
    <outputEntry id="LiteralExpression_0l2bioo">
        <text>Hey! This person was a client before. For a
new account, modify the CIF</text>
    </outputEntry>
</rule>
<rule id="DecisionRule_1jj1rv2">
    <inputEntry id="UnaryTests_0cf2e91">
        <text></text>
    </inputEntry>
    <outputEntry id="LiteralExpression_1b9jkr4">
        <text></text>
    </outputEntry>
    <outputEntry id="LiteralExpression_12hua2f">
        <text></text>
    </outputEntry>
</rule>
</decisionTable>
</decision>
</definitions>
```

You can use this XML example with FLOWX Designer, adding it to a Business Rule Action - using an MVEL script. Then you can switch to DMN if you need to generate a graphical representation of the model.

Was this page helpful?

BUILDING BLOCKS / Actions / Send data to user interface

! INFO

What is it? Send data to user interface

The fallback content to display on prerendering is based on Server-Sent Events (SSE), a web technology that enables servers to push real-time updates or events to clients over a single, long-lived HTTP connection. It provides a unidirectional communication channel from the server to the client, allowing the server to send updates to the client without the need for the client to continuously make requests.

Why is it useful? It provides real-time updates and communication between the

The fallback content to display on prerendering and the frontend application.

Configuring a Send data to user interface action

Multiple options are available for this type of action and can be configured via the

The fallback content to display on prerendering

- . To configure a Send data to user interface, use the **Actions** tab at the **task node level**, which has the following configuration options:

- Action Edit
- Back in steps (for Manual actions)
- Parameters
- Data to send (for Manual actions)

Action Edit

- **Name** - used internally to make a distinction between different actions on nodes in the process. We recommend defining an action naming standard to be able to quickly find the process actions
- **Order** - if multiple actions are defined on the same node, the running order should be set using this option
- **Timer expression** - it can be used if a delay is required on that action. The format used for this is ISO 8601 duration format (for example, a delay of 30 seconds will be set up as PT30S)
- **Action type** - should be set to Send data to user interface
- **Trigger type** (options are Automatic/Manual) - choose if this action should be triggered automatically (when the process flow reaches this step) or manually (triggered by the user); in most use cases, this will be set to automatic
- **Required type** (options are Mandatory/Optional) - automatic actions can only be defined as mandatory. Manual actions can be defined as mandatory or optional.
- **Repeatable** - 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

Back in steps

- **Allow BACK on this action** - back in process is a functionality that allows you to go back in a business process and redo a series of previous actions in the process. For more details, check [Moving a token backwards in a process](#) section.

Action Edit

ID: 540185
Name

db77e7be-30fb-4357-8fed-23d074ede1db
Order

1

Timer Expression

Websocket Send Action

▼

Automatic Manual

Mandatory Optional

Repeatable

Autorun Children?

Save

Parameters

The following fields are required for a minimum configuration of this type of action:

- **Message Type** - if you only want to send data, you can set this to **Default** (it defaults to the **data** message type)

DANGER

If you need to start a new process using a **Send data to user interface**, you can do that by setting the **Message Type** to **Action** and you will need to define a **Message** with the following format:

```
{  
  "processName": "demoProcess",  
  "type": "START_PROCESS_INHERIT",  
  "clientDataKeys": ["webAppKeys"],  
  "params": {  
    "startCondition": "${startCondition}",  
    "paramsToCopy": []  
  }  
}
```

(!) INFO

- **paramsToCopy** - choose which of the keys from the parent process parameters to be copied to the subprocess
- **withoutParams** - choose which of the keys from the parent process parameters are to be ignored when copying parameter values from the parent process to the subprocess

- **Message** - here you define the data to be sent as a JSON object, you can use constant values and values from the process instance data.
- **Target Process** - is used to specify to what running process instance should this message be sent - **Active process** or **Parent process**

(!) INFO

If you are defining this action on a **subprocess**, you can send the message to the parent process using **Target Process: Parent process**.

Data to send

- **Keys** - are 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)



Data to send option is configurable only when the action **trigger type** is **Manual**.

Parameters

Message Type

Message

```
1  {
2    "processName": "demoProcess",
3    "type": "START_PROCESS_INHERIT",
4    "clientDataKeys": ["webAppKeys"],
5    "params": {
6      "startCondition": "${startCondition}",
7      "paramsToCopy": []
8    }
9 }
```

Send Update Data example

To send the latest value from the **process instance** data found at `application.client.firstName` key, to the frontend app, you can do the following:

1. Add a **Send data to user interface**.
2. Set the **Message Type** to **Default** (this is default value for `data`).
3. Add a **Message** with the data you want to send:
 - `{ "name": "${application.client.firstName}" }`
4. Choose the **Target Process**.

Node: Task node (ID: 542401) ▼ X

Node Config Actions

▼ Actions + |

{ "name": "\${application.client.firstName}" }

Was this page helpful?

BUILDING BLOCKS / Actions / Upload File action

!(INFO)

What is it? An **Upload File action** is an

The fallback content to display on prerendering
type that allows you to upload a file to a service available on
The fallback content to display on prerendering

Why is it useful? The action will receive a file from the frontend and send it to Kafka, and will also attach some metadata.

Configuring an Upload File action

Multiple options are available for this type of action and can be configured via the

The fallback content to display on prerendering

. To configure an Upload File action, use the **Actions** tab at the **task node level**, which has the following configuration options:

- Action Edit
- Back in steps (for Manual actions)
- Parameters
- Data to send (for Manual actions)

Action Edit

- **Name** - used internally to make a distinction between different actions on nodes in the process. We recommend defining an action naming standard to be able to quickly find the process actions
- **Order** - if multiple actions are defined on the same node, the running order should be set using this option

- **Timer expression** - it can be used if a delay is required on that action. The format used for this is **ISO 8601 duration format** (for example, a delay of 30 seconds will be set up as `PT30S`)
- **Action type** - should be set to **Upload File**
- **Trigger type** (options are Automatic/Manual) - choose if this action should be triggered automatically (when the process flow reaches this step) or manually (triggered by the user); in most use cases, this will be set to automatic
- **Required type** (options are Mandatory/Optional) - automatic actions can only be defined as mandatory. Manual actions can be defined as mandatory or optional.
- **Repeatable** - 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

Action Edit

ID: 540672

Name

Send Update Data

Order

1

▼

Timer Expression

Upload File

▼

Automatic Manual

Mandatory Optional

Repeatable

Autorun Children?

Allow BACK on this action?

Save

Back in steps

- **Allow BACK on this action** - back in process is a functionality that allows you to go back in a business process and redo a series of previous actions in the process. For more details, check [Moving a token backwards in a process](#) section.

Parameters

- **Address** - the Kafka topic where the file will be posted
- **Document Type** - other metadata that can be set (useful for the [document plugin](#))
- **Folder** - allows you to configure a value by which the file will be identified in the future
- **Advanced configuration (Show headers)** - this represents a JSON value that will be sent on the headers of the Kafka message

Data to send

- **Keys** - are 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)

DANGER

Data to send option is configurable only when the action **trigger type** is **Manual**.

Example

An example of **Upload File Action** is to send a file to the **document plugin**. In this case, the configuration will look like this:

Parameters configuration

- **Address (topicName)** - will be set to (the id of the document plugin service)
`ai.flowx.in.document.persist.v1`
- **Document Type** - metadata used by the document plugin, here we will set it to **BULK**
- **Folder** - the value by which we want to identify this file in the future (here we use the **client.id** value available on the process instance data):
 `${application.client.id}`

Advanced configuration

- **Headers** - headers will send extra metadata to this topic -

```
{"processInstanceId": ${processInstanceId}, "destinationId":  
"currentNodeName"})
```

Parameters

Address

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

 Replace Values

Document Type

```
BULK
```

 Replace Values

Folder

```
 ${application.client.id}
```

 Replace Values

Advanced configuration

Show Headers

```
1 {"processInstanceId": ${processInstanceId}, "destinationId": "currentNodeName"}
```

 Replace Values

Data to send

Was this page helpful?

BUILDING BLOCKS / Actions / Start Subprocess action

ⓘ INFO

What is it? A **Start Subprocess action** is an

The fallback content to display on prerendering
that allows you to start a

The fallback content to display on prerendering
from another (parent)

The fallback content to display on prerendering

Why is it important? Using **subprocesses** is a good way to split the complexity of your business flow into multiple, simple and reusable processes.

Configuring a Start Subprocess action

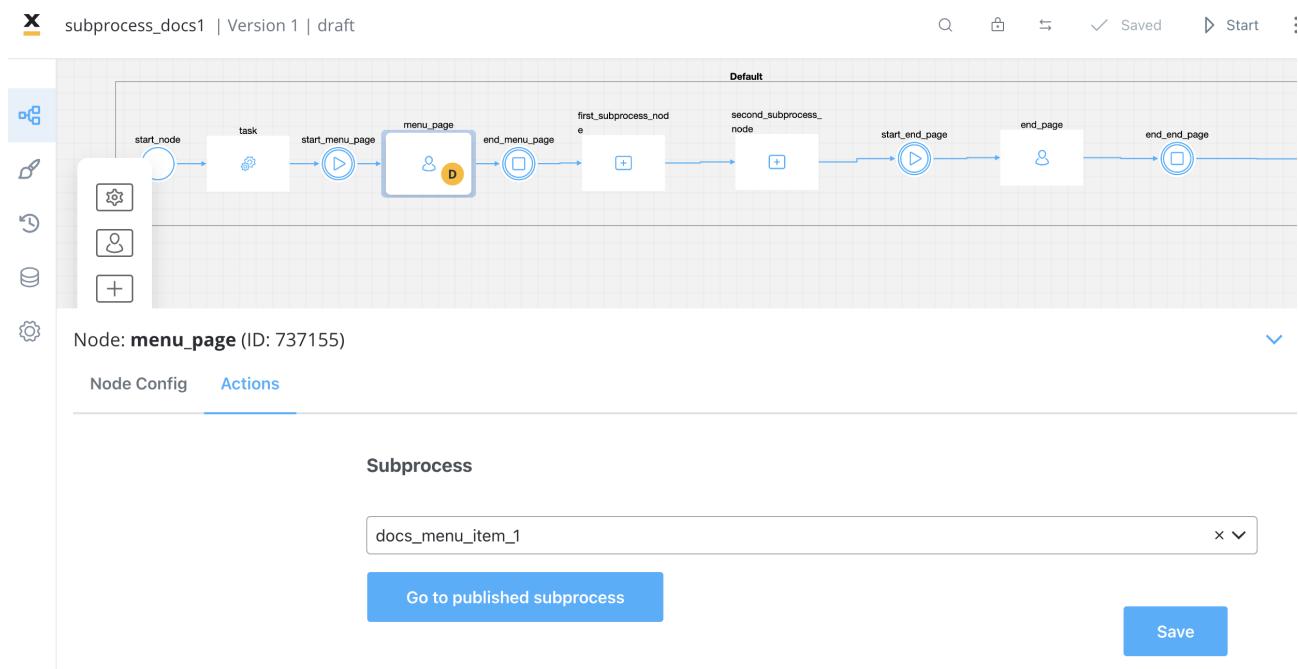
To use a process as a **subprocess**, you must first create it. Once the subprocess is created, you can start it from another (parent) process. To do this, you will need to add a **Start Subprocess** action to a **User task** node in the parent process or by using a **subprocess run node**.

Here are the steps to start a subprocess from a parent process:

1. First, create a **process** designed to be used as a **subprocess**.

2. In the parent process, create a **User task** node where you want to start the subprocess created at step 1.
3. Add a **Start Subprocess** action to the Task Node.
4. Configure the **Start Subprocess** action and from the dropdown list choose the subprocess created at step 1.

By following these steps, you can start a subprocess from a parent process and control its execution based on your specific use case.



The following properties must be configured for a **Start subprocess** action:

- Action Edit
- Back in steps (for Manual actions)
- Parameters
- Data to send (for Manual actions)

Action Edit

- **Name** - used internally to make a distinction between different actions on nodes in the process. We recommend defining an action naming standard to be able to quickly find the process actions
- **Order** - if multiple actions are defined on the same node, the running order should be set using this option
- **Timer expression** - it can be used if a delay is required on that action. The format used for this is [ISO 8601 duration format](#) (for example, a delay of 30 seconds will be set up as `PT30S`)
- **Action type** - should be set to **Start Subprocess**
- **Trigger type** (options are Automatic/Manual) - choose if this action should be triggered automatically (when the process flow reaches this step) or manually (triggered by the user); in most use cases, this will be set to automatic
- **Required type** (options are Mandatory/Optional) - automatic actions can only be defined as mandatory. Manual actions can be defined as mandatory or optional.
- **Repeatable** - 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

Back in steps

- **Allow BACK on this action** - back in process is a functionality that allows you to go back in a business process and redo a series of previous actions in the process. For more details, check [Moving a token backwards in a process](#) section.

Parameters

- **Subprocess** - the name of the process that you want to start as a subprocess
- **Exclude from current state** - what fields do you want to exclude when copying the data from the parent process to the subprocess (by default all data fields are copied)
- **Copy from current state** - if a value is set here, it will overwrite the default behavior (of copying the whole data from the subprocess) with copying just the data that is specified (based on keys)

Node: **menu_page** (ID: 737155) ▼ ▲

Node Config Actions

Copy from current state

selectedCustomer trash

selectedSubscription trash

url trash

trash

trash

Add Param

Advanced configuration

Show Target Process

Save

Advanced configuration

- **Show Target Process** - ID of the current process, to allow the subprocess to communicate with the parent process (which is the process where this action

is configured)

Data to send

- **Keys** - are 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)



Data to send option is configurable only when the action **trigger type** is **Manual**.

Parameters

Subprocess name



Exclude from current state

Add Param

Copy from current state

Add Param

Advanced configuration

Show Target Process

Replace Values

Data to send

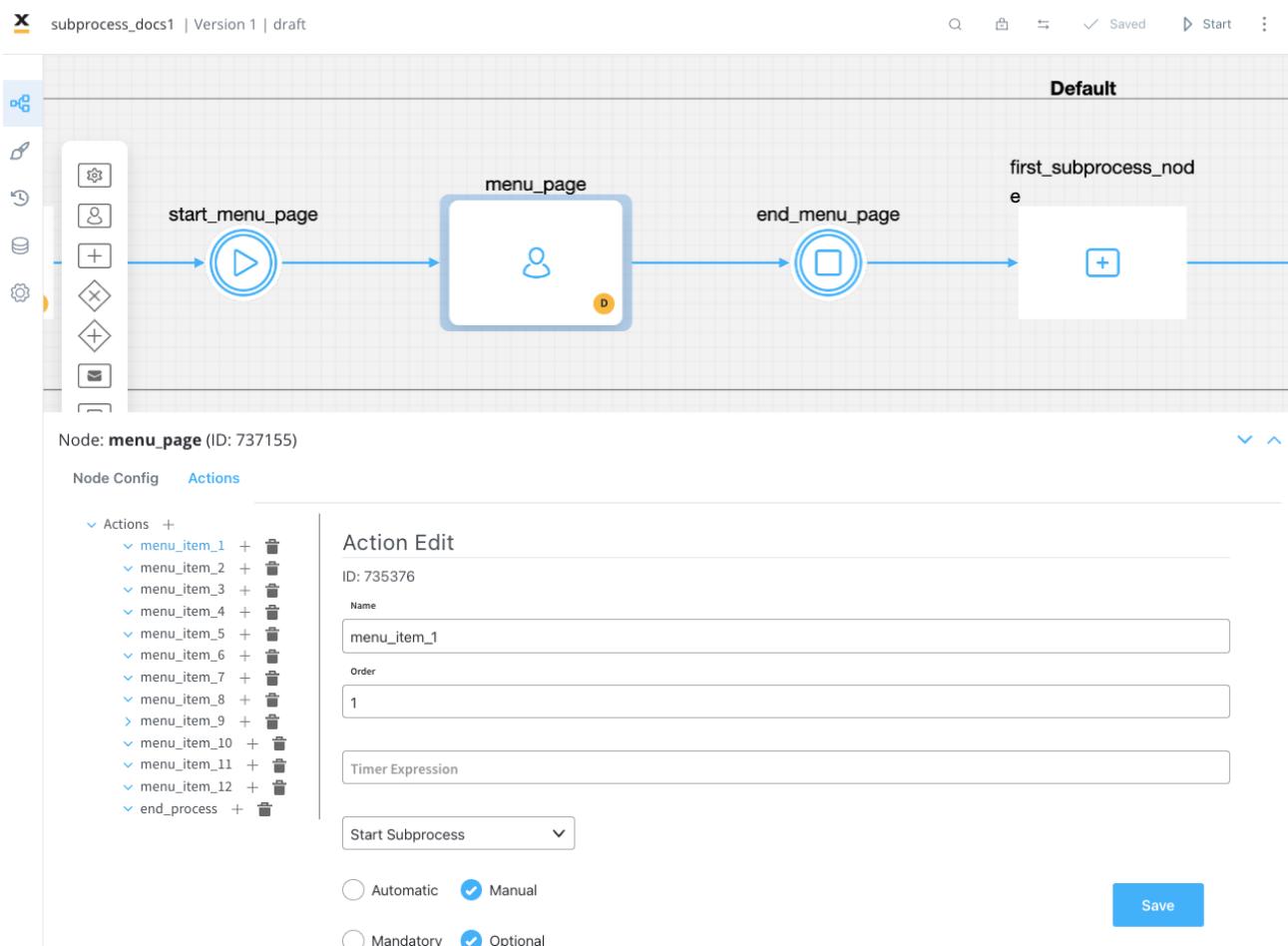
Add KeySave

Example

Let's create a main

The fallback content to display on prerendering

, in this process we will add a user task node that will represent a menu page. In this newly added node we will add multiple subprocess actions that will represent menu items. When you select a menu item, a subprocess will run representing that particular menu item.



To start a subprocess, we can, for example, create the following minimum configuration in a user task node (now we configure the process where we want to start a subprocess):

- **Action** - `menu_item_1` - used internally to make a distinction between different actions on nodes in the process. We recommend defining an action naming standard to be able to quickly find the process actions
- **Trigger type** - Manual; Optional
- **Repeatable** - yes
- **Subprocess** - `docs_menu_item_1` - the name of the process that you want to start as a subprocess

The screenshot shows the configuration interface for a subprocess named 'subprocess_docs1'. The top bar includes a back button, a search icon, a save icon, and other navigation buttons. The main area is divided into sections: 'Node' (showing 'menu_page' ID 737155), 'Actions' (selected tab), 'Parameters' (empty), 'Subprocess' (set to 'docs_menu_item_1'), and 'Exclude from current state' (containing 'test.price'). A sidebar on the left provides quick access to various settings.

- **Exclude from current state** - `test.price` - copy all the data from the parent, except the price data
- **Copy from current state** - leave this field empty in order to copy all the data (except the keys that are specified in the **Exclude from current state** field), if not, add the keys from which you wish to copy the data

The screenshot shows the FLOWX.AI subprocess configuration interface. At the top, it displays the subprocess name "subprocess_docs1 | Version 1 | draft" and various navigation icons. Below this, the title "Node: menu_page (ID: 737155)" is shown, followed by tabs for "Node Config" and "Actions", with "Actions" being the active tab. A sub-header "Copy from current state" is present. On the left, there is a sidebar with icons for copy, paste, undo, redo, and settings. The main area lists several parameters in input fields, each with a delete icon: "application", "application1", "application2", "application3", "application4", "application5", "webSocketAddress", and "webSocketPath". At the bottom of this list is a blue "Add Param" button.

⚠ CAUTION

When copying from the current state using a subprocess, it is mandatory to specify the `webSocketAddress` and `webSocketPath` as parameters. This ensures that the Engine can accurately transmit the relevant information to the frontend, enabling it to display the appropriate UI.

Advanced configuration

- **Target process (parentProcessInstanceId)** - `#{processInstanceId}` - current process ID

Result

The screenshot shows the FLOWX.AI platform interface. On the left, there is a sidebar with navigation links for 'Processes' (Definitions, Active process, Process Instances, Failed process start) and 'Content Management' (Enumerations, Substitution tags, Content models, Languages, Source systems, Media Library). At the bottom of the sidebar, there is a user profile icon for 'Dragos Caravasile' and a three-dot menu. The main area is titled 'Process Definitions' and contains two sections: 'Drafts / In progress' and 'Published'. In the 'Published' section, there is one entry named 'subprocess_docs1' with a version of 1, published on 12 Apr 2023, 4:22 PM by Dragos Caravasile. There are also edit and delete icons next to the entry.

Was this page helpful?

BUILDING BLOCKS / Actions / Append Params to Parent Process

(!) INFO

What is it? It is a type of

The fallback content to display on prerendering
that allows you to send data from a subprocess to a parent process.

Why is it important? If you are using subprocesses that produce data that needs to be sent back to the main

The fallback content to display on prerendering , you can do that by using an **Append Params to Parent Process** action.

Configuring an Append Params to Parent Process

After you create a process designed to be used as a **subprocess**, you can configure the action. To do this, you need to add an **Append Params to Parent Process** on a **Task Node** in the subprocess.

The following properties must be configured:

- Action Edit
- Back in steps (for Manual actions)
- Parameters
- Data to send (for Manual actions)

Action Edit

- **Name** - used internally to make a distinction between different actions on nodes in the process. We recommend defining an action naming standard to be able to quickly find the process actions
- **Order** - if multiple actions are defined on the same node, the running order should be set using this option
- **Timer expression** - it can be used if a delay is required on that action. The format used for this is **ISO 8601 duration format** (for example, a delay of 30 seconds will be set up as **PT30S**)

- **Action type** - should be set to **Append Params to Parent Process**
- **Trigger type** (options are Automatic/Manual) - choose if this action should be triggered automatically (when the process flow reaches this step) or manually (triggered by the user); in most use cases, this will be set to automatic
- **Required type** (options are Mandatory/Optional) - automatic actions can only be defined as mandatory. Manual actions can be defined as mandatory or optional.
- **Repeatable** - 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

Back in steps

- **Allow BACK on this action** - back in process is a functionality that allows you to go back in a business process and redo a series of previous actions in the process. For more details, check **Moving a token backwards in a process** section

Parameters

- **Copy from current state** - data that you want to be copied back to the parent process
- **Destination in the parent state** - on what key to copy the param values



To recap: if you have a **Copy from current state** with a simple **JSON** - `{"age": 17}`, that needs to be available in the parent process, on the `application.client.age` key, you will need to set this field (**Destination**

in the parent state) with `application.client`, which will be the key to append to in the parent process.

Advanced configuration

- **Show Target Process** - ID of the parent process where you need to copy the params, this was made available on to the `${parentProcessInstanceId}` variable, if you defined it when you **started the subprocess**

Data to send

- **Keys** - are 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)

🔥 DANGER

Data to send option is configurable only when the action **trigger type** is **Manual**.

Example

We have a subprocess that allows us to enter the age of the client on the **data.client.age** key, and we want to copy the value back to the parent process. The key to which we want to receive this value in the parent process is **application.client.age**.

This is the configuration to apply the above scenario:

Parameters

- **Copy from current state** - `{"client": ${data.client.age}}` to copy the age of the client (the param value we want to copy)
- **Destination in the parent state** - `application` to append the data to the **application** key on the parent process

Advanced configuration

- **Show Target Process** - `${parentProcessInstanceId}` to copy the data on the parent of this subprocess

Node: Task node (ID: 546052)

Node Config Actions

Actions + appendToParent +

Action Edit

ID: 546651
Name: appendToParent
Order: 1
Timer Expression:
Append Params to Parent Process

Automatic Manual
 Mandatory Optional
 Repeatable
Autorun Children?

Parameters

Copy from current state

```
1 {"client":${data.client.age}}
```

Replace Values

Destination in the parent state

```
application
```

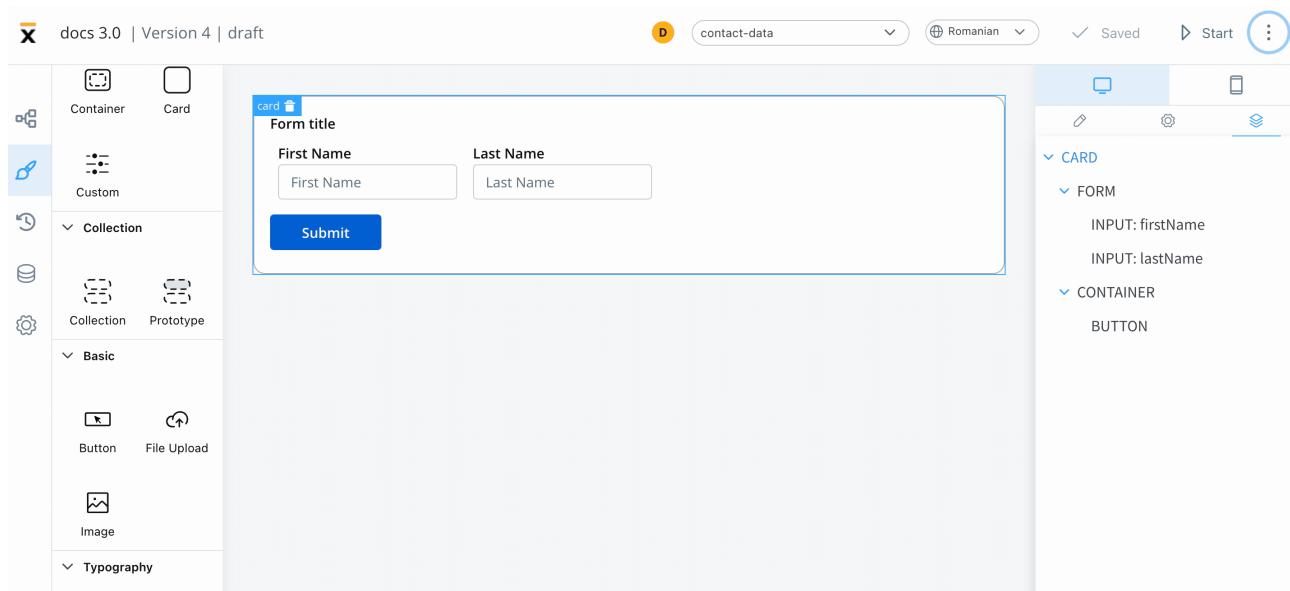
Replace Values

Advanced configuration

Was this page helpful?

BUILDING BLOCKS / UI Designer / UI component types / Root components / Container

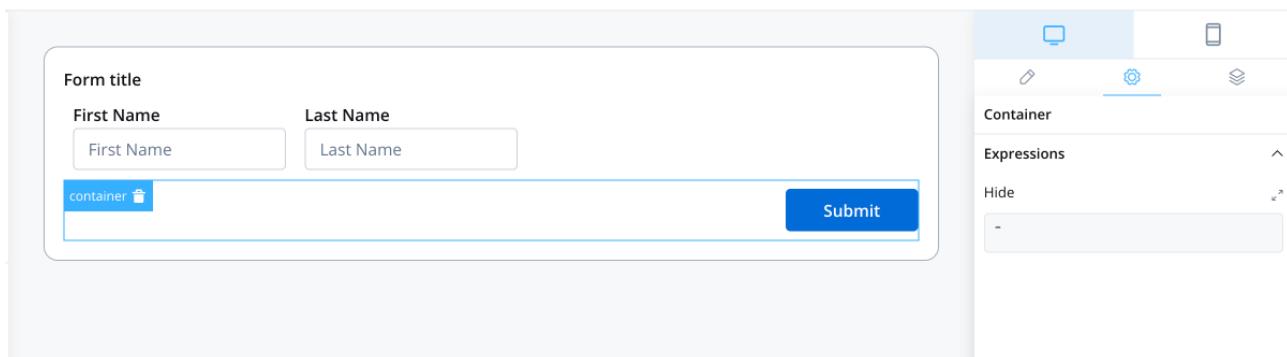
A container is a versatile element that allows you to group components and align them as desired.



The following properties can be configured in the container:

Settings

- **Expressions (Hide)** - JavaScript expressions used to hide components when they evaluate to true



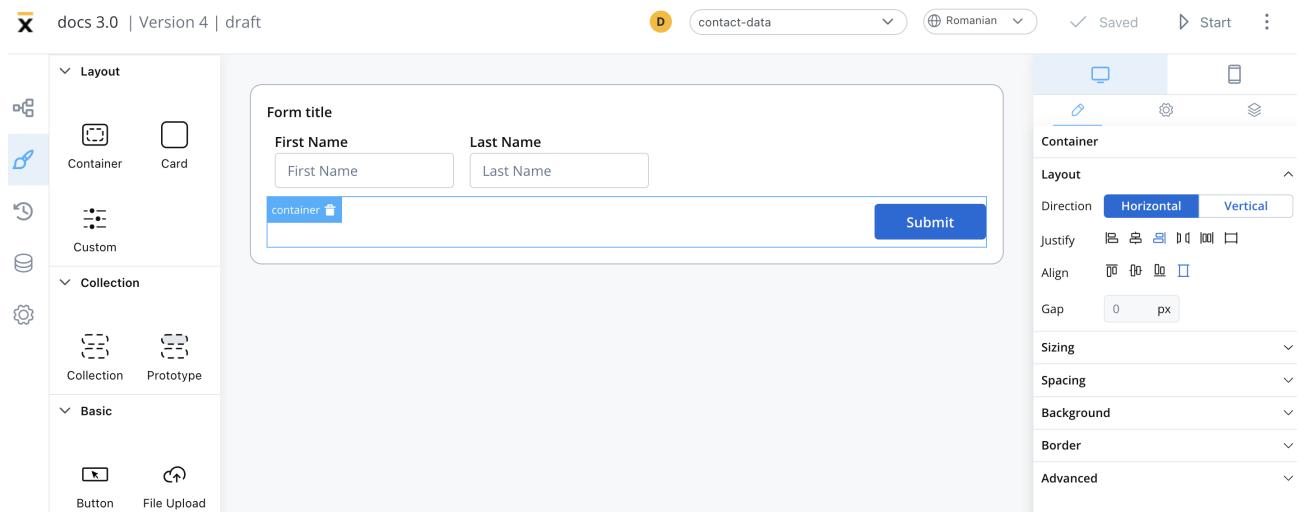
Styling

- **Layout** - this property is available for components that group children and includes the following options:
 - Direction - Horizontal / Vertical (for example, select *Horizontal*)
 - Justify (H) - (for example, select *end*)
 - Align (V) - this option allows you to align components vertically
 - Gap - you can set the gap between components

More layout demos available below:

» [Layout Demos](#)

When you apply the above properties, you can generate the following output, with the button appearing on the right side of the container, underneath the form with three form elements:



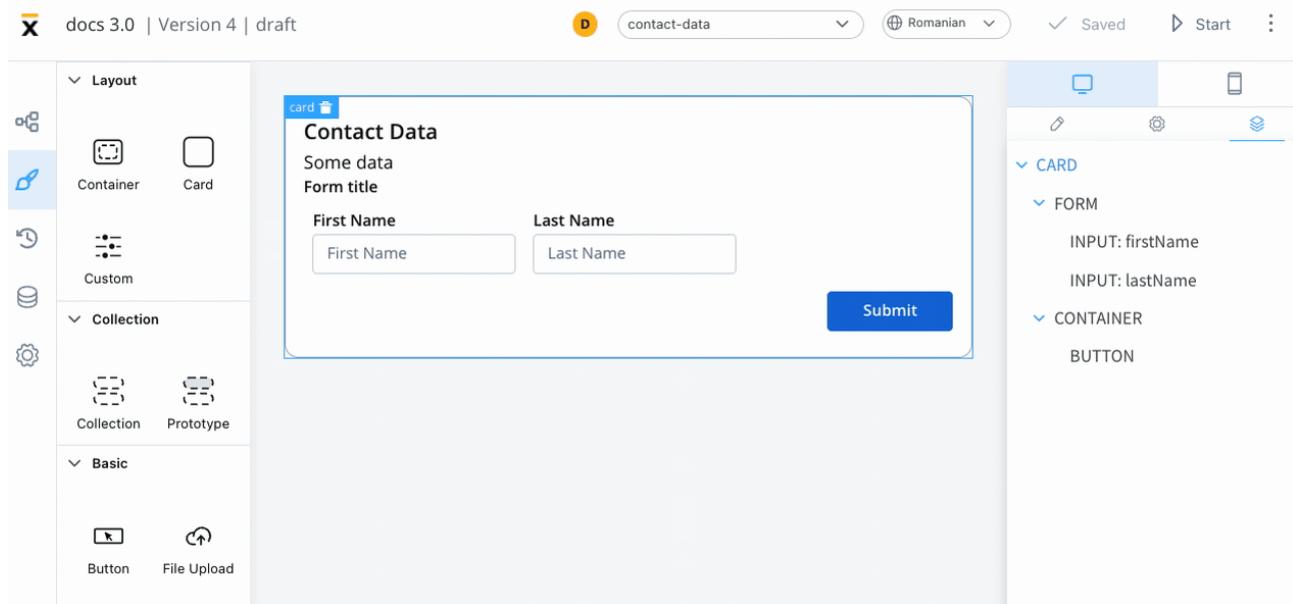
For more information about styling and layout configuration, check the following section:

» [UI Designer](#)

Was this page helpful?

BUILDING BLOCKS / UI Designer / UI component types / Root components / Card

A card is a graphical component that allows grouping and alignment of other components. It can also include an accordion element for expanding and collapsing content.



The following properties that can be configured:

Settings

- **Message** - a valid JSON that describes the data pushed to the frontend application when the process reaches a specific user task
- **Title** - the title of the card
- **Subtitle** - the subtitle of the card
- **Card style** - you can choose between a border or raised style
- **Has accordion?** - this feature allows you to add a Bootstrap accordion, which organizes content within collapsible items and displays only one collapsed item at a time

⚠ CAUTION

Accordion element is not available for mobile.

The screenshot shows a user interface for configuring a card. At the top, there are icons for desktop and mobile devices, followed by edit, gear, and file/folder icons. Below these are sections for 'Card' and 'Message'. The 'Message' section contains a text input field with a placeholder '-' and a clear button (X). A blue header bar labeled 'Properties' has an upward arrow icon. Under 'Properties', there are fields for 'Title' (placeholder: eg. title) and 'Subtitle' (placeholder: eg. subtitle). A 'Card style' section shows two options: 'border' (selected) and 'raised'. At the bottom, there is a checkbox labeled 'Has Accordion'.

Card

Message

Properties

Title eg. title

Subtitle eg. subtitle

Card style border raised

Has Accordion

Styling

- **Layout** - This property is available for components that group children and includes the following options:
 - Direction - Horizontal / Vertical (for example, select *Vertical*)
 - Justify (H) - (for example, select *center*)
 - Align (V) - this option allows you to align components vertically
 - Gap - you can set the gap between components

More layout demos available below:

» [Layout Demos](#)

This example will generate a card with the following layout configuration:

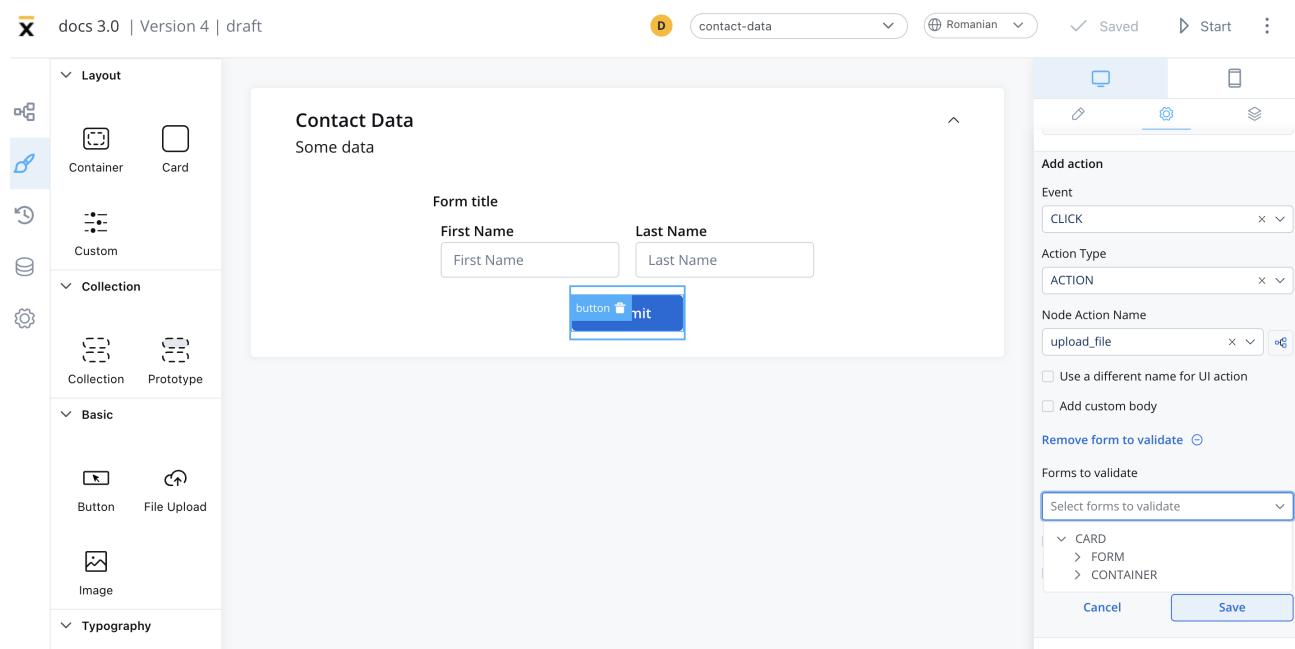
The screenshot shows the FLOWX AI interface. On the left, there's a sidebar with icons for Card, Container, Custom, Collection, Prototype, Button, File Upload, and Image. The main area displays a card component titled "Contact Data" with the sub-section "Some data". Inside the card, there's a form title "Form title" followed by two input fields: "First Name" and "Last Name", each with a placeholder "First Name" and "Last Name" respectively. Below these is a blue "Submit" button. To the right of the card, there's a detailed "Layout" configuration panel. It includes sections for "Card", "Layout", "Direction" (set to "Horizontal"), "Justify", "Align", "Gap" (set to 0 px), "Sizing" (Fit W set to "auto"), and "Spacing" (with values 0, 0, 0, 0). The "Layout" section also has tabs for "Horizontal" and "Vertical", with "Horizontal" currently selected.

For more information about styling and layout configuration, check the following section:

» [UI Designer](#)

Validating elements

To validate all form elements under a card, you need to set the key of the form/element on the property of the button: *Forms To Validate*.

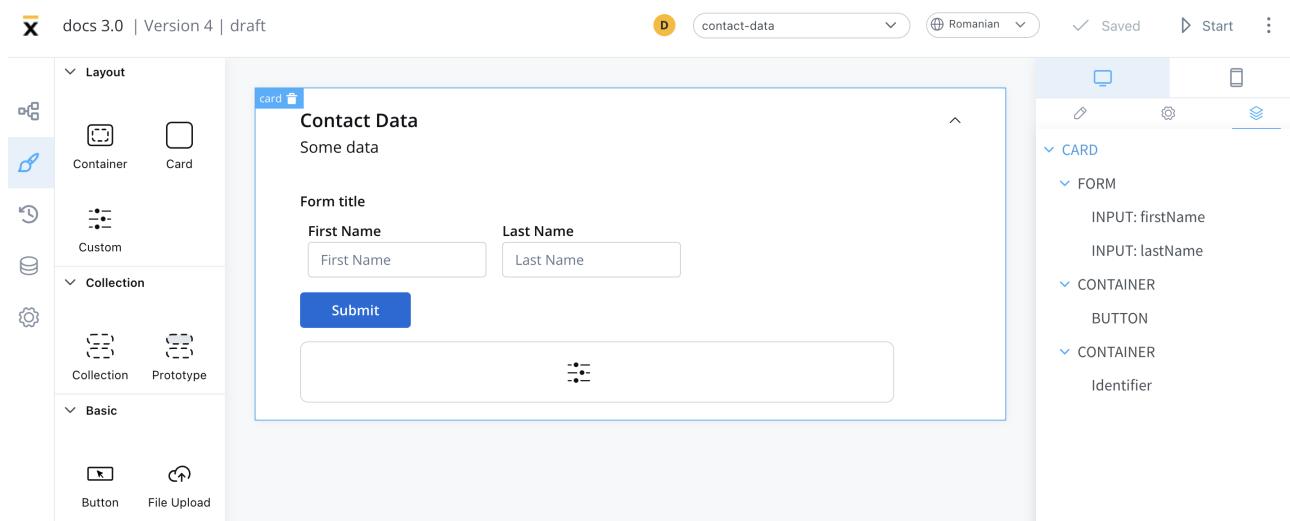


Was this page helpful?

BUILDING BLOCKS / UI Designer / UI component types / Root components / Custom

Custom components are developed in the web application and referenced here by component identifier. This will dictate where the component is displayed in the component hierarchy and what actions are available for the component.

To add a custom component in the template config tree, we need to know its unique identifier and the data it should receive from the process model.



The properties that can be configured are as follows:

- **Identifier** - this will enable the custom component to be displayed in the component hierarchy and what actions are available for the component
- **Input keys** - used to define the process model paths from which the components will receive their data

- **UI Actions** - actions defined here will be made available to the custom component

The screenshot shows the FLOWX.AI platform's configuration interface for a custom component. At the top, there are icons for a computer monitor and a smartphone, followed by a gear icon (selected) and a stack of three squares icon. The main area is divided into sections:

- Custom Identifier**: A section containing a text input field labeled "Identifier". To the right of the input field are two small icons: a left arrow and a right arrow.
- Input Keys**: A section containing a button labeled "CustomComponent" with edit and delete icons to its right. Below it is a button labeled "Add an option" with a plus sign icon.
- UI Action**: A section containing a button labeled "Add UI action" with a plus sign icon.

Display of User Interface Elements

When a process instance is initiated, the web application receives all the UI elements that can be displayed in the process under the `templateConfig` key.

When a user task is reached in the process instance, the **events-gateway** receive requests, triggering it to display the associated UI element.

Example:

1. Starting a process:

- The following is an example of starting a process instance via a **POST** request to

```
{{processUrl}}/api/internal/process/DemoProcess/start:
```

```
{
  "processDefinitionName" : "DemoProcess",
  "tokens" : [ {
    "id" : 662631,
    "startNodeId" : null,
    "currentNodeId" : 662807,
    "currentnodeName" : null,
    "state" : "ACTIVE",
    "statusCurrentNode" : "ARRIVED",
    "dateUpdated" : "2023-02-09T12:23:19.464155Z",
    "uuid" : "ae626fda-8166-49e8-823b-fe24f36524a7"
  } ],
  "state" : "CREATED",
  "templateConfig" : [ {
    "id" : 630831,
    "flowxUuid" : "80ea0a85-2b0b-442a-a123-2480c7aa2dce",
    "nodeDefinitionId" : 662856,
    "uiDefinitions" : [
      {
        "id" : 662856,
        "name" : "User Task 1"
      }
    ]
  } ]
}
```

```
"componentIdentifier" : "CONTAINER",
"type" : "FLOWX",
"order" : 1,
"canGoBack" : true,
"displayOptions" : {
  "flowxProps" : { },
  "style" : null,
  "flexLayout" : {
    "fxLayoutGap" : 0,
    "fxLayoutAlign" : "start stretch",
    "fxLayout" : "column"
  },
  "className" : null,
  "platform" : "DEFAULT"
},
"templateConfig" : [ {
  "id" : 630832,
  "flowxUuid" : "38e2c164-f8cd-4f6e-93c8-39b7cdd734cf",
  "nodeDefinitionId" : 662856,
  "uiTemplateParentId" : 630831,
  "componentIdentifier" : "TEXT",
  "type" : "FLOWX",
  "order" : 0,
  "key" : "",
  "canGoBack" : true,
  "displayOptions" : {
    "flowxProps" : {
      "text" : "Demo text"
    },
    "style" : null,
    "flexLayout" : null,
    "className" : null,
    "platform" : "DEFAULT"
  },
  "expressions" : {
```

```
        "hide" : """",
    },
    "templateConfig" : [ ],
    "dataSource" : [
        "processData" : {
            "parentFlowxUuid" : null
        },
        "nomenclator" : {
            "parentFlowxUuid" : null
        }
    ]
},
{
    "uuid" : "d985d128-ae45-4408-a643-1dd026a644d3",
    "generalData" : null,
    "backCounter" : 0,
    "startedByActionId" : null,
    "subProcesses" : null,
    "subprocessesUuids" : null,
    "baseUrl" : null
}
```

2. The following is an example of a progress message:

```
{
    "progressUpdateDTO": {
        "processInstanceUuid": "5f24c66f-04a7-433a-b64a-
a765d3b8121a",
        "tokenUuid": "11c32ba6-b3e7-4267-9383-25d69b26492c",
        "currentNodeId": 662856
    }
}
```

3. **ProgressUpdateDto** will trigger the **SDK** to search for the UI element having the same **nodeId** as the one from the web socket progress event
4. Additionally, it will ask for data and actions that are required for this component via a GET request `{{processUrl}}/api/process/5f24c66f-04a7-433a-b64a-a765d3b8121aa/data/662856`

Was this page helpful?

BUILDING BLOCKS / UI Designer / UI component types / Collection / Collection Prototype

Description

This additional container type is needed to allow multiple prototypes to be defined for a single **Collection**. This allows elements from the same collection to be displayed differently.

For example, suppose you are creating a piece of UI in which the user is presented a list of possible products from which to choose, but you want one of the products to be highlighted as the recommended one. This example requires a collection with two **collection prototypes** (one for the normal product and one for the recommended one).

Configurable properties:

1. **Prototype Identifier Key** - the key where to look in the iterated object to determine the prototype to be shown - in the below example the key is "type"
2. Prototype Identifier Value - the value that should be present at the **Prototype Identifier Key** when this **COLLECTION_PROTOTYPE** should be displayed - in the below example the value is "normal" or "recommended"

Example

The screenshot shows the FLOWX.AI interface for a project titled "test-collections" (version 1, draft). The left sidebar contains a navigation menu with sections like Layout, Collection, and Basic. Under Collection, there are icons for Container, Card, Custom, Collection, and Prototype. The main workspace displays a "Collection prototype example" with two items. The first item is a "Container" with a blue header bar labeled "collection". It contains two text fields: "\${name}" and "\${description}". The second item is a "Collection" with a yellow header bar labeled "collection". It also contains two text fields: "\${name}" and "\${description}". On the right side, there is a toolbar with icons for preview, settings, and export. Below the toolbar, there is a detailed view of the selected item's properties. For the Container, it shows "CONTAINER" and "TEXT" under "COLLECTION". For the Collection, it shows "COLLECTION_PROTOTYPE" with "IMAGE" and "TEXT" options, and another "COLLECTION_PROTOTYPE" section with "IMAGE".

The screenshot shows the FLOWX.AI interface for a project titled "test-collections" (version 1, draft). On the left, there is a sidebar with icons for Container, Card, Custom, Collection, and Prototype. Below these are sections for Layout, Collection, and Basic. In the main area, there is a title "Collection prototype example" followed by two cards labeled "collection_prototype". Each card contains placeholder text "\${name}" and "\${description}". To the right of the cards is a "Prototype Settings" panel with fields for "Prototype Identifier Key" (set to "type") and "Prototype Identifier Value" (set to "normal"). There is also a section for "ui Actions" with a button to "Add ui action".

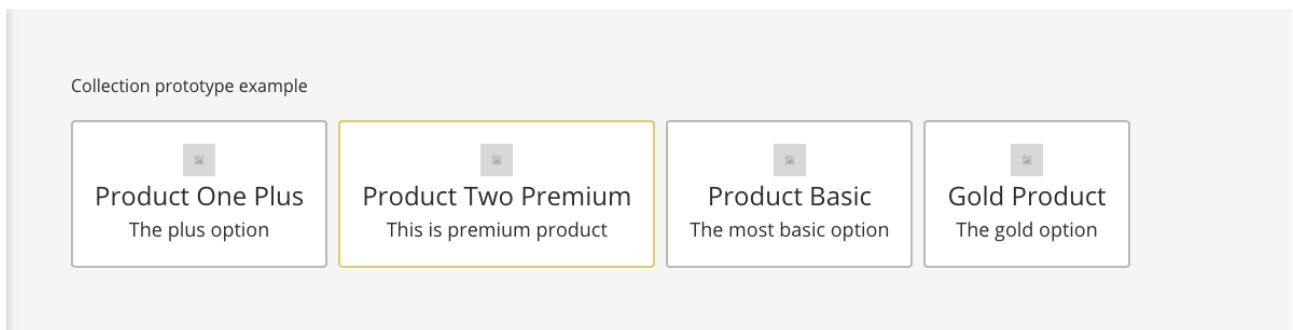
This screenshot is similar to the one above, but the "Prototype Identifier Value" field in the "Prototype Settings" panel is now set to "recommended". The rest of the interface remains the same, showing the collection prototype example with two cards and the associated settings.

Source collection data example for products:

```
products: [
  {
    name: 'Product One Plus',
    description: 'The plus option'
    type: 'normal'
  },
  {
    name: 'Product Two Premium',
    description: 'The premium option'
    type: 'normal'
  }
]
```

```
        description: 'This is premium product'
        type: 'recommended',
    },
{
    name: 'Product Basic',
    description: 'The most basic option'
    type: 'normal'
},
{
    name: 'Gold Product',
    description: 'The gold option'
    type: 'normal',
}
]
```

The above configuration will render:



Adding elements with UI Actions

There are a few differences you need to take into consideration when configuring elements that make use of **UI Actions** inside a **Collection Prototype**.

To showcase these differences, we'll use the next example:

(Some) FlowX employees:

 Mihai Saru	<button>Select</button>	<button>Save Name</button>
 Andra Popazu	<button>Select</button>	<button>Save Name</button>

We have a **Collection** with two employees and we want to provide the user with the option of selecting one of the employees (eg. to allow for further processing in the next steps of the process).

Step 1 - Defining the Node Action

To select one employee from the list, we first must add an **Action** to the **User Task Node** this UI is attached to:

Node: collection-1 (ID: 431461)

Actions

Action Edit
ID: 431905
Name:
save-item
Order:
1
Timer Expression
Save Data
 Automatic Manual
 Mandatory Optional
 Repeatable
Autorun Children?
Allow BACK on this action?
Data to send
selectedEmployee

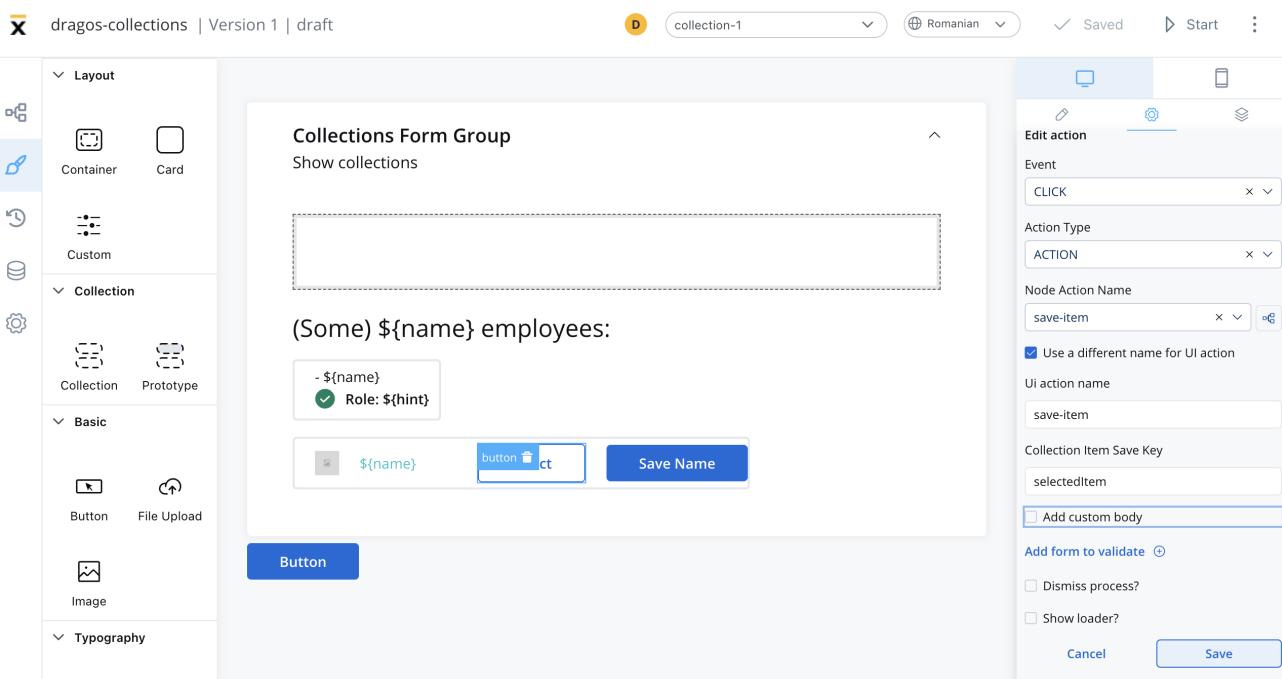
This **save-item** action is **manual** (since it will be triggered by the user) and **optionally** (since selecting an employee is not a requirement to go to the next **Node** in the process).

To allow the user to change his mind about the selected employee, this action is also marked as **Repeatable**.

Keep in mind to check the **Data to send** section. Here we are telling the platform where we want the selected employee (for which the user pressed the **Select** button) to be saved in the **process data**. In this example, we want it to be saved under the `selectedEmployee` key.

Step 2 - Adding the Button & UI Action

Now that we have a **Node Action** defined, we can go ahead and add the **Select** button in the UI of the **User Task** which contains the Employees Collection.



Collection Item Save Key field has an important role in the UI Action configuration of the **Select** button. This field represents how we pass the value of the **Employee** that the user has selected to the **Node Action** that we created in **Step 1**, named `save-item`.

In our example, we set **Collection Item Save Key** to be `selectedEmployee`.

🔥 DANGER

IMPORTANT: `selectedEmployee` key is how we expose the data from the **Collection** to the Node Action. It is **imperative** that the name in the **Collection Item Save Key** is the same as the one used in the **Data to send** input in the Node Action.

The button and UI action are mostly configured as any other Button and UI Action would be configured.

Result

This is how the process data looked before we pressed the **Select** button for an employee:

Process status

Data:

```
processInstanceId: 483001
processData: Object {"companies": [{"employees": [{"name": "Mihai Saru", "imageSrc": "https://upload.wikimedia.org/wikipedia/commons/thumb/1/1b/Svelte_Logo.svg/170px-Svelte_Logo.svg.png", "type": "color"}, {"name": "John Doe", "imageSrc": "https://upload.wikimedia.org/wikipedia/commons/thumb/1/1b/Svelte_Logo.svg/170px-Svelte_Logo.svg.png", "type": "color"}], "id": 1}, {"id": 1}], tokenId: 483051, tokenUuid: "ae109887-e5aa-46f9-ba18-668ccce33ce8", webSocketPath: "/ws/updates/process", processInstanceId: "01d5b64d-7c61-4d98-a590-d23336f89f95", webSocketAddress: "ws://public.qa.flowxai.dev/01d5b64d-7c61-4d98-a590-d23336f89f95"}
```

Tokens

Token uuid	Token Status	Status Current Node	Date updated
ae109887-e5aa-46f9-ba18-668ccce33ce8	ACTIVE	EXECUTED_PARTIAL	04 May 2022, 12:09 PM

This is how the process data looks after we selected an employee from the list (notice the new field `selectedEmployee`):

Process status

Data:

```
processInstanceId: 483001
processData: Object {"companies":[{"employees":[{"name":"Mihai Saru","imageSrc":"https://upload.wikimedia.org/wikipedia/commons/thumb/1/1b/Svelte_Logo.svg/1702px-Svelte_Logo.svg.png","type":"colored"}]}]
tokenId: 483051
selectedEmployee:
  name: "Mihai Saru"
  imageSrc: "https://upload.wikimedia.org/wikipedia/commons/thumb/1/1b/Svelte_Logo.svg/1702px-Svelte_Logo.svg.png"
  type: "colored"
tokenUuid: "ae109887-e5aa-46f9-ba18-668ccce33ce8"
webSocketPath: "/ws/updates/process"
processInstanceUuid: "01d5b64d-7c61-4d98-a590-d23336f89f95"
webSocketAddress: "wss://public.qa.flowxai.dev/01d5b64d-7c61-4d98-a590-d23336f89f95"
```

Tokens	Token Status	Status Current Node	Date updated
Token uuid ae109887-e5aa-46f9-ba18-668ccce33ce8	ACTIVE	EXECUTED_PARTIAL	04 May 2022, 12:09 PM

Was this page helpful?

BUILDING BLOCKS / UI Designer / UI component types / Buttons

There are two types of buttons available, each with a different purpose. These types are:

- Basic button
- File upload button

▼ Basic



Button



File Upload

Basic button

Basic buttons are used to perform an action such as unblocking a token to move forward in the process, sending an OTP, and opening a new tab.

Configuring a basic button

When configuring a basic button, you can customize the button's settings by using the following options:

- **Properties**
- **UI action**
- **Button styling**

Sections that can be configured regarding general settings:

Properties

- **Label** - it allows you to set the label that appears on the button

UI action

Here, you can define the UI action that the button will trigger.

- **Event** - possible value: CLICK
- **Action Type** - select the action type

The screenshot shows the FLOWX.AI UI Designer interface. On the left, there's a sidebar with various components like Layout, Collection, and Basic. In the center, a form titled 'Amaazing Process | Version 1 | draft' is displayed. The form contains fields for Customer Name, Income, Gender, and two radio buttons for contact preferences. Below these are sections for selecting a date and newsletter subscription. An input label field contains placeholder text 'Some text here'. At the bottom is a blue 'Submit' button, which is highlighted with a red border. On the right side, there's a toolbar with icons for desktop and mobile view, and a panel for configuring the 'Button' properties, including the label 'Submit' and the 'UI Action' section which has an 'Add UI action' button.

More details on how to configure UI actions can be found [here](#).

Button styling

Properties

This section enables you to select the type of button using the styling tab in the UI Designer. There are four types available:

- Primary
- Secondary
- Ghost
- Text

!(INFO)

For more information on valid CSS properties, click [here](#)

Icons

To further enhance the Button UI element with icons, you can include the following properties:

- **Icon Key** - the key associated in the Media library, select the icon from the **Media Library**
- **Icon Color** - select the color of the icon using the color picker

! INFO

When setting the color, the entire icon is filled with that color, the SVG's fill. Avoid changing colors for multicolor icons.

- **Icon Position** - define the position of the icon within the button:
 - Left
 - Right
 - Center

! INFO

When selecting the center position for an icon, the button will display the icon only.

The form includes the following fields:

- Loan amount: A slider set to 255000 \$.
- Form title: A field labeled "Form title".
- Down payment: A slider set to 38250 \$.
- Form title: A second field labeled "Form title".
- Loan type: A dropdown menu showing "USDA".
- A blue button at the bottom left with a white icon.

By utilizing these properties, you can create visually appealing Button UI elements with customizable icons, colors, and positions to effectively communicate their purpose and enhance the user experience.

File upload

This button will be used to select a file and do custom validation on it. Only the Flowx props will be different.

Configuring a file upload button

When configuring a file upload button, you can customize the button's settings by using the following options:

- **Properties**
- **UI action**
- **Button styling**

Sections that can be configured regarding general settings:

Properties

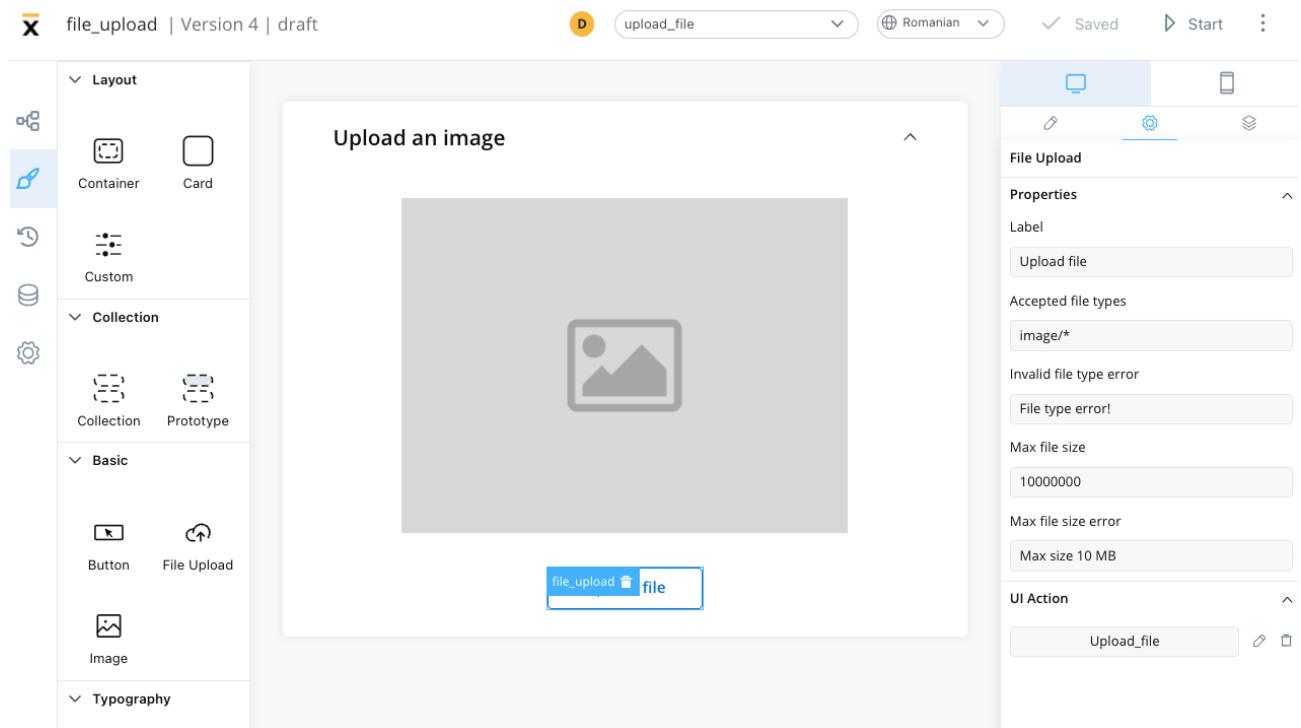
- **Label** - it allows you to set the label that appears on the button
- **Accepted file types** - the accept attribute takes as its value a string containing one or more of these unique file type specifiers, **separated by commas**, may take the following forms:

Value	Definition
audio/*	Indicates that sound files are accepted

Value	Definition
image/*	Indicates that image files are accepted
video/*	Indicates that video files are accepted
MIME type with no params	Indicates that files of the specified type are accepted
string starting with U+002E FULL STOP character (.) (for example, .doc, .docx, .xml)	Indicates that files with the specified file extension are accepted

- **Invalid file type error**
- **Max file size**
- **Max file size error**

Example of an upload file button that accepts image files:



UI action

Here, you can define the UI action that the button will trigger.

- **Event** - possible value: `CLICK`
- **Action Type** - select the action type

Upload an image



file_upload file

File Upload

Properties

Label

Upload file

Accepted file types

image/*

Invalid file type error

File type error!

Max file size

10000000

Max file size error

Max size 10 MB

UI Action

Upload_file

Edit action

Event

CLICK

Action Type

ACTION

DISMISS

ACTION

START_PROCESS_INHERIT

UPLOAD

EXTERNAL

Add form to validate +

Dismiss process?

Show loader?

Cancel Save

INFO

More details on how to configure UI actions can be found [here](#).

Button styling

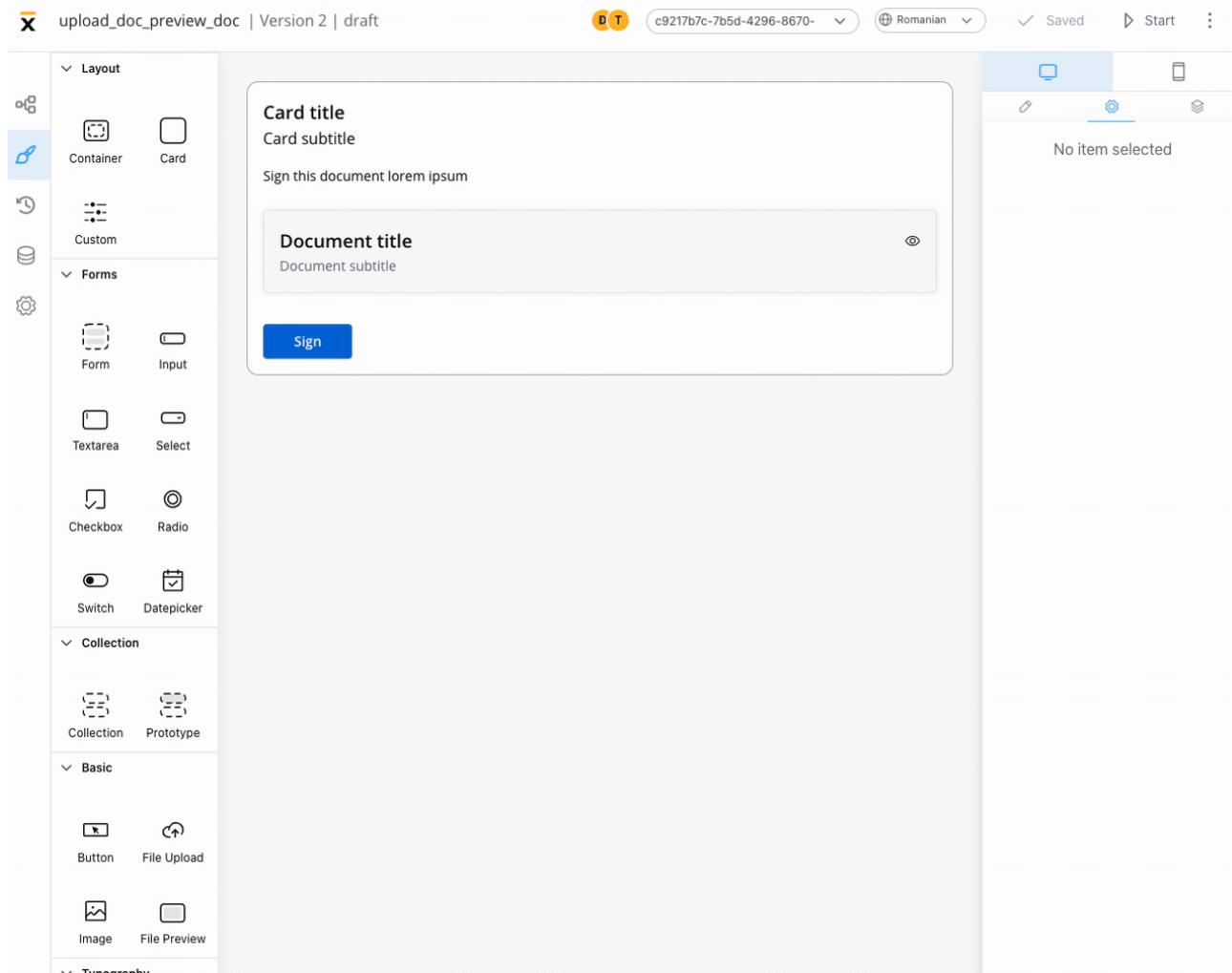
The file upload button can be styled using valid CSS properties (more details [here](#))

[Was this page helpful?](#)

BUILDING BLOCKS / UI Designer / UI component types / File Preview

What is a File Preview UI element?

The File Preview UI element is a user interface component that enables users to preview the contents of files quickly and easily without fully opening them. It can save time and enhance productivity, providing a glimpse of what's inside a file without having to launch it entirely.

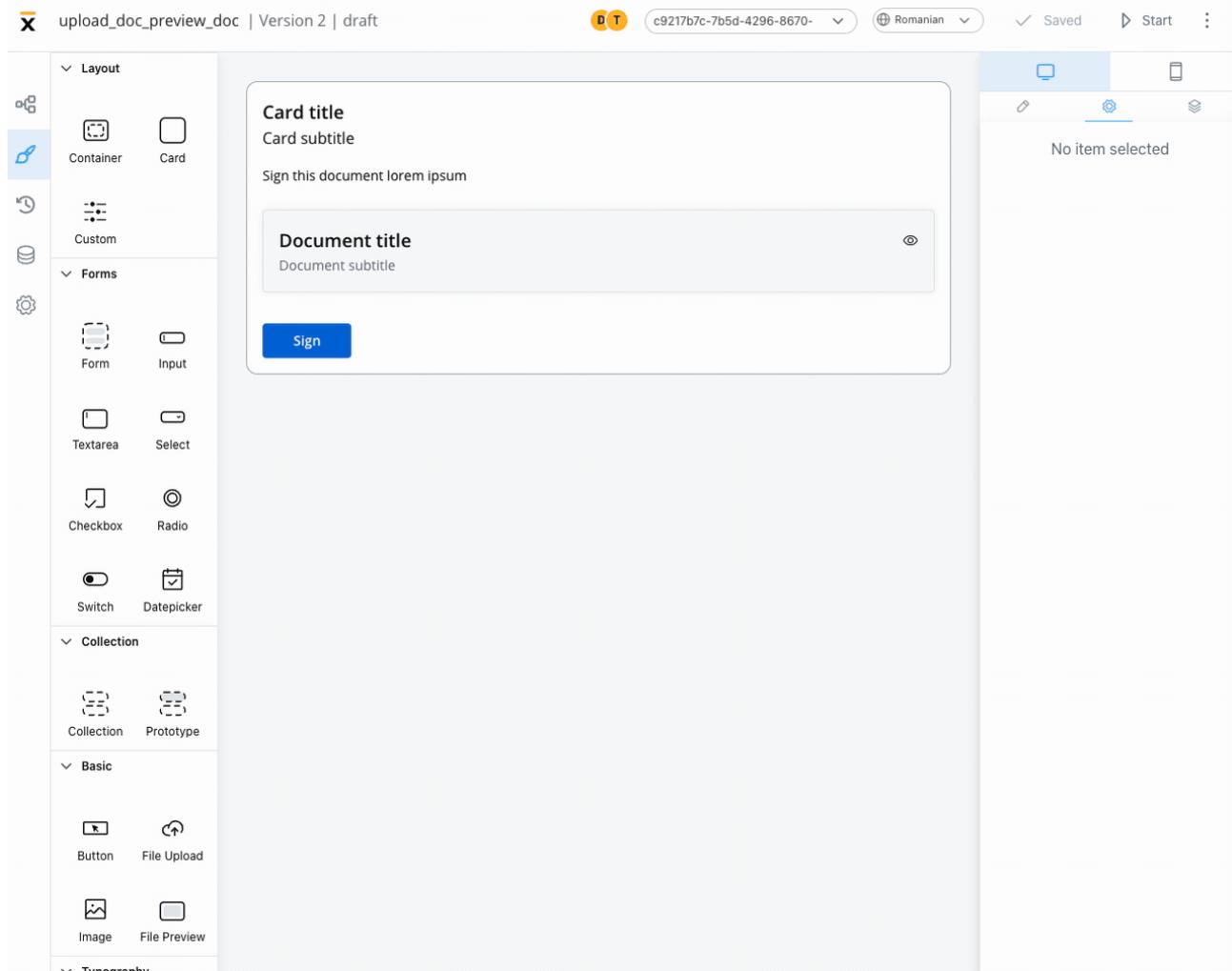


File Preview UI elements offer various benefits such as conveying information, improving the aesthetic appeal of an interface, providing visual cues and feedback or presenting complex data or concepts in a more accessible way.

Configuring a File Preview element

A File Preview element can be configured for both mobile and web applications.

File Preview properties (web)



The File Preview element settings consist of the following properties:

- **Title** - the title of the element (if it is downloaded or shared - the file name should be the title used in preview component)
- **Has subtitle** - the subtitle of the element
- **Display mode** - depending on the selected display method the following properties are available:
 - **Inline → Has accordion:**

- `false` - display preview inline, without expand/collapse option
 - `true` - Default View: Collapsed - display preview inline, with expand/collapse option, by default collapsed
 - `true` - Default View: Expanded - display preview inline, with expand/collapse option, by default expanded
- **Modal** → view icon is enabled
- **Source Type** -
 - **Process Data** - process key where the document is found (creates the binding between the element and process data)
 - **Static** - URL of the document

CAUTION

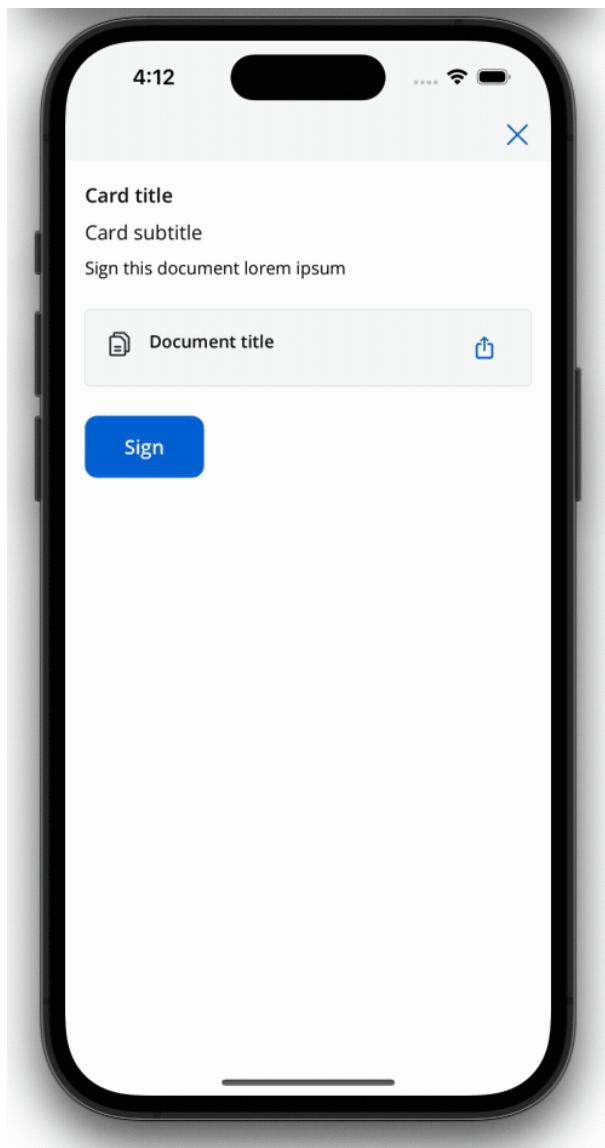
It's worth noting that the inline modal view can raise accessibility issues if the file preview's height exceeds the screen height.

File Preview properties (mobile)

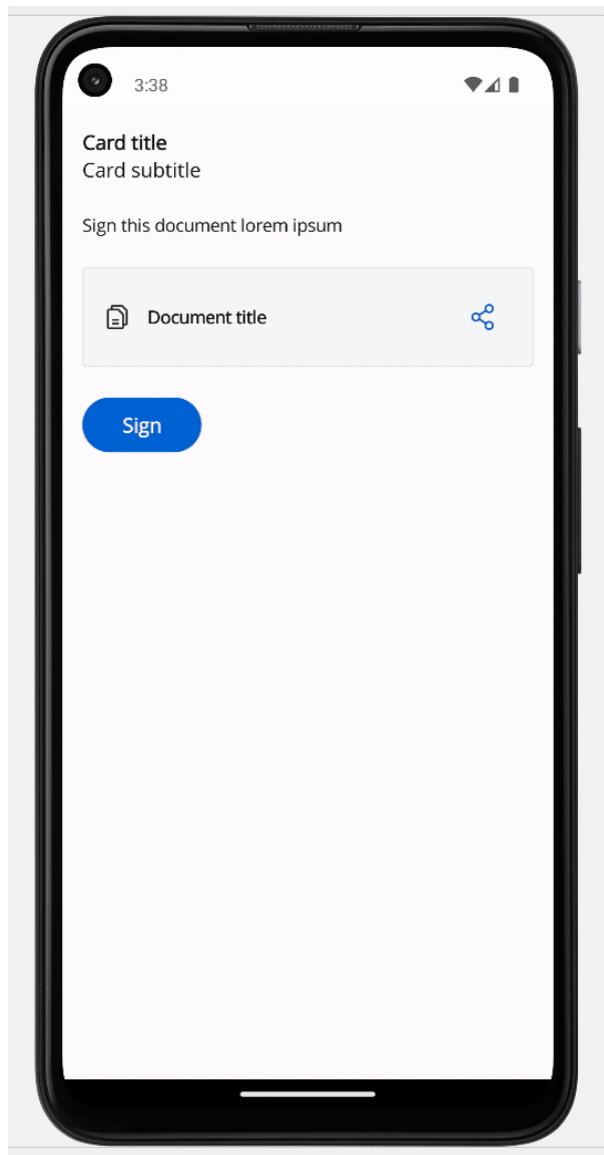
INFO

Both iOS and Android devices support the share button.

iOS



Android



File preview styling

The File Preview styling property enables you to customize the appearance of the element by adding valid CSS properties, for more details, click [here](#).

When drag and drop a File Preview element in UI Designer, it comes with the following default styling properties:

Sizing

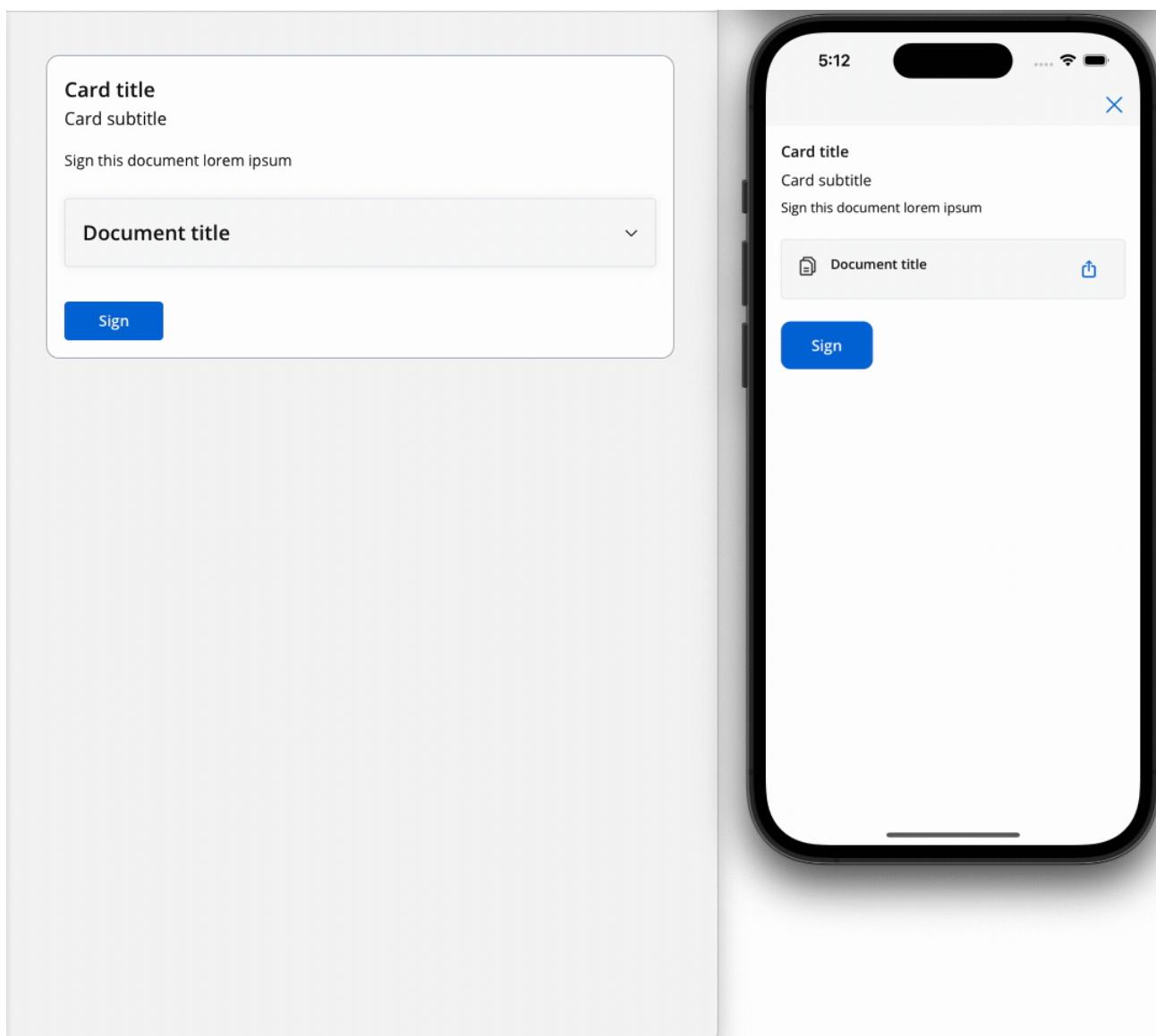
- **Fit W - auto**
- **Fit H - fixed / Height - 400 px**

The screenshot shows the FLOWX.AI interface with a 'File Preview' component. The preview area displays a PDF document titled 'Test PDF' with sample text. The right side of the screen shows the configuration for this element, specifically the 'Sizing' section which is set to 'Fit W: auto' and 'Fit H: fixed Height 400 px'. Other visible properties include 'Typography' (Title and Subtitle colors), 'Background' (color #1E1E1C), 'Border' (radius 0px, width 0px, color #1E1E1C), and 'Advanced' (add class). The left sidebar lists various UI components like Container, Card, Custom, Form, Input, Textarea, Select, Checkbox, Radio, Switch, Datepicker, Collection, Prototype, Button, and File Upload.

File Preview example

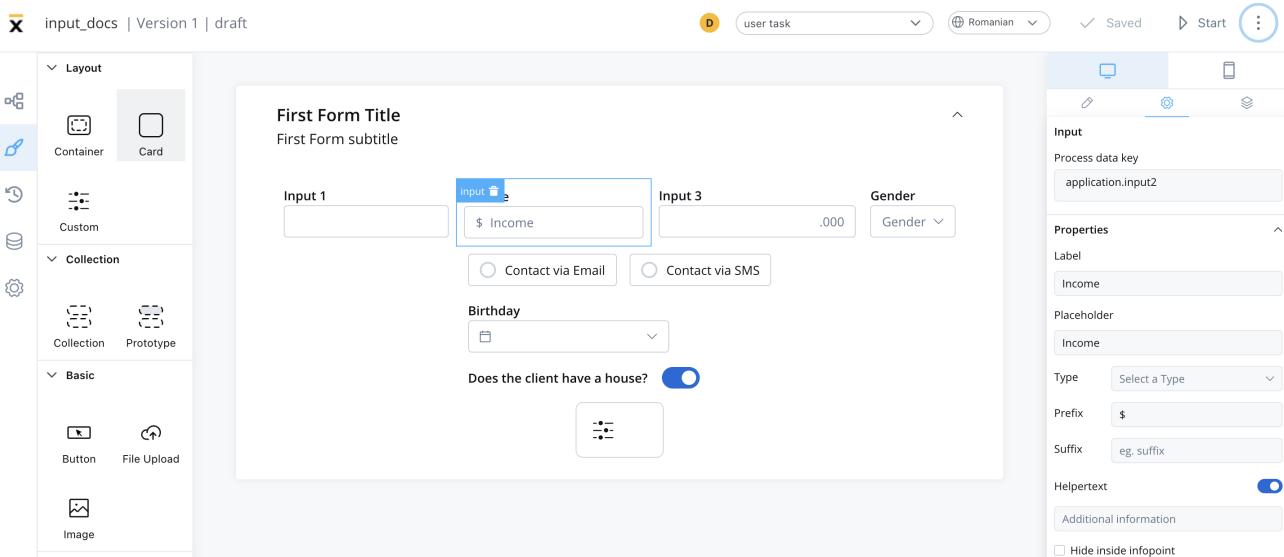
Below is an example of a File Preview UI element with the following properties:

- **Display mode - Inline**
- **Has accordion - True**
- **Default view - Expanded**
- **Source Type - Static**



Was this page helpful?

BUILDING BLOCKS / UI Designer / UI component types / Form elements / Input



An input field is a form element that enables users to input data with validations and can be hidden or disabled.

Configuring the input element

Input settings

The Input Field offers the following configuration options:

- General
- Properties
- Datasource

- **Validators**
- **Expressions**
- **UI actions**
- **Input styling**

General

- **Process data key** - creates the binding between form element and process data, so it can be later used in [decisions](#), [business rules](#) or [integrations](#)

Properties

- **Label** - the label that appears on the input field
- **Placeholder** - the placeholder text that appears in the input field when it is empty
- **Type** - the type of data that the input field can accept, such as text, number, email, or password
- **Prefix** - a label that appears as a prefix to the input field
- **Suffix** - a label that appears as a suffix to the input field
- **Helpertext** - additional information about the input field (can be hidden inside an infopoint)

The screenshot shows the FLOWX.AI process builder interface. On the left is a sidebar with icons for Image, Typography (Text, Link), Forms (Form, Input, Textarea, Select, Checkbox, Radio, Switch, Datepicker), and a general category. The main area displays a form titled "First Form Title" with a subtitle "First Form subtitle". The form contains several input fields: "Input 1" (Text), "Input 2" (Text with placeholder "\$ Income" and helper text "This is a helper text"), "Input 3" (Text with placeholder ".000"), and a "Gender" dropdown. Below these are two radio buttons for "Contact via Email" and "Contact via SMS", and a date picker for "Birthday". A toggle switch labeled "Does the client have a house?" is also present. To the right of the form is a sidebar for "Input" configuration, showing "Process data key: application.input2" and various properties like Label, Placeholder, Type (number), Prefix (\$), Suffix (eg. suffix), and Helpertext (with a checked checkbox). There is also a "Hide inside infopoint" option.

Datasource

The default value for the element can be configured here, this will autofill the input field when you will run the process.

This screenshot is similar to the previous one but includes a "Datasource" configuration sidebar on the right. The sidebar shows the "Default value" field set to "555". Other sections visible include "Validators", "Expressions", and "UI Action". The rest of the interface is identical to the first screenshot, showing the "First Form Title" form and its configuration options.

First Form Title ^

First Form subtitle

Income Mortgage Gender

\$ 555 .000 Gender ▾

This is a helper text

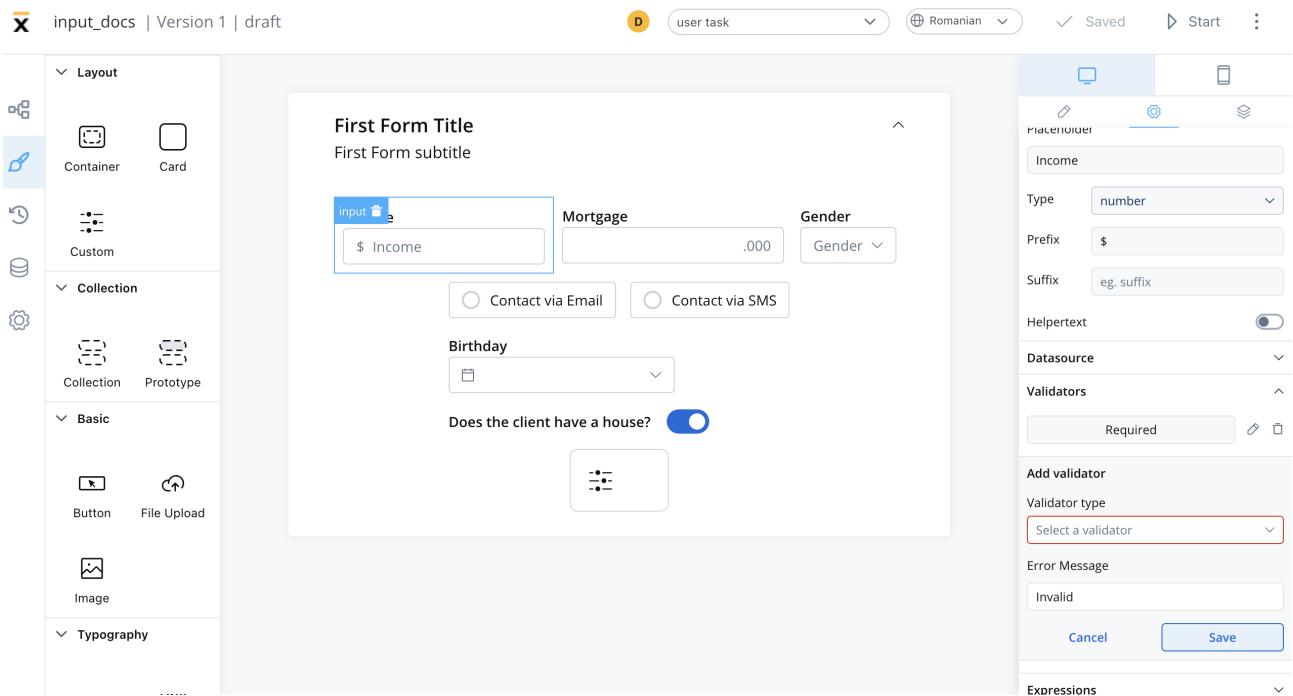
Contact via Email Contact via SMS

Birthday ▼

Does the client have a house?

Validators

There are multiple validators can be added to an input (more details [here](#)).



Expressions

The input field's behavior can be defined using JavaScript expressions for hiding or disabling the element. The following properties can be configured for expressions:

- **Hide** - JavaScript expression used to hide the Input Field when it returns a truthy value
- **Disabled** - JavaScript expression used to disable the Input Field when it returns a truthy value

! INFO

It's important to make sure that disabled fields have the same expression configured under the path expressions → hide.

The screenshot shows the FLOWX.AI UI builder interface. On the left, there's a sidebar with categories: Layout (Container, Card, Custom), Collection (Collection, Prototype), and Basic (Button, File Upload). The main area displays a form titled "First Form Title" with a subtitle "First Form subtitle". The form contains several input fields: a text input for "Income" with a placeholder "\$ Income", a numeric input for "Mortgage" with a placeholder ".000", a dropdown for "Gender", and two radio buttons for "Contact via Email" and "Contact via SMS". Below these are a date input for "Birthday" and a toggle switch for "Does the client have a house?". To the right of the form is a panel for configuring the "Income" field, which includes settings for Placeholder, Type (number), Prefix (\$), Suffix (eg. suffix), Helpertext, Datasource, Validators, and Expressions. The "Expressions" section is highlighted with a red border and contains the expression \${application.key}=='TEST'.

UI actions

UI actions can be added to the Input Field to define its behavior and interactions.

- **Event** - possible value: CHANGE
- **Action Type** - select the action type

The screenshot shows the FLOWX.AI form builder interface. On the left, there's a sidebar with categories like Layout, Collection, and Basic, each containing icons for Container, Card, Custom, Collection, Prototype, Button, File Upload, and Image. The main area displays a form titled "First Form Title" with a subtitle "First Form subtitle". The form contains several fields: an input field with a dollar sign prefix and a placeholder "Income", a text field for "Mortgage", a dropdown for "Gender", two radio buttons for "Contact via Email" and "Contact via SMS", a date picker for "Birthday", and a toggle switch for "Does the client have a house?". To the right of the form is a detailed configuration panel for the "Income" field. It includes sections for Placeholder (Income), Type (number), Prefix (\$), Suffix (eg. suffix), Helpertext, Data source, Validators, Expressions (with a condition \${application.key} != 'TEST'), Hide, Disabled (set to '-'), and UI Action.

(!) INFO

For more details on how to configure a UI action, click [here](#).

Input styling

Icons

- **Icon Key** - the key associated in the Media library, select the icon from the [Media Library](#)
- **Icon Color** - select the color of the icon using the color picker

(!) INFO

When setting the color, the entire icon is filled with that color, the SVG's fill. Avoid changing colors for multicolor icons.

You have the option to enhance the Input element by incorporating two types of icons:

- **Left Icon:** You can include an icon on the left side of the Input element. This icon can serve as a visual cue or symbol associated with the input field's purpose or content.
- **Right Icon:** Same as left icon.

By utilizing these two types of icons, you can provide users with a more intuitive and visually appealing experience when interacting with the Input element.

The screenshot shows a form builder interface with the following details:

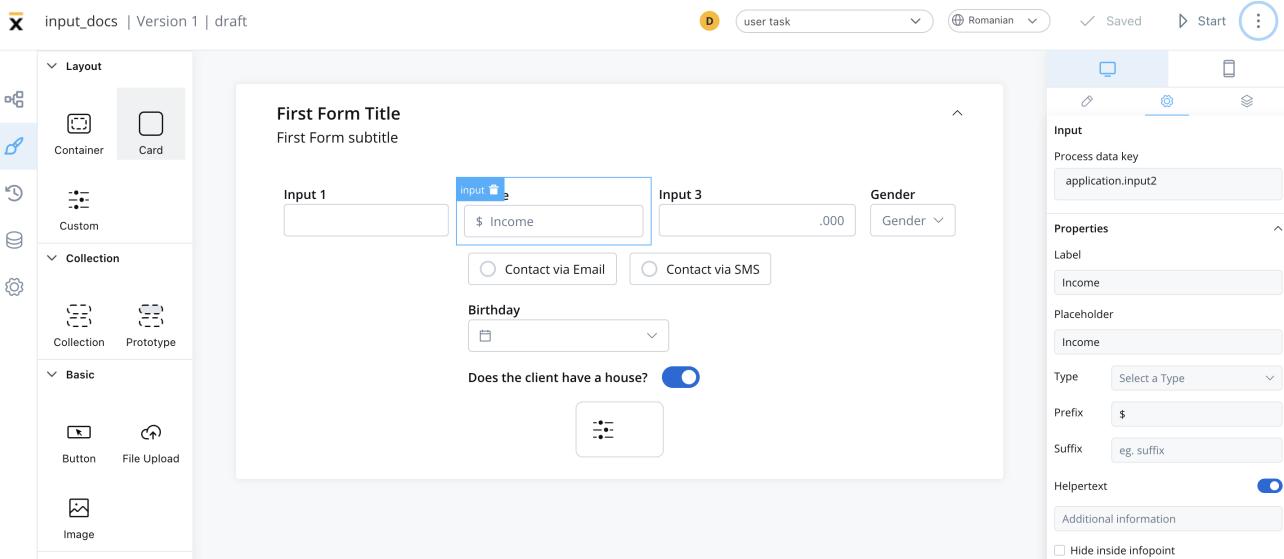
- Form Title:** Enter Personal Information
- Fields:**
 - First Name: Input field with a placeholder "First Name" and a green key icon.
 - Last Name: Input field with a placeholder "Last Name" and a green key icon.
 - Date of birth: Input field with a placeholder "Placeholder" and a green key icon.
 - Employment type:
 - Employed (radio button)
 - Pensioner (radio button)
 - Save personal information: A toggle switch.
 - Loan amount: A slider set to 255000 \$. The range is from 10000 \$ to 500000 \$.
 - Form title: An empty input field.
- Properties Sidebar:**
 - Input**: Left Icon is enabled (green key icon).
 - Properties**: Left Icon is enabled (green key icon). Right Icon is also enabled (green key icon).
 - Sizing**: Fit W is set to "fill".
 - Spacing**: Top, Bottom, Left, and Right values are all set to 0.
 - Typography**: Set color is available.

- The Input Field can be styled using valid CSS properties (more details [here](#))

The screenshot shows the FLOWX.AI UI Designer interface. On the left, there's a sidebar with categories: Layout (Container, Card, Custom), Collection (Collection, Prototype), and Basic (Button, File Upload, Image). The main area displays a form titled "First Form Title" with a subtitle "First Form subtitle". The form contains several input fields: a text input for "Income" with a placeholder "\$ Income", a text input for "Mortgage" with a placeholder ".000", a dropdown for "Gender", two radio buttons for "Contact via Email" and "Contact via SMS", a date input for "Birthday", and a toggle switch for "Does the client have a house?". To the right of the form is a panel with tabs for desktop and mobile viewports. The "Input" tab is active, showing properties for the "Income" field: Process data key "application.input2", Label "Income", Placeholder "Income", Type "number", Prefix "\$", Suffix "eg. suffix", Helpertext (with a toggle switch), Datasource, and Validators. There are also tabs for Properties, Events, and Scripts.

Was this page helpful?

BUILDING BLOCKS / UI Designer / UI component types / Form elements / Text area



A text area is a form element used to capture multi-line input from users in a conversational interface. The text area component is typically used for longer inputs such as descriptions, comments, or feedback, providing users with more space to type their responses.

It is an important tool for creating intuitive and effective conversational interfaces that can collect and process large amounts of user input.

Configuring the text area element

Text area settings

The text area offers the following configuration options:

- **General**
- **Properties**
- **Datasource**
- **Validators**

- **Expressions**
- **UI actions**
- **Text area styling**

General

- **Process data key** - creates the binding between form element and process data, so it can be later used in [decisions](#), [business rules](#) or [integrations](#)

Properties

- **Label** - the label of the text area
- **Placeholder** - the placeholder text that appears in the text area
- **Helpertext** - additional information about the text area field (can be hidden inside an infopoint)

Datasource

The default value for the element can be configured here, this will autofill the text field when you will run the process.

Validators

There are multiple validators can be added to a text area element (more details [here](#)).

The screenshot shows the FLOWX.AI form builder interface. On the left, there's a sidebar with icons for Text, Link, Form, Input, Textarea, Select, Checkbox, Radio, Switch, Datepicker, and Message. The main area displays a form titled "Form title" with fields for Customer Name, Income, Gender, Contact via Email/SMS, Select a date (Date of birth dropdown), and a newsletter subscription toggle. Below these is a Textarea field containing "Some text here". At the bottom is a "Submit" button. To the right of the form, there's a detailed configuration panel for the Textarea field. It includes sections for "Process data key" (set to "textKey"), "Properties" (Label: "Input label", Placeholder: "Some text here", Helpertext: checked), "Datasource" (Default value: "eg. Name"), and "Validators" (with a placeholder "Add a validator").

Expressions

The text area's behavior can be defined using JavaScript expressions for hiding or disabling the element. The following properties can be configured for expressions:

- **Hide** - JavaScript expression used to hide the text area when it returns a truthy value
- **Disabled** - JavaScript expression used to disable the text area when it returns a truthy value

INFO

It's important to make sure that disabled fields have the same expression configured under the path expressions → hide.

The screenshot shows the FLOWX.AI UI builder interface. On the left, there's a sidebar with categories like Layout, Collection, and Basic, each containing icons for Container, Card, Custom, Collection, Prototype, Button, File Upload, and Image. The main area displays a form titled "First Form Title" with a subtitle "First Form subtitle". The form includes fields for "Income" (with a placeholder "\$ Income" and a value ".000"), "Mortgage" (with a dropdown menu), "Gender" (with a dropdown menu), "Contact via Email" (radio button), "Contact via SMS" (radio button), "Birthday" (date picker), and a toggle switch labeled "Does the client have a house?". To the right of the form is a detailed configuration panel for the "Income" field. It shows settings for Placeholder (Income), Type (number), Prefix (\$), Suffix (eg. suffix), Helpertext (checkbox), Datasource (dropdown), Validators (dropdown), and Expressions (checkbox). The Expressions section is expanded and contains the code \${application.key}=='TEST'. There are also sections for Hide and Disabled.

UI actions

UI actions can be added to the text area field to define its behavior and interactions.

- **Event** - possible value: CHANGE
- **Action Type** - select the action type

The screenshot shows the FLOWX.AI UI Designer interface. On the left, there's a sidebar with various icons for creating different UI components like Text, Link, Form, Input, Textarea, Select, Checkbox, Radio, Switch, Datepicker, and Indicators. The main area displays a form titled "Form title". The form contains fields for "Customer Name", "Income", and "Gender". It also includes two radio buttons for "Contact via Email" and "Contact via SMS", and a dropdown for "Select a date" with "Date of birth" as the option. A toggle switch is set to "Do you want to subscribe to our newsletter?". Below these is a large text area with placeholder text "Some text here". At the bottom is a blue "Submit" button. To the right of the form, there are several configuration panels: "Helpertext" (disabled), "Datasource" (disabled), "Default value" (set to "eg. Name"), "Validators" (button to "Add a validator"), "Expressions" (with a red box around the "Hide" section containing the condition "\$ (textKey) !== 'TEST'"), "Disabled" (set to "-"), and "UI Action" (button to "Add UI action").

INFO

For more details on how to configure a UI action, click [here](#).

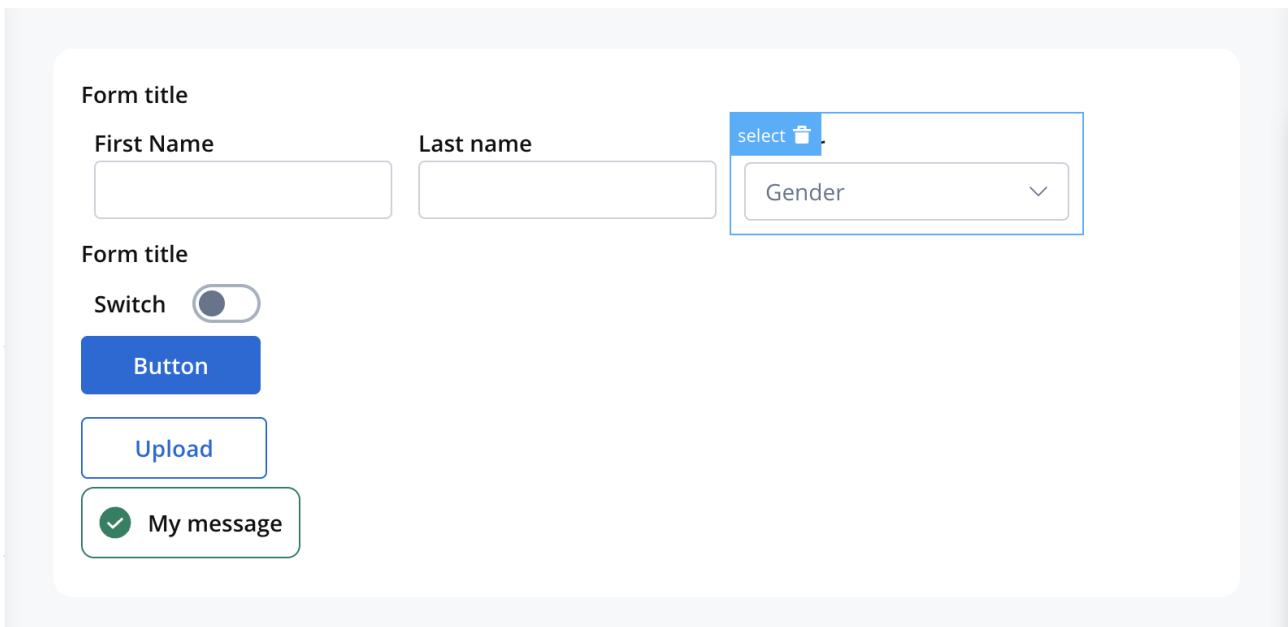
Text area styling

The ability to style the text area element using CSS properties is relevant because it allows you to customize the appearance of the text area to match the overall design of the website or application.

» [UI Designer styling](#)

Was this page helpful?

BUILDING BLOCKS / UI Designer / UI component types / Form elements / Select



The Select form field is an element that enables users to make a choice from a list of predefined options. It consists of multiple values, each of which is defined by a label that is displayed in the dropdown menu, and a code that is saved.

(!) INFO

For instance, you could have a label of "Female" with the value "F" and "Male" with the value "M". This means that when a user selects "Female" in the process instance, the value "F" will be stored for the "Select" key.

Configuring the Select element

Select Settings

These allow you to customize the settings for the Select Field:

- **General**
- **Properties**
- **Datasource**
- **Validators**
- **Expressions**
- **UI actions**
- **Select styling**

General

- **Process data key** - creates the binding between form element and process data so it can be later used in **decisions**, **business rules** or **integrations**

Properties

- **Label** - the label of the select
- **Placeholder** - placeholder when the field has no value
- **Empty message** - text displayed for custom type when no results are found
- **Search for options** - displays a search to filter options
- **Helpertext** - additional information about the select field (can be hidden inside an infopoint)

Datasource

- **Default value** - autofill the select with this value. Going back to the example with Woman label with F value and Man with M to have a default value of Woman we need to configure here F
- **Source Type** - it can be Static, Enumeration, or Process Data
- **Add option** - label - value pairs can be defined here

Validators

There are multiple validators can be added to a select (more details [here](#)).

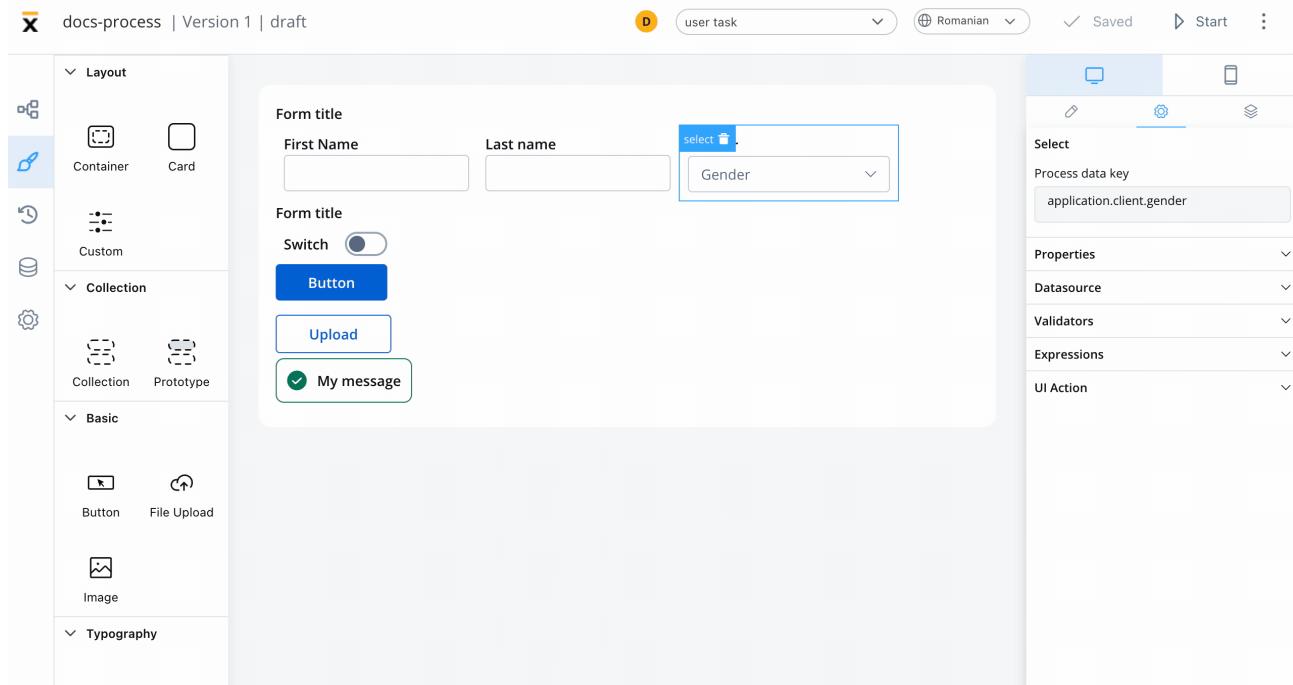
Expressions

The select field's behavior can be defined using JavaScript expressions for hiding or disabling the element. The following properties can be configured for expressions:

- **Hide** - JavaScript expression used to hide the Select Field when it returns a truthy value
- **Disabled** - JavaScript expression used to disable the Select Field when it returns a truthy value

(!) INFO

It's important to make sure that disabled fields have the same expression configured under the path expressions → hide.



UI actions

UI actions can be added to the select element to define its behavior and interactions.

- **Event** - possible value: `CHANGE`
- **Action Type** - select the action type

INFO

For more details on how to configure a UI action, click [here](#).

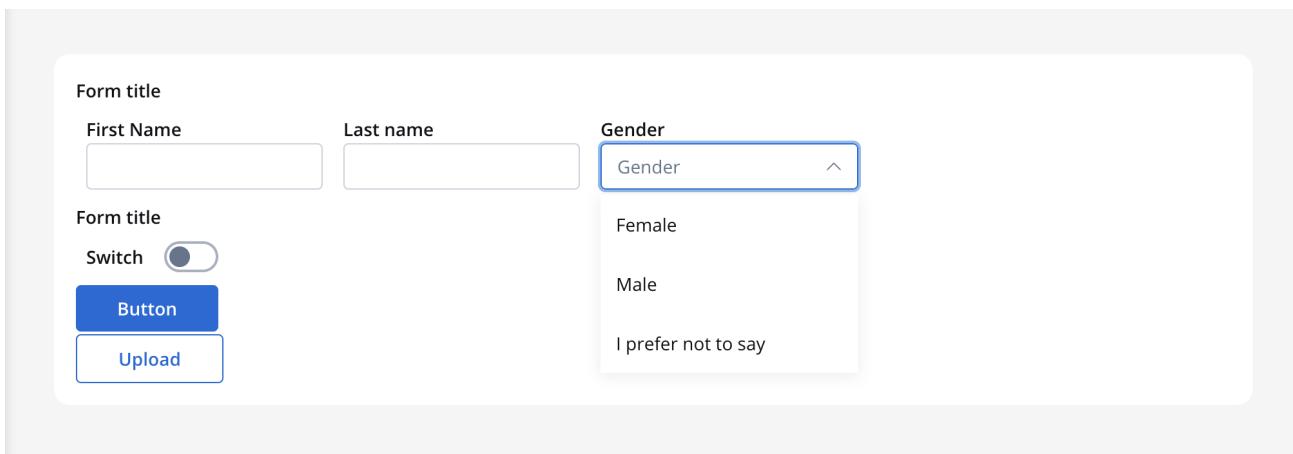
Select styling

Styling the Select field using CSS properties allows you to customize the appearance of the dropdown list and make it more visually appealing and

consistent with the overall design of the website or application.

» UI Designer styling

For example, a FORM element with a **layout** configuration including direction of Horizontal and some inputs, and a select element will look like this:



Icons

When customizing the appearance of a Select UI element that includes an icon, you can utilize the following properties:

- **Icon Key** - the key associated in the Media library, select the icon from the **Media Library**
- **Icon Color** - select the color of the icon using the color picker

(!) INFO

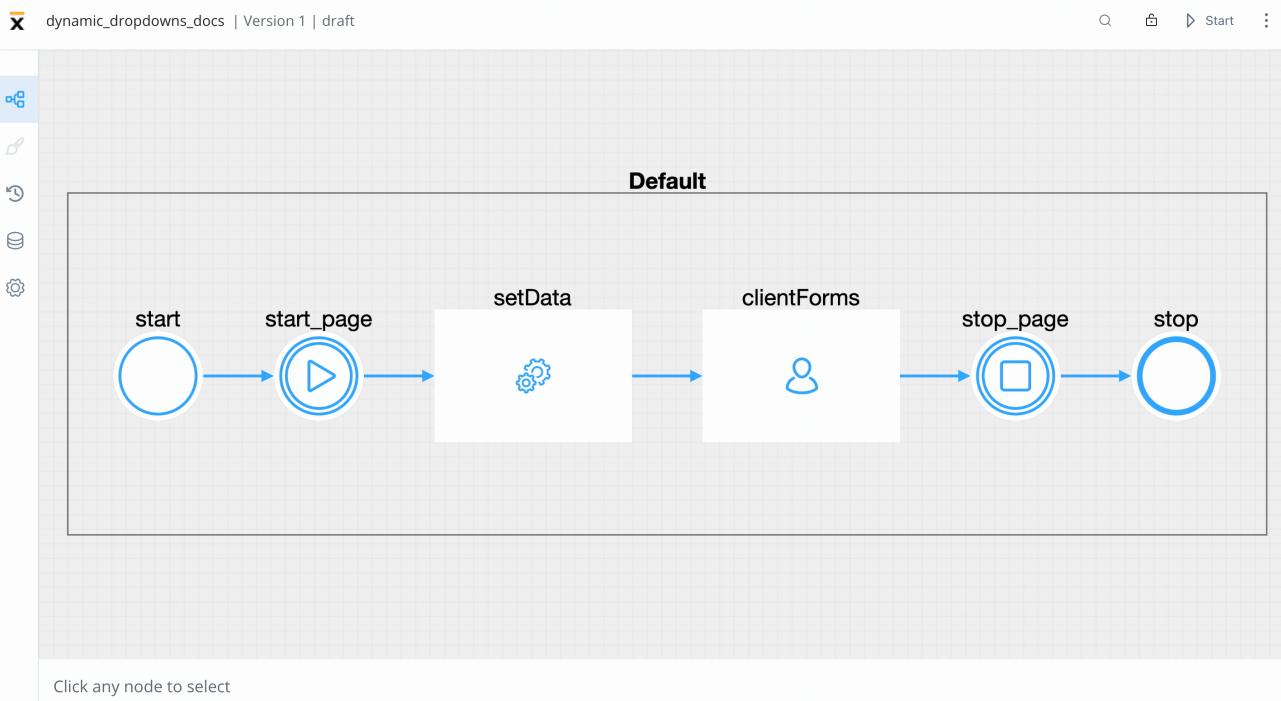
When setting the color, the entire icon is filled with that color, the SVG's fill. Avoid changing colors for multicolor icons.

Example - Dynamic dropdowns

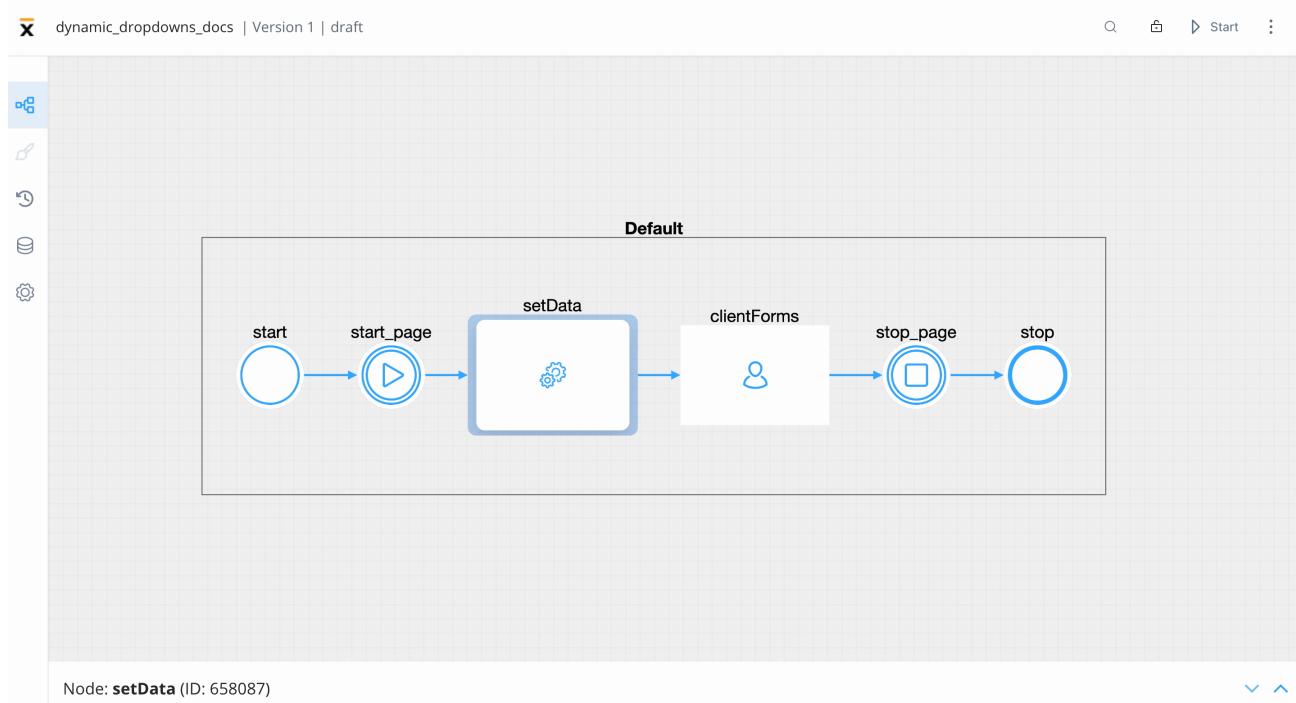
As mentioned previously, you can create dropdowns including static data, enumerations, or **process data**. Let's create an example using **process data** to create a process that contains **dynamic dropdowns**.

To create this kind of process, we need the following elements:

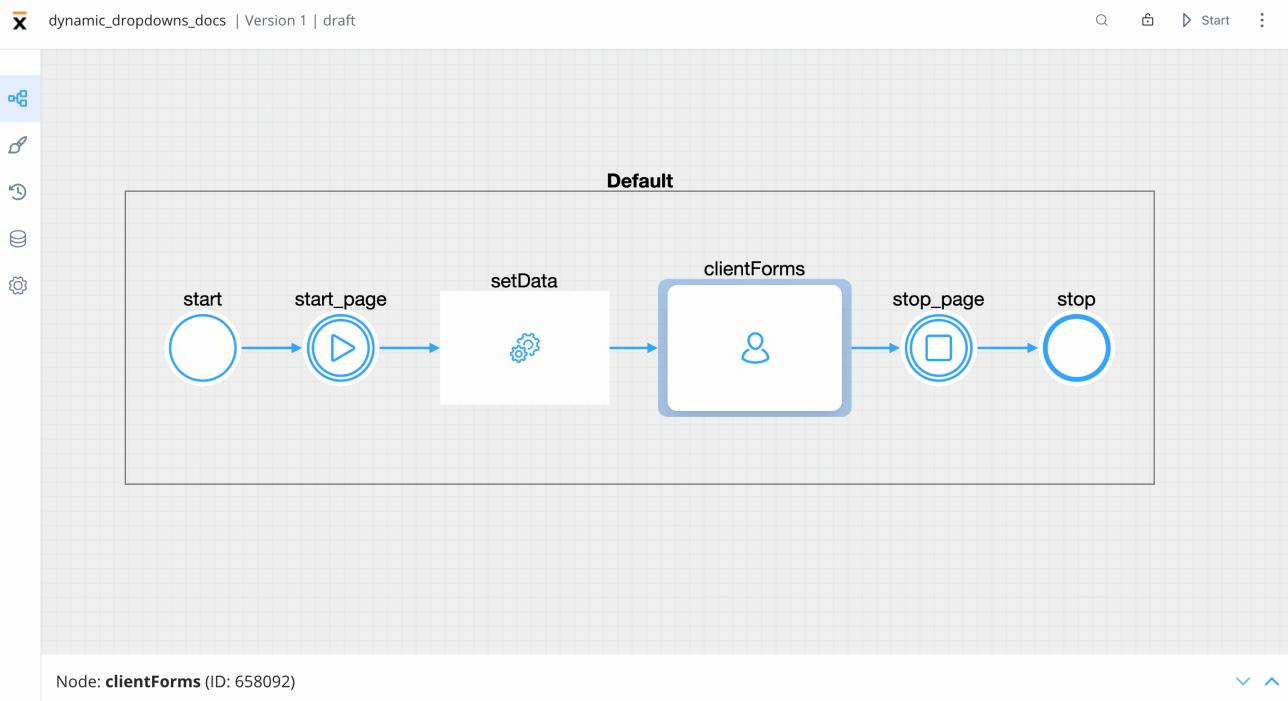
- a **start** node and an **end** node
- a **start milestone** UI element to it and an **end milestone** node



- a **task node** (this will be used to set which data will be displayed on the dropdowns)



- a **user task node** (here we have the client forms and here we add the SELECT elements)



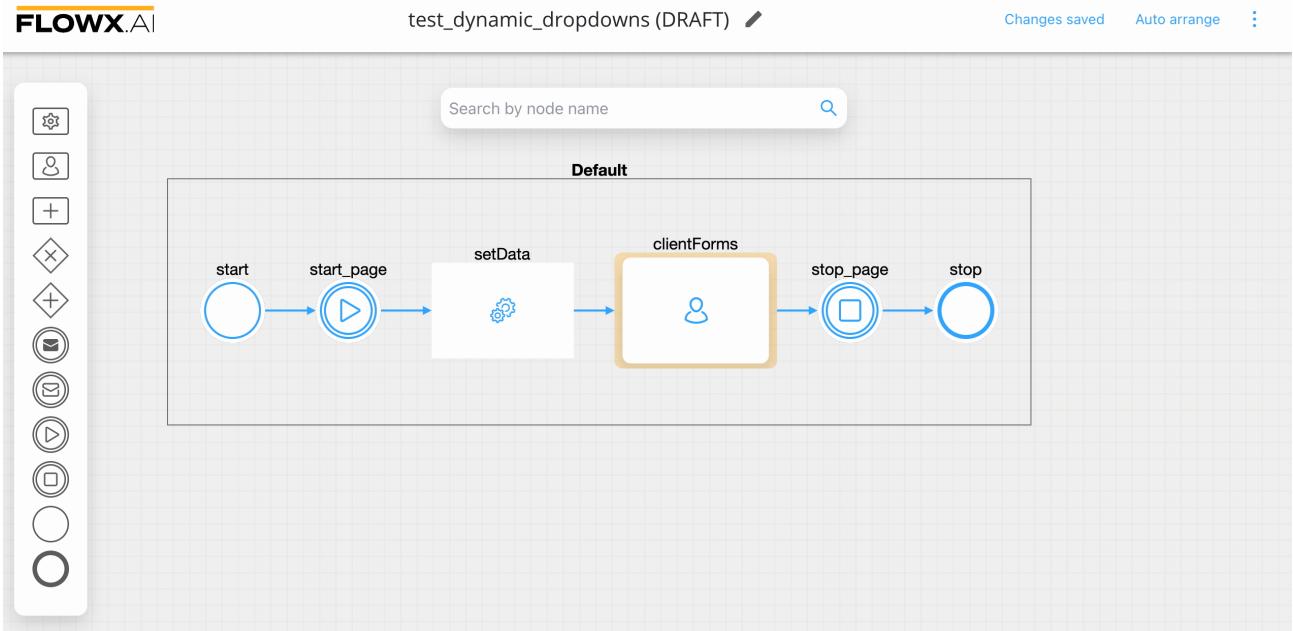
Creating the process

Follow the next steps to create the process from scratch:

1. Open **FLOWX Designer** and from the **Processes** tab select **Definitions**.
2. Click on the breadcrumbs (top-right corner) then click **New process** (the Process Designer will now open).
3. Now add all the **necessary nodes** (as mentioned above).

Configuring the nodes

1. On the **start milestone** node, add a **page** UI element.
2. On the **task node**, add a new **Action** (this will set the data for the dropdowns) with the following properties:
 - Action type - **Business Rule**
 - **Automatic**
 - **Mandatory**
 - **Language** (we used an **MVEL** script to create a list of objects)
3. On the **user task node**, add a new **Action** (submit action, this will validate the forms and save the date) with the following properties:
 - **Action type** - Save Data
 - **Manual**
 - **Mandatory**
 - **Data to send** (the key where the data will be sent) - **application**



Below you can find the MVEL script used in the above example:

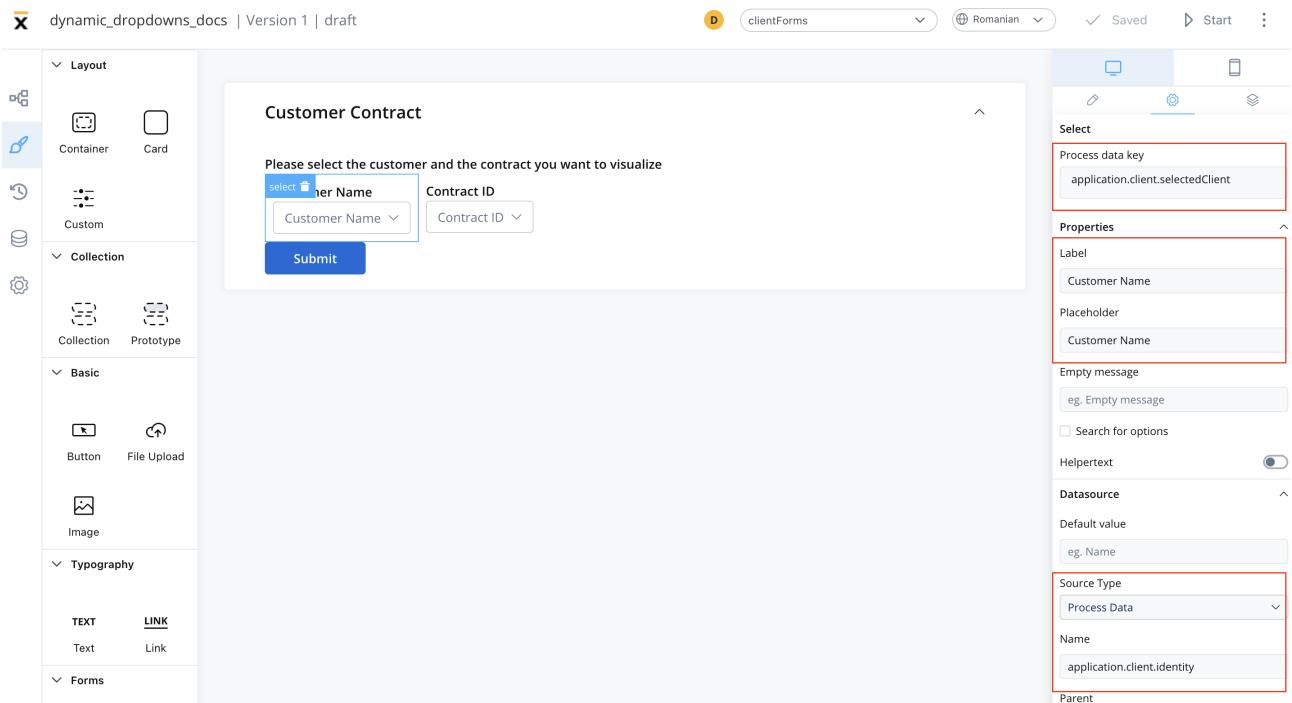
```
output.put("application",
{
    "client": {
        "identity": [
            {
                "value": "001",
                "label": "Eddard Stark"
            },
            {
                "value": "002",
                "label": "Sansa Stark"
            },
            {
                "value": "003",
                "label": "Catelyn Stark"
            }
        ]
    },
}
```

```
"contracts": {
    "001": [
        {
            "value": "c001",
            "label": "Eddard Contract 1"
        },
        {
            "value": "c007",
            "label": "Eddard Contract 2"
        }
    ],
    "003": [
        {
            "value": "c002",
            "label": "Catelyn Contract 1",
        },
        {
            "value": "c003",
            "label": "Catelyn Contract 2",
        },
        {
            "value": "c004",
            "label": "Catelyn Contract 3"
        }
    ],
    "002": [
        {
            "value": "c005",
            "label": "Sansa Contract 1",
        }
    ]
});
```

Configuring the UI

Follow the next steps to configure the UI needed:

1. Select the **user task node** and click the **brush icon** to open **UI Designer**
2. Add a **card** element as a **root component** (this will group the other elements inside it) with the following properties:
 - **Message** - `{"application": ${application}}`
 - **Title** - *Customer Contract*
3. Inside the **card**, add a **form element**.
4. Inside the **form** add two **select elements**, first will represent, for example, the *Customer Name* and the second the *Contract ID*.
5. For first select element (Customer Name) set the following properties:
 - **Process data key** - `application.client.selectedClient`
 - **Label** - Customer Name
 - **Placeholder** - Customer Name
 - **Source type** - Process Data (to extract the data added in the **task node**)
 - **Name** - `application.client.identity`



6. For the second select element (Contract ID) set the following properties:

- o **Process data key** - `application.client.selectedContract`
- o **Label** - Contract ID
- o **Placeholder** - Contract ID
- o **Source Type** - Process Data
- o **Name** - `application.contracts`
- o **Parent** - `SELECT` (choose from the dropdown list)

The screenshot shows the FLOWX.AI clientForms interface. On the left is a sidebar with categories like Layout, Collection, Basic, and Typography, each containing various UI element icons. The main area displays a form titled "Customer Contract" with the instruction "Please select the customer and the contract you want to visualize". It contains two dropdown menus: "Customer Name" and "Contract ID", both with a placeholder "Customer Name" and "Contract ID" respectively. Below the dropdowns is a blue "Submit" button. To the right of the form is a configuration panel for the "Contract ID" dropdown. The "Select" section is expanded, showing the process data key "application.client.selectedContract". The "Source Type" section is also highlighted with a red box, showing it is set to "Process Data". Other sections visible include "Properties", "Datasource", "Default value", "Validators", "Expressions", and "Hide".

7. Add a button under the form that contains the select elements with the following properties:

- **Label** - Submit
- **Add UI action** - add the submit action attached earlier to the user task node

dynamic_dropdowns_docs | Version 1 | draft

clientForms Romanian Saved Start :

Customer Contract

Please select the customer and the contract you want to visualize

Customer Name Contract ID

mit

Properties

Label: Submit

UI Action

Add action

Event: CLICK

Action Type: ACTION

Node Action Name: submit

Use a different name for UI action

Add custom body

Add form to validate

Dismiss process?

Show loader?

8. Test and run the process by clicking **Start process**.

dynamic_dropdowns_docs | Version 1 | draft

Default

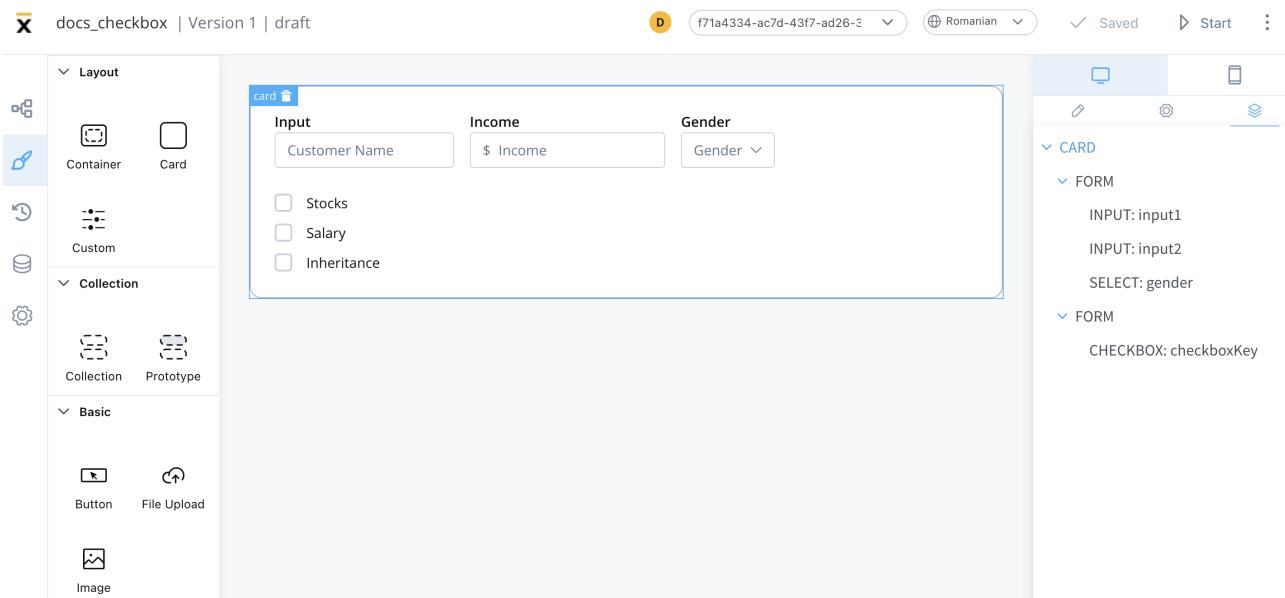
```

graph LR
    start((start)) --> startPage((start_page))
    startPage --> setData[setData]
    setData --> clientForms[clientForms]
    clientForms --> stopPage((stop_page))
    stopPage --> stop((stop))
  
```

Click any node to select

Was this page helpful?

BUILDING BLOCKS / UI Designer / UI component types / Form elements / Checkbox



A checkbox form field is an interactive element in a web form that provides users with multiple selectable options. It allows users to choose one or more options from a pre-determined set by simply checking the corresponding checkboxes.

This type of form field can be used to gather information such as interests, preferences, or approvals, and it provides a simple and intuitive way for users to interact with a form.

Configuring the checkbox element

Checkbox settings

The available configuration options for this form element are:

- [General](#)
- [Properties](#)
- [Datasource](#)
- [Validators](#)
- [Expressions](#)
- [UI actions](#)
- [Checkbox styling](#)

General

- **Process data key** - creates the binding between form element and process data, so it can be later used in [decisions](#), [business rules](#) or [integrations](#)

Properties

- **Label** - the label that appears on the checkbox
- **Helpertext** - additional information about the checkbox (can be hidden inside an infopoint)

The screenshot shows the FLOWX.AI platform's form builder. On the left, there's a sidebar with various UI component icons grouped under categories like Layout, Collection, Basic, and Data. The main area displays a form with three input fields: 'Customer Name' (text), 'Income' (dropdown), and 'Gender' (dropdown). Below these is a section titled 'checkbox' containing three checkboxes labeled 'Stocks', 'Salary', and 'Inheritance'. To the right of the form is a detailed panel for the 'checkbox' component, showing settings for 'Process data key' (checkboxKey), 'Label' (eg. Name), 'Helpertext' (Select an income source), and a checked 'Hide inside infopoint' option. There are also sections for 'Datasource', 'Validators', 'Expressions', and 'UI Action'.

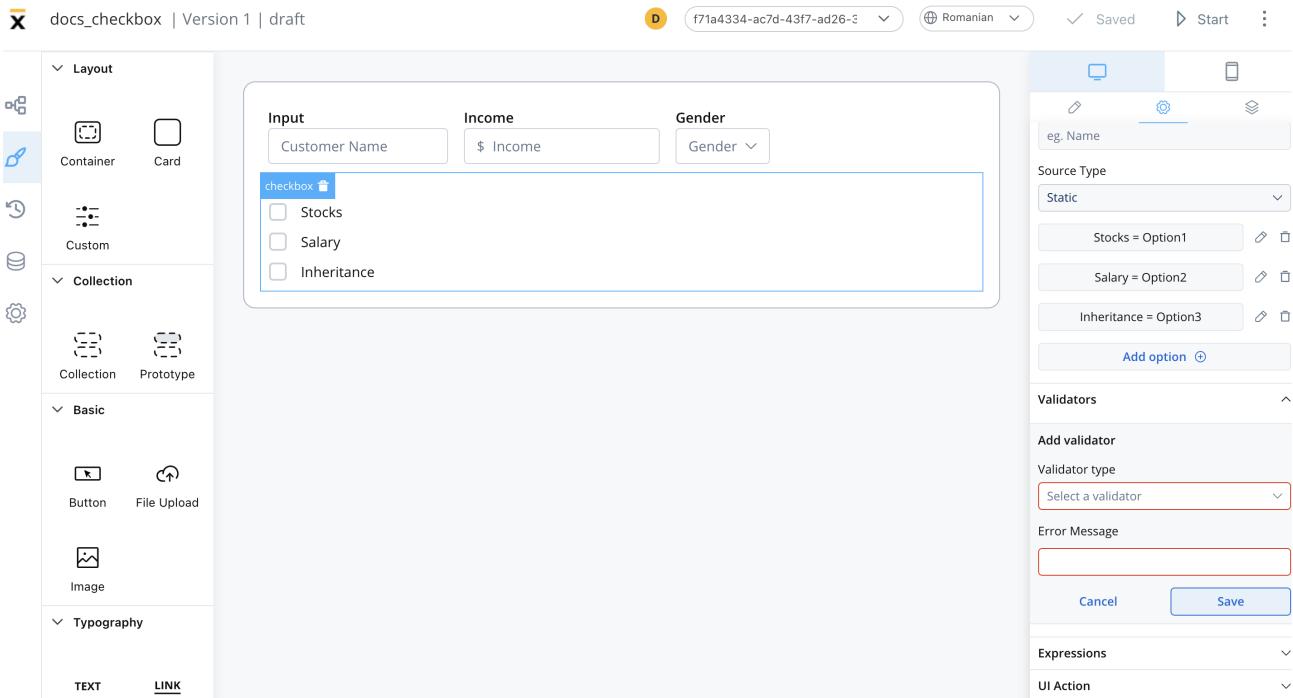
Datasource

- **Default Value** - the default value of the checkbox
- **Source Type** - it can be Static, Enumeration, or Process Data
- **Add option** - label - value pairs can be defined here

The screenshot shows the FLOWX.AI platform's form builder. On the left, there is a sidebar with various UI component icons categorized under 'Layout' (Container, Card), 'Collection' (Collection, Prototype), and 'Basic' (Button, File Upload, Image). The main workspace displays a form card with three input fields: 'Customer Name', '\$ Income', and 'Gender'. Below these is a 'checkbox' field containing three options: 'Stocks', 'Salary', and 'Inheritance'. To the right of the form is a 'Properties' panel. The 'Datasource' section is highlighted with a red box and contains three static options: 'Stocks = Option1', 'Salary = Option2', and 'Inheritance = Option3'. There is also a 'Add option' button. Other sections in the properties panel include 'Default value', 'Source Type' (set to 'Static'), 'Validators', 'Expressions', and 'UI Action'.

Validators

The following validators can be added to a checkbox: `required` and `custom` (more details [here](#)).



Expressions

The checkbox behavior can be defined using JavaScript expressions for hiding or disabling the element. The following properties can be configured for expressions:

- **Hide** - JavaScript expression used to hide the checkbox when it returns a truthy value
- **Disabled** - JavaScript expression used to disable the checkbox when it returns a truthy value

INFO

It's important to make sure that disabled fields have the same expression configured under the path expressions → hide.

UI actions

UI actions can be added to the checkbox element to define its behavior and interactions.

- **Event** - possible value: `CHANGE`
- **Action Type** - select the action type

! **INFO**

For more details on how to configure a UI action, click [here](#).

Checkbox styling

The type of the checkbox can be selected by using the **styling** tab in **UI Designer**, possible values:

- clear
- bordered

! **INFO**

For more valid CSS properties, click [here](#).

A clear checkbox element with three options added, and a column layout will look like as it follows:

✓ First Form Title

First Form subtitle

Input 1

\$ Income

Input 3

.000

Gender

▼

- Stocks
- Salary
- Inheritance

Was this page helpful?

BUILDING BLOCKS / UI Designer / UI component types / Form elements / Radio

The screenshot shows the FLOWX.AI platform's UI Designer. On the left, there's a sidebar with navigation links for Processes (Definitions, Active process, Process Instances, Failed process start) and Content Management (Enumerations, Substitution tags, Content models, Languages, Source systems, Media Library). The main area displays a form titled 'Form title'. The form contains fields for 'Customer Name' (input), 'Income' (input with a dollar sign prefix), 'Gender' (dropdown menu), and two radio button options: 'Contact via Email' and 'Contact via SMS'. The 'Contact via SMS' option is selected, indicated by a blue border around its radio button. A large blue 'Submit' button is at the bottom of the form.

Radio buttons are normally presented in radio groups (a collection of radio buttons describing a set of related options). Only one radio button in a group can be selected at the same time.

Configuring the radio field element

Radio settings

The available configuration options for this form element are:

- General
- Properties
- Datasource
- Validators
- Expressions
- UI actions
- Radio styling

General

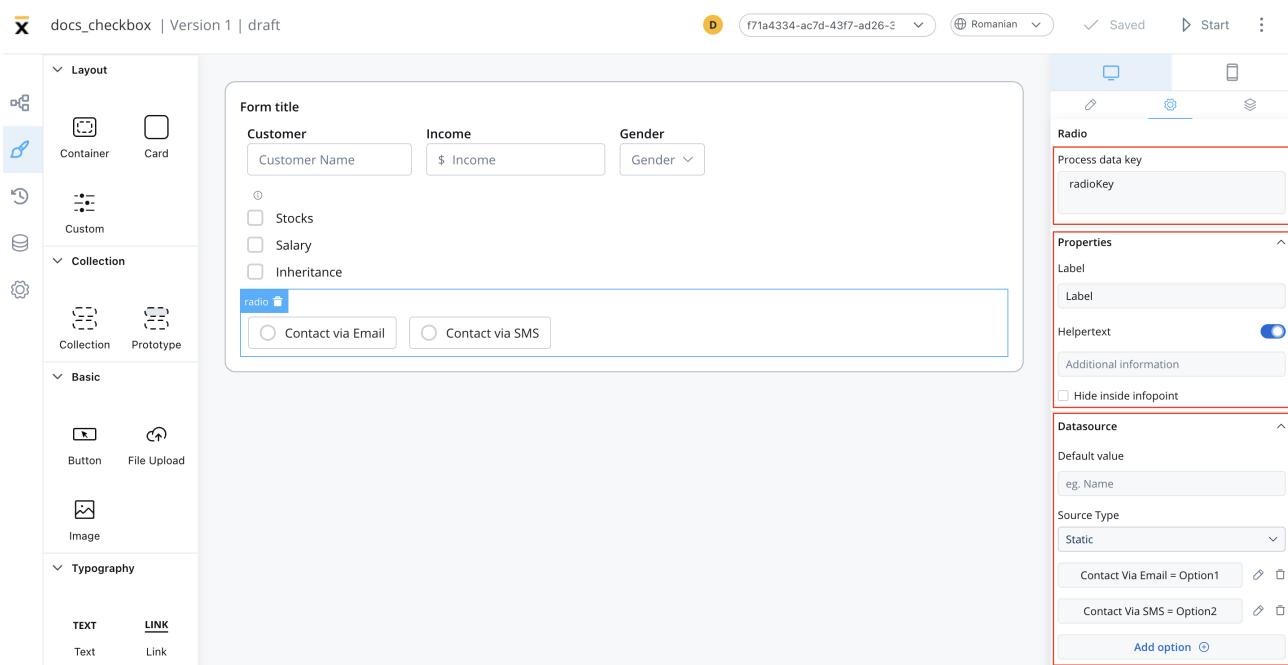
- **Process data key** - creates the binding between form element and process data so it can be later used in decisions, business rules or integrations

Properties

- **Label** - the label that appears on the radio
- **Helpertext** - additional information about the radio (can be hidden inside an infopoint)

Datasource

- **Default Value** - the default values of the radio element
- **Source Type** - it can be Static, Enumeration, or Process Data
- **Add option** - label - value pairs can be defined here



Validators

The following validators can be added to a radio: `required` and `custom` (more details [here](#))

Expressions

The radio's element behavior can be defined using JavaScript expressions for hiding or disabling the element. The following properties can be configured for expressions:

- **Hide** - JavaScript expression used to hide the Radio element when it returns a truthy value
- **Disabled** - JavaScript expression used to disable the Radio element when it returns a truthy value

!(INFO)

It's important to make sure that disabled fields have the same expression configured under the path expressions → hide.

The screenshot illustrates the FLOWX.AI UI builder interface. On the left, there's a sidebar with categories like Layout, Collection, and Basic, each containing various UI component icons. The main workspace shows a form with several fields: 'Customer Name' (text input), 'Income' (text input), and 'Gender' (dropdown). Below these are three checkboxes: 'Stocks', 'Salary', and 'Inheritance'. At the bottom of the form are two radio buttons: 'Contact via Email' and 'Contact via SMS'. To the right, the properties panel is open for a 'Radio' component. It includes sections for 'Process data key' (set to 'radioKey'), 'Properties', 'Datasource', and 'Validators'. A red box highlights the 'Validators' section, which contains a 'Required' validator and a button to 'Add a validator'. Another red box highlights the 'Expressions' section, which contains fields for 'Hide' and 'Disabled', both currently empty. The top of the screen shows the document title 'docs_checkbox | Version 1 | draft', a save status indicator, and language settings.

UI actions

UI actions can be added to the radio element to define its behavior and interactions.

- **Event** - possible value: CHANGE
- **Action Type** - select the action type

! INFO

For more details on how to configure a UI action, click [here](#).

Radio styling

The type of the radio can be selected by using the **styling** tab in **UI Designer**, possible values:

- clear
- bordered

! INFO

For more valid CSS properties, click [here](#).

A Radio element with two options added, and with a layout configuration set to horizontal will look like as it follows:

✓ First Form Title

First Form subtitle

The screenshot shows a UI Designer interface with a horizontal layout. At the top, there are three input fields: 'Input 1' (text), 'Income' (with a dollar sign icon and a dropdown arrow), and 'Input 3' (text). To the right of 'Input 3' is a field 'Gender' with a dropdown arrow. Below these are three radio buttons: 'Stocks' (unchecked), 'Salary' (checked), and 'Inheritance' (unchecked). At the bottom, there are two buttons: 'Contact via Email' (green background, checked) and 'Contact via SMS' (white background, unchecked).

Was this page helpful?

BUILDING BLOCKS / UI Designer / UI component types / Form elements / Switch

Form title

Customer Income Gender

Customer Customer \$ 999999999 Gender

Contact via Email Contact via SMS

Select a date

09.02.1972

Do you want to subscribe to our newsletter?

Submit

A switch, a toggle switch, is another form element that can be utilized to create an intuitive user interface. The switch allows users to select a response by toggling it between two states. Based on the selection made by the user, the corresponding Boolean value of either true or false will be recorded and stored in the process instance values for future reference.

Configuring the switch element

Switch settings

The available configuration options for this form element are:

- General
- Properties
- Datasource
- Validators
- Expressions
- UI actions
- Switch styling

General

- **Process data key** - creates the binding between form element and process data so it can be later used in decisions, business rules or integrations

Properties

- **Label** - the label of the switch
- **Helpertext** - additional information about the switch element (can be hidden inside an infopoint)

Datasource

- **Default Value** - the default value of the switch form field (it can be switched on or switched off)

Validators

The following validators can be added to a switch element: `requiredTrue` and `custom` (more details [here](#)).

The screenshot shows the FLOWX.AI UI builder interface. On the left is a sidebar with categories like Layout, Collection, Basic, and Typography, each containing icons for different UI elements. The main area displays a form titled "Form title". The form contains fields for "Customer Name", "Income", and "Gender", and two radio button groups for "Contact via Email" and "Contact via SMS". A "Select a date" dropdown is also present. Below these is a switch element labeled "want to subscribe to our newsletter?" with a blue toggle button. At the bottom is a "Submit" button. To the right of the form is a properties panel with tabs for "Properties", "Validators", "Expressions", and "UI Action". The "Properties" tab shows "Label: Do you want to subscribe to ou" and "Default value" checked. The "Validators" tab shows "Validator type: requiredTrue" and "custom". The "Expressions" and "UI Action" tabs are collapsed.

Expressions

- **Hide** - JavaScript expression used to hide the Switch element when it returns a truthy value
- **Disabled** - JavaScript expression used to disable the Switch element when it returns a truthy value

UI actions

UI actions can be added to the switch element to define its behavior and interactions.

- Event - possible value: CHANGE
- Action Type - select the action type



For more details on how to configure a UI action, click [here](#).

Switch styling

The label of the switch can be positioned either as `start` or `end`.

The screenshot shows the FLOWX AI UI builder interface. On the left, there's a sidebar with categories like Layout, Custom, Collection, Basic, and Media. The main area displays a form titled "Form title". The form includes fields for "Customer Name", "Income", and "Gender", and two radio buttons for "Contact via Email" and "Contact via SMS". Below these is a date picker labeled "Select a date" with "Date of birth". A switch component is present with the label "Do you want to subscribe to our newsletter?". The right side of the interface has a toolbar with icons for mobile devices and a "Switch" button, followed by a panel for "Properties" which includes sections for Label position (set to "end"), Sizing, Spacing, Background, Typography, and Advanced.

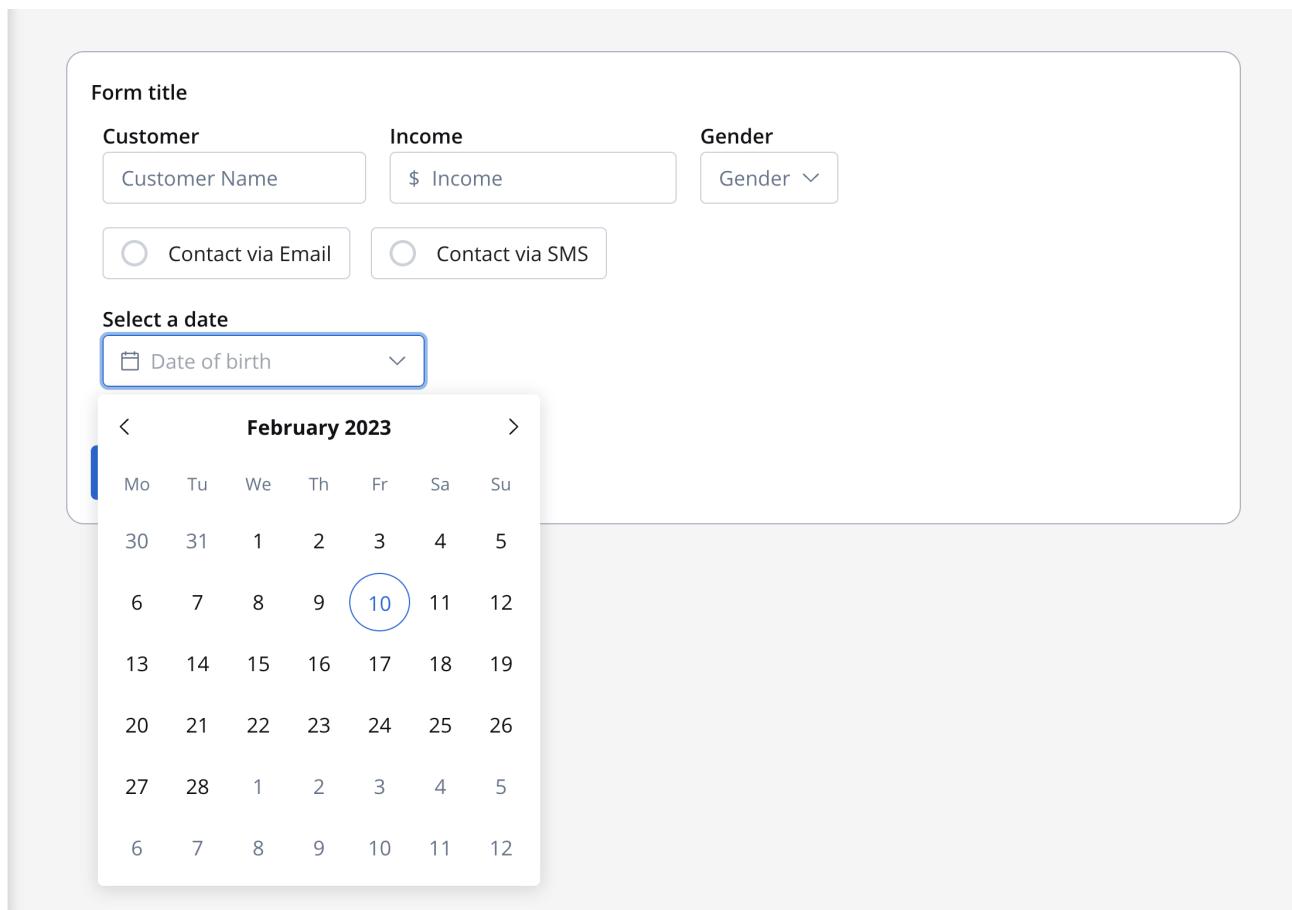
INFO

For more valid CSS properties, click [here](#).

Was this page helpful?

BUILDING BLOCKS / UI Designer / UI component types / Form elements /

Datepicker



The datepicker (Calendar Picker) is a lightweight component that allows end users to enter or select a date value.

!(INFO)

The default datepicker value is `DD.MM.YYYY`.

Configuring the datepicker element

Datepicker settings

The available configuration options for this form element are:

- **General**
- **Properties**
- **Datasource**
- **Validators**
- **Expressions**
- **UI actions**
- **Datepicker styling**

General

- **Process data key** - creates the binding between form element and process data so it can be later used in **decisions**, **business rules** or **integrations**

Properties

- **Label** - the label of the datepicker
- **Placeholder** - placeholder when the field has no value
- **Min Date** - set the minimum valid date selectable in the datepicker
- **Max Date** - set the maximum valid date selectable in the datepicker
- **Min Date, Max Date error** - when a date is introduced by typing, define the error message to be displayed
- **Helpertext** - additional information about the input field (can be hidden inside an infopoint)

The screenshot shows the FLOWX.AI process builder interface. On the left, there's a sidebar with various components like Layout, Collection, and Basic. The main area displays a form titled "Form title" with fields for "Customer Name", "Income", "Gender", and two radio buttons for "Contact via Email" or "Contact via SMS". Below these is a "datepicker" element with a placeholder "Date of birth". A "Submit" button is at the bottom. To the right of the form is a properties panel for the "datepicker" element, which includes fields for "Process data key" (set to "datepickerKey"), "Label" (set to "Select a date"), "Placeholder" (set to "Date of birth"), and date range settings for "Min date" and "Max date". There are also sections for "Min date error" and "Max date error". A "Helpertext" section is shown with a toggle switch.

Datasource

- **Default Value** - the default values of the datepicker element, this will autofocus the datepicker when you will run the process

Validators

The following validators can be added to a datepicker: `required`, `custom`, `isSameOrBeforeToday` or `isSameOrAfterToday` (more details [here](#)).

The screenshot shows the FLOWX.AI UI builder interface. On the left, there's a sidebar with various components like Container, Card, Custom, Collection, and Basic. In the center, there's a form titled "Form title" with fields for Customer Name, Income, Gender, Contact via Email, Contact via SMS, and a datepicker field labeled "Date of birth". A "Submit" button is at the bottom. On the right, there's a configuration panel for the datepicker component. It includes sections for "Process data key" (datepickerKey), "Properties" (Datasource, Default value: "eg. Name"), and "Validators" (Validator type dropdown set to "Select a validator", with options like "required", "isSameOrBeforeToday", "isSameOrAfterToday", and "custom"). The "Default value" and "Validators" sections are highlighted with a red border.

Expressions

The datepicker behavior can be defined using JavaScript expressions for hiding or disabling the element. The following properties can be configured for expressions:

- **Hide** - JavaScript expression used to hide the datepicker when it returns a truthy value
- **Disabled** - JavaScript expression used to disable the datepicker when it returns a truthy value

INFO

It's important to make sure that disabled fields have the same expression configured under the path expressions → hide.

UI actions

UI actions can be added to the datepicker element to define its behavior and interactions.

- **Event** - possible value: `CHANGE`
- **Action Type** - select the action type

INFO

For more details on how to configure a UI action, click [here](#).

The screenshot shows the FLOWX.AI platform interface. On the left, there's a sidebar with icons for Layout (Container, Card, Custom, Collection, Prototype), Basic (Button, File Upload, Image), and a search bar. The main area displays a form titled "Form title" with fields for "Customer Name", "Income", "Gender", "Contact via Email", "Contact via SMS", and a "datepicker" field for "Date of birth". A "Submit" button is at the bottom. To the right, there's a configuration panel with tabs for Desktop, Mobile, and Settings. The "Expressions" tab is open, showing sections for "Hide" and "Disabled". The "UI Action" tab is also open, showing "Event" set to "CHANGE" and "Action Type" set to "Select an action type". Buttons for "Cancel" and "Save" are at the bottom of this panel.

Datepicker styling

The styling of a datepicker element can be customized in various ways using CSS properties like typography color, border-radius/width, or advanced CSS params. This allows you to create a datepicker that fits seamlessly with the overall design of the application you are developing.

! INFO

For more valid CSS properties, click here.

Was this page helpful?

BUILDING BLOCKS / UI Designer / UI component types / Form elements / Slider

The screenshot shows a UI designer's preview of a form titled "Enter Personal Information". The form contains the following fields:

- Form title: Enter Personal Information
- First Name: John
- Last Name: Doe
- Date of birth: 18.12.1995
- Income range: Segmented button options include < \$50.000, \$50.000 - \$100.000, \$100.000 - \$150.000, and > \$150.000.
- Rate your experience with us: A horizontal slider with a scale from 1 to 5. The value is currently set to 1.
- Save personal information: A toggle switch that is turned on.
- Submit button: A blue rectangular button at the bottom of the form.

It allows users to pick only one option from a group of options, and you can choose to have between 2 and 5 options in the group. The segmented button is

easy to use, and can help make your application easier for people to use.

Configuring the slider element

Slider settings

The available configuration options for this form element are:

- **General**
- **Properties**
- **Datasource**
- **Validators**
- **Expressions**
- **UI actions**
- **Slider styling**

General

- **Process data key** - creates the binding between form element and process data so it can be later used in **decisions**, **business rules** or **integrations**

Properties

- **Label** - the label of the slider
- **Show value label** - a toggle option that determines whether the current selected value of the slider is displayed as a label alongside the slider handle
- **Helptext** - an optional field that provides additional information or guidance related to the usage or function of the slider, it can be hidden within an

infopoint, which users can expand or access for further detail

- **Min Value** - the minimum value or starting point of the slider's range, it defines the lowest selectable value on the slider
- **Max Value** - the maximum value or end point of the slider's range, it sets the highest selectable value on the slider
- **Suffix** - an optional text or symbol that can be added after the displayed value on the slider handle or value label, it is commonly used to provide context or units of measurement
- **Step size** - the increment or granularity by which the slider value changes when moved, it defines the specific intervals or steps at which the slider can be adjusted, allowing users to make more precise or discrete value selections

Datasource

- **Default Value** - the default value of the slider (static value - integer) the initial value set on the slider when it is first displayed or loaded, it represents a static value (integer), that serves as the starting point or pre-selected value for the slider, users can choose to keep the default value or adjust it as desired

Validators

The following validators can be added to a slider: `required` and `custom` (more details [here](#)).

UI actions

UI actions can be added to the slider element to define its behavior and interactions.

- **Event** - possible value: CHANGE
- **Action Type** - select the action type, ! for more details on how to configure a UI action, click [here](#)

Multiple sliders

You can also use multiple sliders UI elements that are interdependent, as you can see in the following example:

The screenshot shows a user interface for entering personal information. On the left, there's a sidebar with icons for close, edit, info, and refresh. The main area has a title 'Enter Personal Information'. It includes fields for First Name (placeholder 'First Name'), Last Name ('Doe'), and Date of birth ('18.12.1995'). Under 'Employment type', there are two radio buttons: 'Employed' and 'Pensioner'. A toggle switch is labeled 'Save personal information'. Below these are three slider components:

- Loan amount:** Range from 10000 \$ to 500000 \$. The value is set to 255000 \$.
- Down payment:** Range from 38250 \$ to 89250 \$. The value is set to 38250 \$.
- Loan type:** A dropdown menu is open, showing 'Conventional' as the selected option.

A large blue 'Submit' button is at the bottom.

INFO

You can improve the configuration of the slider using computed values as in the example above. These values provide a more flexible and powerful approach for handling complex use cases. You can find an example by referring to the following documentation:

Dynamic & computed values

Slider styling

To create a slider with specific styling, sizing, typography, and color settings, you can use the following configuration:

- **Sizing**
- **Typography**
- **Background**

Sizing

- set the width of the button - fill/fixed/auto

Typography

Choose an appropriate font family, size, and weight for the button label text.

- **Label Color** - set the color of the button label text
- **Min/Max values** - set the color of
- **Result** - set the color of the

Background

- **Selected BG** - set the background color of the button.
- **ComponentBg** - set the background color of the button.

The screenshot shows the FLOWX.AI UI Designer interface. On the left is a sidebar with categories like Layout, Forms, and Collections, each containing icons for different UI components. The main area displays a form titled "Enter Personal Information" with fields for First Name, Last Name, Date of birth, Employment type (radio buttons for Employed and Pensioner), Income range (a segmented button with options < \$50.000, \$50.000 - \$100.000, \$100.000 - \$150.000, and > \$150.000), and a Loan amount slider ranging from 10000 € to 100000 €. There's also a "Save personal information" toggle switch and a "Submit" button. To the right of the form is a detailed component settings panel for the "Slider" component, which includes sections for Sizing (Fit W: fill), Spacing (0 to 16), Typography (Label, Min/Max values, Result), Background (Selected BG, ComponentBg), and Advanced (Add class). The "Slider" section is highlighted with a red box.

INFO

For more valid CSS properties, click [here](#).

Was this page helpful?

BUILDING BLOCKS / UI Designer / UI component types / Form elements / Segmented button

The screenshot shows a 'Form title' section with three input fields: 'First Name' (John), 'Last Name' (Doe), and 'Date of birth' (18.12.1995). Below this is a 'Income range' section with four options: '< \$50.000', '\$50.000 - \$100.000', '\$100.000 - \$150.000', and '> \$150.000'. A 'Save personal information' toggle switch is turned on. At the bottom is a blue 'Submit' button.

It allows users to pick only one option from a group of options, and you can choose to have between 2 and 5 options in the group. The segmented button is easy to use, and can help make your application easier for people to use.

Configuring the segmented button

Segmented button settings

The available configuration options for this form element are:

- General
- Properties
- Datasource
- Validators
- Expressions

- **UI actions**
- **Segmented button styling**

General

- **Process data key** - creates the binding between form element and process data so it can be later used in **decisions**, **business rules** or **integrations**

Properties

- **Label** - the label of the segmented button
- **Helpertext** - additional information about the segmented button (can be hidden inside an infopoint)

Datasource

- **Default Value** - the default value of the segmented button (it can be selected from one of the static source values)
- **Source Type** - it is by default Static
- **Add option** - value/label pairs can be defined here

Validators

The following validators can be added to a segmented button: **required** and **custom** (more details [here](#)).

The screenshot shows the FLOWX.AI platform's form builder interface. On the left, a sidebar lists various UI components like Layout, Forms, and Collections. The main workspace displays a form titled "Enter Personal Information" with fields for First Name, Last Name, and Date of birth, followed by a Segmented Button for income ranges. The right side features a properties panel for the Segmented Button, with sections for Properties, Datasource, and Validators, all highlighted with red boxes.

UI actions

UI actions can be added to the segmented button element to define its behavior and interactions.

- **Event** - possible value: CHANGE
- **Action Type** - select the action type

INFO

For more details on how to configure a UI action, click [here](#).

Segmented button styling

To create a segmented button with specific styling, sizing, typography, and color settings, you can use the following configuration:

- **Sizing**
- **Typography**
- **Background**

Sizing

- set the width of the button - fill/fixed/auto

Typography

Choose an appropriate font family, size, and weight for the button label text.

- **Label Color** - set the color of the button label text
- **Selected State** - set the color of the label text when the button is selected
- **Unselected State** - set the color of the label text when the button is not selected

Background

- **Selected state** - set the background color of the button
- **Unselected state** - set the background color of the button

The screenshot shows the FLOWX.AI UI Designer interface. On the left is a sidebar with categories like Layout, Forms, and Collections, each containing various UI component icons. The main area displays a form titled "Enter Personal Information" with fields for First Name, Last Name, Date of birth, and an income range selector. To the right of the form is a detailed styling panel with sections for Sizing, Spacing, Typography, Background, Border, and Advanced settings. Red boxes highlight the "Sizing" and "Background" sections.

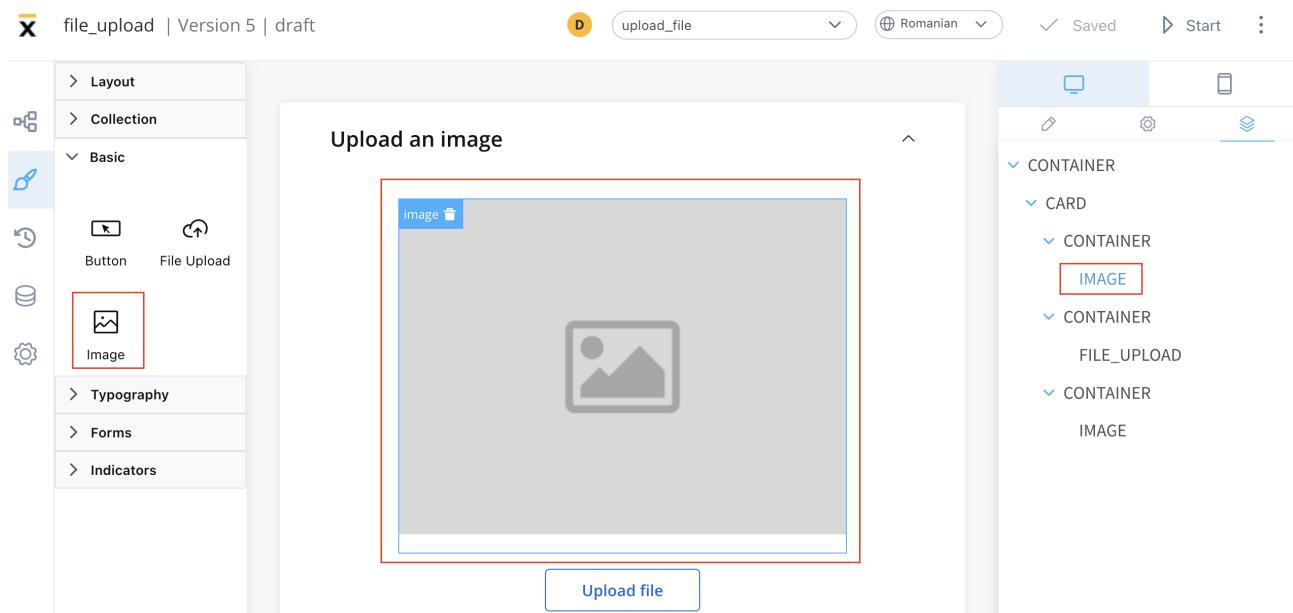
INFO

For more valid CSS properties, click [here](#).

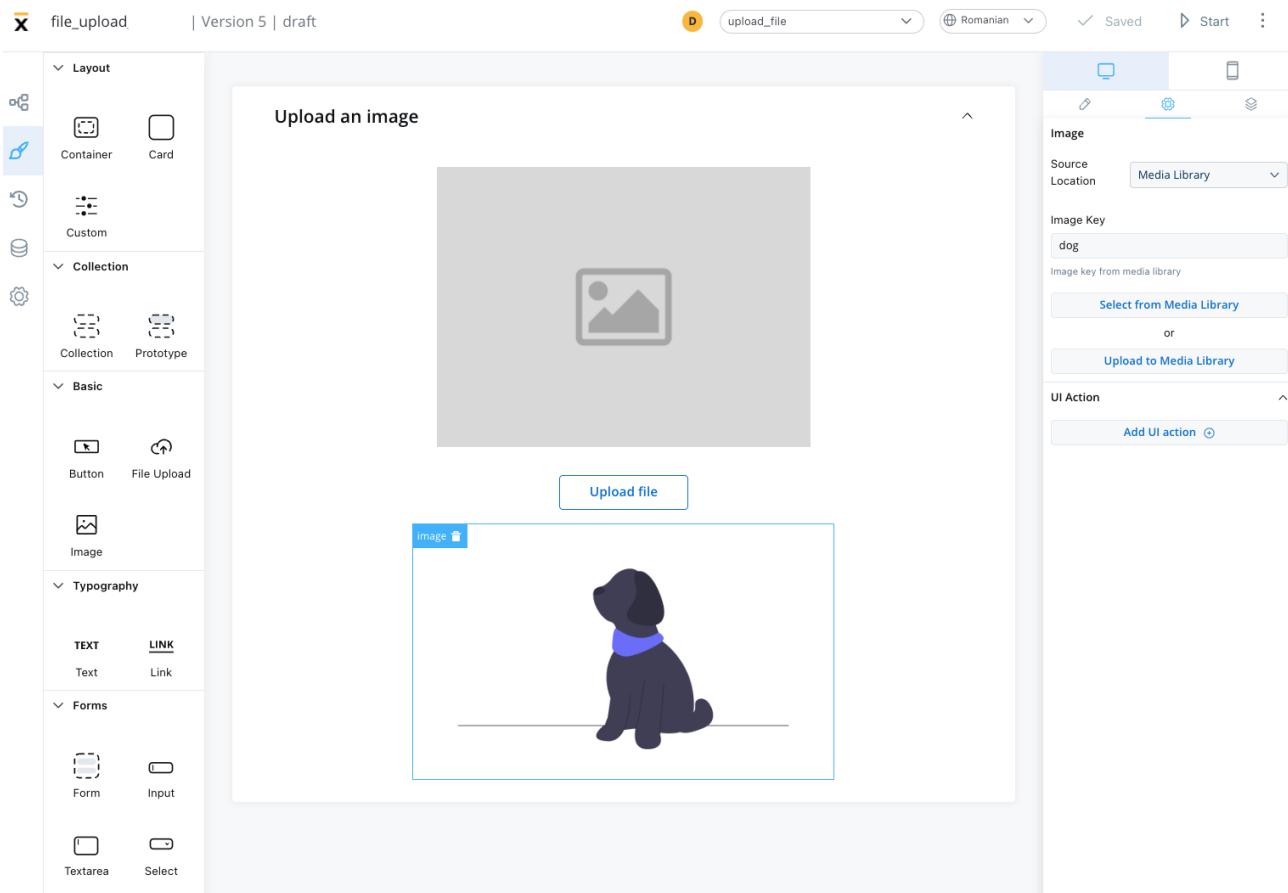
Was this page helpful?

BUILDING BLOCKS / UI Designer / UI component types / Image

Image UI elements are graphical components of a user interface that display a static or dynamic visual representation of an object, concept, or content.



These elements can be added to your interface using the UI Designer tool, and they are often used to convey information, enhance the aesthetic appeal of an interface, provide visual cues and feedback, support branding and marketing efforts, or present complex data or concepts in a more intuitive and accessible way.



Configuring an image

Configuring an image in the UI Designer involves specifying various settings and properties. Here are the key aspects of configuring an image:

Image settings

The image settings consist of the following properties:

- **Source location** - the location from where the image is loaded:
 - **Media Library**

- o **Process Data**
- o **External**

Depending on which **Source location** is selected, different configurations are available:

Media library

The screenshot shows the FLOWX.AI interface with the following components:

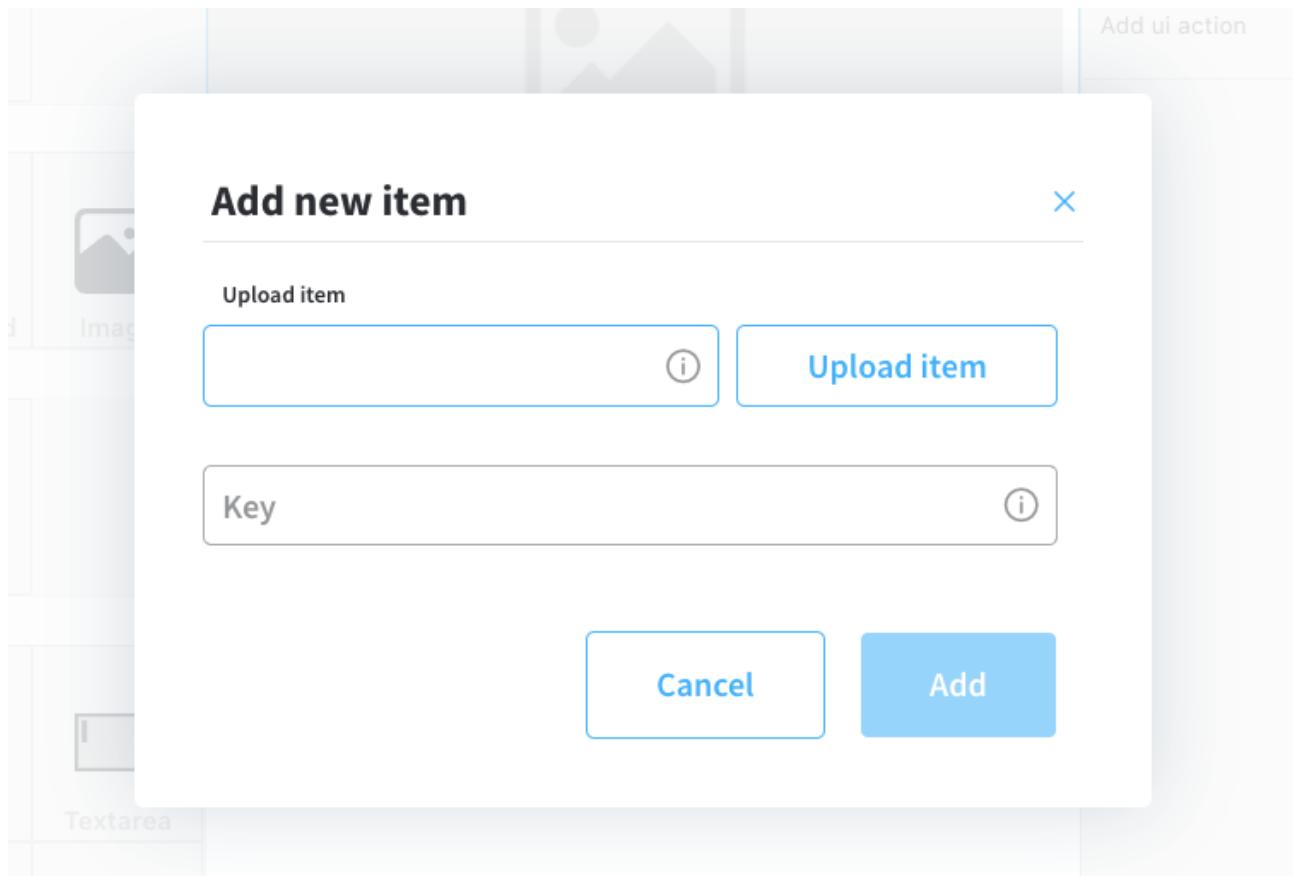
- Header:** file_upload | Version 5 | draft, upload_file, Romanian, Saved, Start, three-dot menu.
- Left Sidebar:** A tree view of available components under categories: Layout, Collection, Basic, Typography, and Forms. Under Basic, "Image" is selected, showing a preview of a black dog sitting.
- Middle Panel:** A card titled "Upload an image" with a placeholder image icon and a "Upload file" button below it.
- Right Panel:** A configuration panel for the "Image" component. It includes:
 - Image:** A preview area showing the black dog image.
 - Source Location:** A dropdown set to "Media Library".
 - Image Key:** A text input field containing "dog".
 - UI Action:** A section with a "Select from Media Library" button and a "Upload to Media Library" button.

- **Image key** - the key of the image from the media library

- **Select from media library** - search for an item by key and select it from the media library

Preview	Key	Format	Size	Edited at	Edited by	
<input type="checkbox"/>		video_file	png	20.58 KB	23 Feb 2023, 11:29 AM	John Doe
<input type="checkbox"/>		switches	png	0.03 MB	23 Feb 2023, 11:28 AM	John Doe
<input type="checkbox"/>		form_example	png	22.29 KB	23 Feb 2023, 11:27 AM	John Doe
<input type="checkbox"/>		avatar2	png	0.03 MB	23 Feb 2023, 11:26 AM	John Doe
<input type="checkbox"/>		avatar1	png	27.73 KB	23 Feb 2023, 11:26 AM	John Doe
<input type="checkbox"/>		cat	png	15.63 KB	22 Feb 2023, 1:18 PM	John Doe
<input type="checkbox"/>		dog	png	20.65 KB	22 Feb 2023, 1:18 PM	John Doe

- **Upload to media library** - add a new item (upload an image on the spot)
 - **upload item** - supported formats: PNG, JPG, GIF, SVG, WebP;
! (maximum size - 1 MB)
 - **key** - the key must be unique and cannot be changed afterwards



Process Data

The screenshot shows the FLOWX.AI platform interface. On the left, there is a sidebar with various UI element icons categorized into sections: Layout, Collection, Basic, Typography, and Forms. The 'Image' section under 'Basic' is currently selected, showing a placeholder image icon and a 'File Upload' button. In the center, a modal window titled 'Upload an image' is displayed, featuring a large input field labeled 'image' with a delete icon, a placeholder image icon, and a 'Upload file' button at the bottom. To the right of the modal, there is a configuration panel for the 'Image' component. It includes fields for 'Source Location' (set to 'Process Data'), 'Source Type' (set to 'URL'), 'Process Data Key' (set to 'app.image'), and a 'Placeholder Url' field containing 'Public url'. Below these, there is a 'UI Action' section with a 'Add UI action' button.

- Identify the **Source Type**. It can be either a **URL** or a **Base 64 string**.
- Locate the data using the **Process Data Key**.
- If using a URL, provide a **Placeholder URL** for public access. This is the URL where the image placeholder is available.

file_upload | Version 5 | draft D upload_file Romanian Saved Start :

Layout

- Container
- Card

Custom

Collection

- Collection
- Prototype

Basic

- Button
- File Upload

Image

Typography

- Text
- Link

Forms

Upload an image

Image Placeholder

Upload file

Dog Placeholder

Image

Source Location Process Data

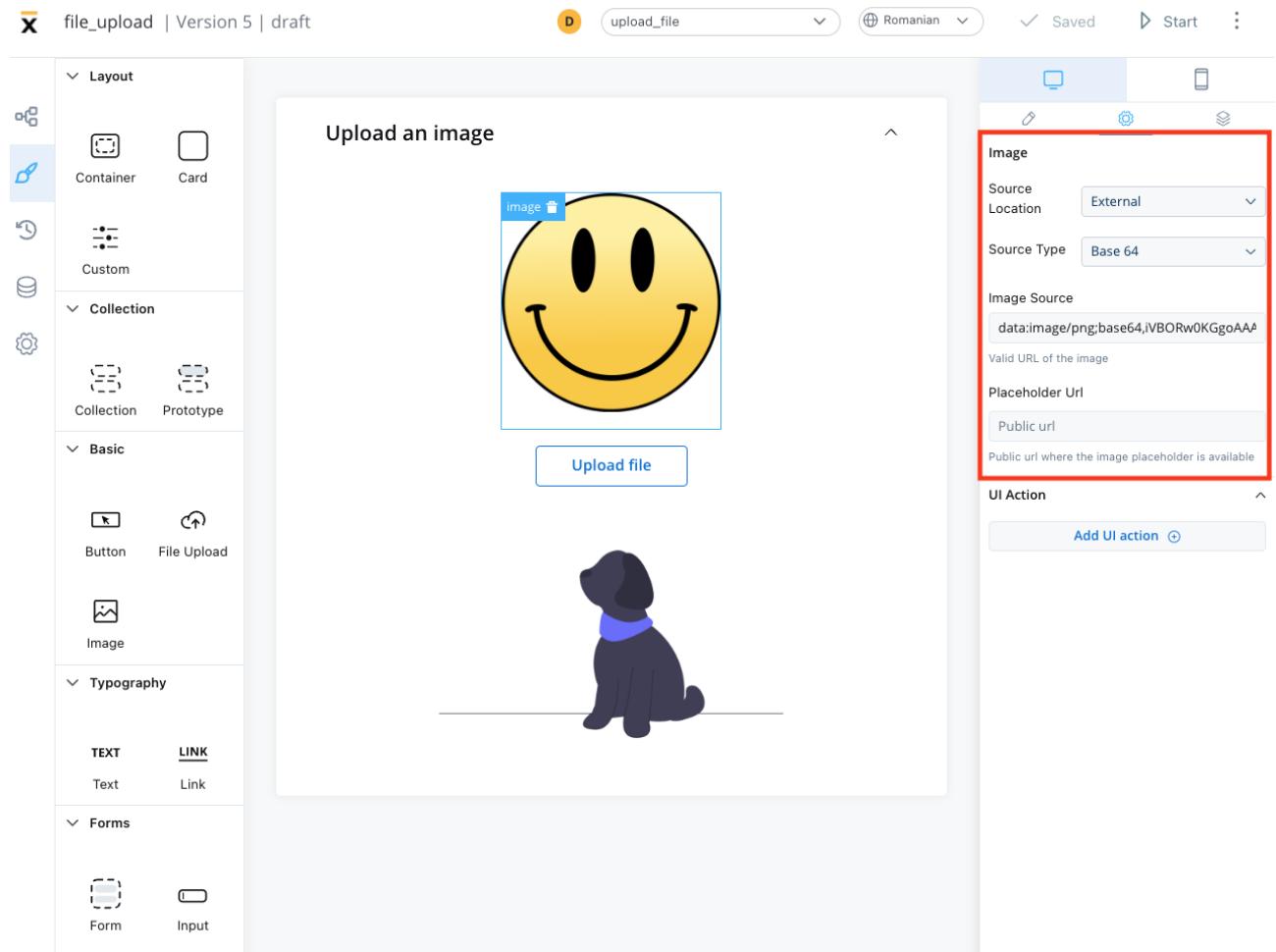
Source Type URL

Process Data Key app.image

Placeholder Url Public url

Add UI action

External



- **Source Type:** it can be either a **URL** or a **Base 64 string**
- **Image source:** the valid URL of the image.
- **Placeholder URL:** the public URL where the image placeholder is available

UI actions

The UI actions property allows you to add a UI Action, which must be configured on the same node. For more details on UI Actions, refer to the documentation [here](#).

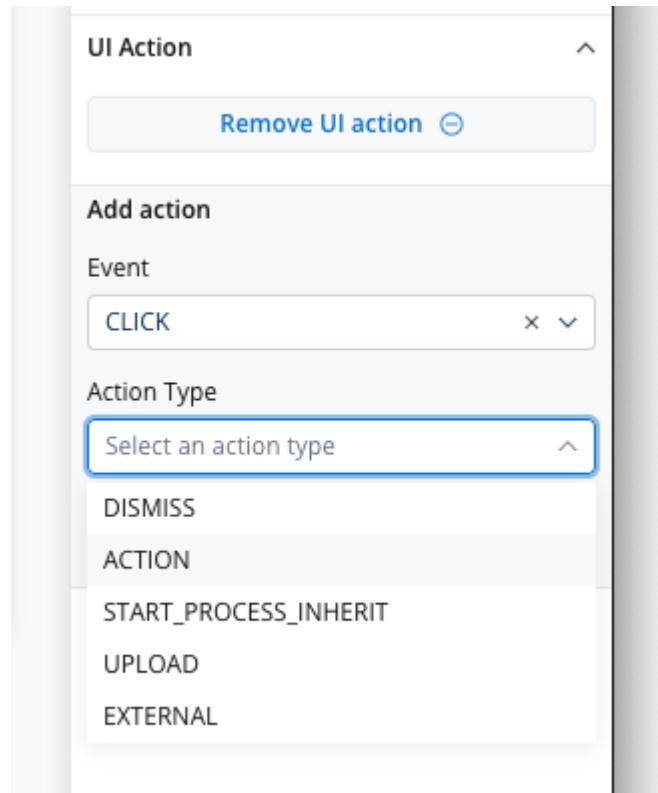


Image styling

The image styling property allows you to add or to specify valid CSS properties for the image. For more details on CSS properties, click [here](#).

The screenshot shows a user interface for a UI designer. On the left, there is a preview area containing a yellow smiley face icon with a black outline and a small dog icon sitting on a line. Above the smiley face is a placeholder text "Upload an image" and a blue "Upload file" button. To the right of the preview is a detailed property panel for the smiley face image. The property panel includes sections for "Image" (with icons for desktop, mobile, and tablet), "Sizing" (Fit W: fixed, Width: 220 px; Fit H: auto), "Spacing" (top: 0, right: 0, bottom: 0, left: 0, gutter: 16), "Border" (Radius: 0 px, Width: 0 px, Color: #1E1E1C), and "Advanced" (Add class: Comma separated class names). The dog icon has its own separate property panel.

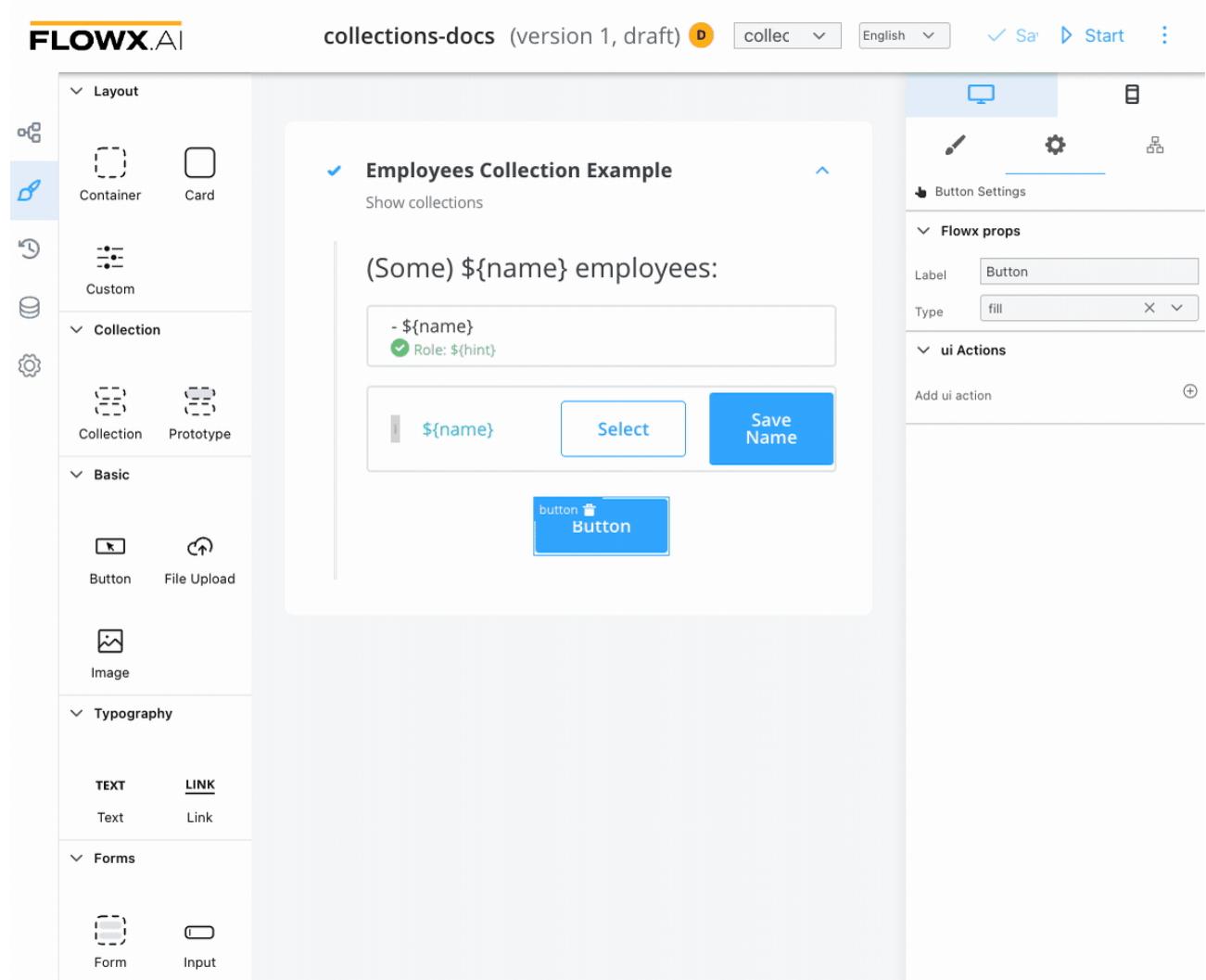
Was this page helpful?

BUILDING BLOCKS / UI Designer / UI actions

Multiple UI elements can be linked to an **action** via a UI Action. If the action is just a method to interact with the process, the UI Action adds information about how that UI should react. For example, should a loader appear after executing the action, should a modal be dismissed, or if some default data should be sent back to the process.

UI actions create a link between an **action** and a UI component or **custom component**.

The UI action informs the UI element to execute the given action when triggered. Other options are available for configuration when setting an action to a button and are detailed below.



There are two main types of UI Actions:

- Process UI Actions
- External UI Actions

Process UI actions

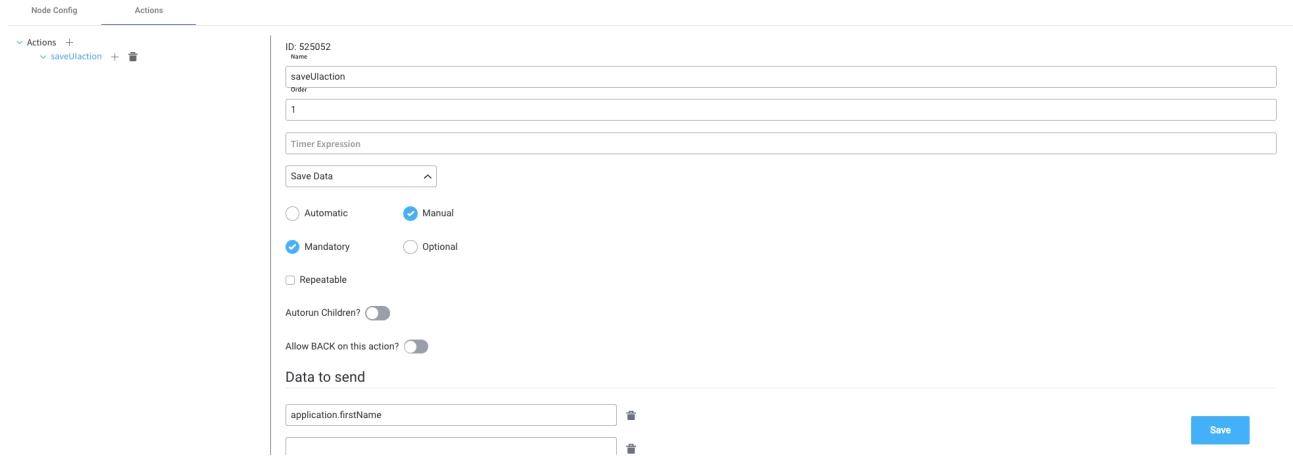
This is a UI action that describes how a **Button** (generated or custom) should interact with a process Manual action.

First, we need to configure the **manual action** that will be referred from the UI Action. For more information on how to add an action to a node, and how to configure it, check the following section:

» [Adding an action to a node](#)

Manual action configuration example - Save Data

1. Add an **action** to a node.
2. Select the action type - for example **Save Data**.
3. The action **type** should be **manual**.
4. **Keys** - it has two important implications:
 - First, this is a prefix of the keys that will send back by the UI Action link to this action. For example, if we have a big form with a lot of elements, but we need an action that just sends the email back (maybe creating email validation functionality) we will add just the key of that field:
`application.client.email`; if we need a button that will send back all the form elements that have keys that start with application.client we can add just this part
 - Second, a backend validation will be run to accept and persist just the data that start with this prefix. If we have three explicit keys,
`application.client.email`, `application.client.phone`,
`application.client.address` and we send
`application.client.age` this key will not be persisted



When this prerequisite is ready we can define the UI Action.

⚠ CAUTION

UI Actions and Actions should be configured on the same node.

UI action configuration example

Multiple configurations are available - **ACTION** type example:

- **Event**
- **Type**
- **Node Action Name** - dropdown with available actions for this node. If the dropdown is empty please add a manual action that is required before we create the UI Action
- **Use a different name for UI action**
- **UI action name - this becomes** important when the action is used in a **Custom component** as it will be used to trigger the action. For UI actions added on a generated button component this name is just descriptive

- **Custom body** - this is the default response in JSON format that will be merged with any extra parameters added explicitly when executing the action, by a web application (from a [custom component](#))
- **Forms to validate** - select from the dropdown what element will be validated (you can also select the children)
- **Dismiss process** - if the UI Actions is added on a subprocess and this parameter is true, triggering this UI action will dismiss the subprocess view (useful for modals subprocess)
- **Show loader?** - a loader will be displayed if this option is true until a web-socket event will be received (new screen or data)

ui Actions

Add ui action +

Add action

×

Event x ▾
CLICK

Type x ▾
ACTION

Node Action Name x ▾
saveUlation

Use a different name for UI action

Custom body

Forms to validate ▼
FORM

Dismiss process? Show loader?

Cancel Save

UI actions elements

Events

You can add an event depending on the element that you select. There are two events types available: **CLICK** and **CHANGE**.

The screenshot shows the FLOWX.AI UI builder interface. On the left, there is a sidebar with various UI element categories: Layout (Container, Card, Custom), Collection (Collection, Prototype), Basic (Button, File Upload, Image), Typography (Text, Link), and Forms (Form, Input). In the center, there is a card titled "First Form Title" with a subtitle "First Form Subtitle". Inside the card, there is a "Form title" section and three input fields labeled "Input 1", "Input 2", and "Input 3". Below the card, there is a blue button labeled "button" with the text "Button". On the right side, there is a sidebar with "Flowx props" settings for the button, showing "Label: Button" and "Type: fill". Below the props, there is a section for "ui Actions" with a "Add ui action" button. The top right of the interface shows the project name "test_docs_node_UI (version 1, draft)", a save button, and other navigation options.



! Not available for *UI Actions* on Custom Components.

Action types

The **action type** dropdown will be pre-filled with the following UI action types:

- DISMISS - used to dismiss a modal after action execution
- ACTION - used to link an action to a UI action
- START_PROCESS_INHERIT - used to inherit data from another process
- UPLOAD - used to create an un upload action
- EXTERNAL - used to create an action that will open a link in a new tab

External UI actions

Used to create an action that will open a link in a new tab.

If we toggle the EXTERNAL type, a few new options are available:

1. **URL** - web URL that will be used for the external action
2. **Open in new tab** - this option will be available to decide if we want to run the action in the current tab or open a new one

Add action

Type

EXTERNAL

Ui action Name

RedirectToGoogle

Url

www.google.com

Params

Dismiss process? Open in new tab?

[Cancel](#) [Save](#)

For more information on how to add actions and how to configure a UI, check the following section:

» [Managing a process flow](#)

Was this page helpful?

BUILDING BLOCKS / UI Designer / Validators

Validators are an essential part of building robust and reliable applications. They ensure that the data entered by the user is accurate, complete, and consistent. In Angular applications, validators provide a set of pre-defined validation rules that can be used to validate various form inputs such as text fields, number fields, email fields, date fields, and more.

The screenshot shows the FLOWX.AI platform interface. On the left, there's a sidebar with various UI component icons categorized under 'Layout', 'Collection', 'Basic', 'Typography', and 'Forms'. In the center, a preview window displays a simple form with a 'Form title' section containing an 'input' field with a 'label' placeholder and another 'input' field with a 'Placeholder'. Below this is a 'Submit' button. To the right of the preview is a detailed configuration panel for the selected 'input' component. This panel includes tabs for desktop and mobile views, fields for 'Suffix' (with placeholder 'eg. suffix') and 'Helpertext' (with a toggle switch), dropdowns for 'Datasource' and 'Default value' (both with placeholder 'eg. Name'), and a 'Validators' section. The 'Validators' section is expanded, showing a list of validators: 'Required', 'Minlength', 'maxlength', 'min', 'max', 'email', 'pattern', and 'custom'. A red box highlights the 'Select a validator' dropdown and the list of validators below it.

Angular provides default validators such as:

1. **min**
2. **max**
3. **minLength**
4. **maxLength**
5. **required**
6. **email**

Other predefined validators are also available:

1. `isSameOrBeforeToday`: validates that a **datepicker** value is in the past
2. `isSameOrAfterToday`: validates that a datepicker value is in the future

 **INFO**

Form validation is triggered by default when the button set to validate a **form** is pressed.

It's also possible to build **custom validators** inside the container application and reference them here by name.

Predefined validators

required validator

This validator checks whether a value exists in the input field.

The screenshot shows the FLOWX.AI interface. On the left is a sidebar with various components: Layout (Container, Card, Custom), Collection (Collection, Prototype), Basic (Button, File Upload, Image), and Typography (Text, Link). The main area displays a form titled "Form title" with two input fields labeled "Input label" and "Placeholder". A "Submit" button is at the bottom. To the right is a detailed configuration panel for the inputs. It includes sections for "Type" (text), "Prefix" (eg. prefix), "Suffix" (eg. suffix), "Helpertext" (with a toggle switch), "Datasource" (with a dropdown arrow), "Default value" (eg. Name), and "Validators". The "Validators" section is highlighted with a red box and contains three items: "Required", "Minlength", and a button "Add a validator". Below this are sections for "Expressions", "Hide" (with a dropdown arrow), and "Disabled" (with a dropdown arrow).

It is recommended to use this validator with other validators like **minlength** to check if there is no value at all.

The screenshot shows the final state of the form. The "Input label" field now has a red border, indicating it is required. Below it, the message "This input is required" is displayed in red. The "Submit" button is blue.

minlength validator

This validator checks whether the input value has a minimum number of characters. If there are no characters at all, this validator will not trigger. It is advisable to use this validator with a **required** validator.

The screenshot shows a modal dialog titled "Add validator". Under "Validator type", "minlength" is selected. In the "Params" field, the value "5" is entered. In the "Error Message" field, the message "At least 5 characters" is displayed. At the bottom, there are "Cancel" and "Save" buttons, with "Save" being highlighted.

maxlength validator

This validator checks whether the input value has a maximum number of characters. If there are no characters at all, this validator will not trigger. It is advisable to use this validator with a **required** validator.

Add validator

Validator type

maxlength

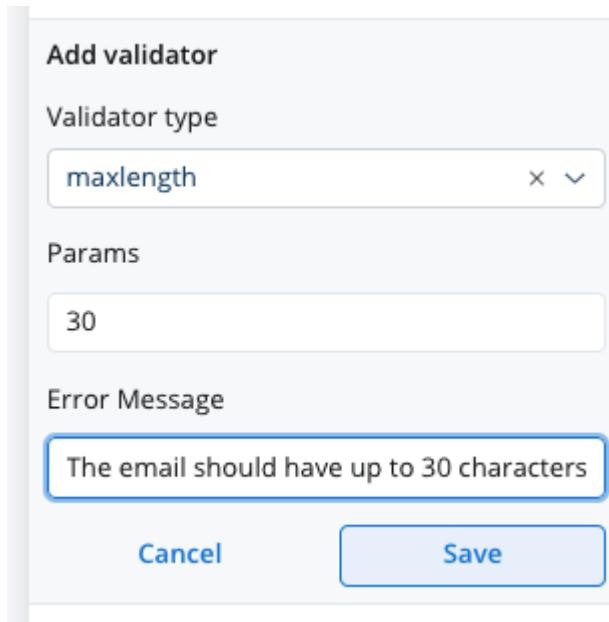
Params

30

Error Message

The email should have up to 30 characters

Cancel Save



min validator

This validator checks whether a numeric value is smaller than the specified value. If there are no characters at all, this validator will not trigger. It is advisable to use this validator with a [required](#) validator.

Edit validator

Validator type

min

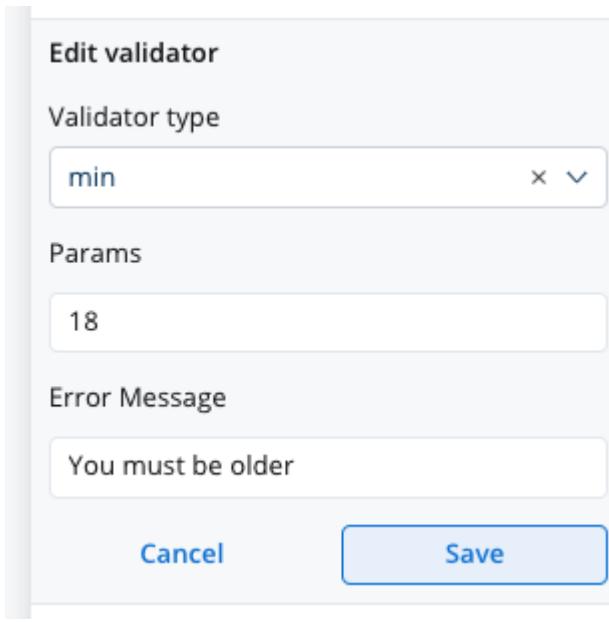
Params

18

Error Message

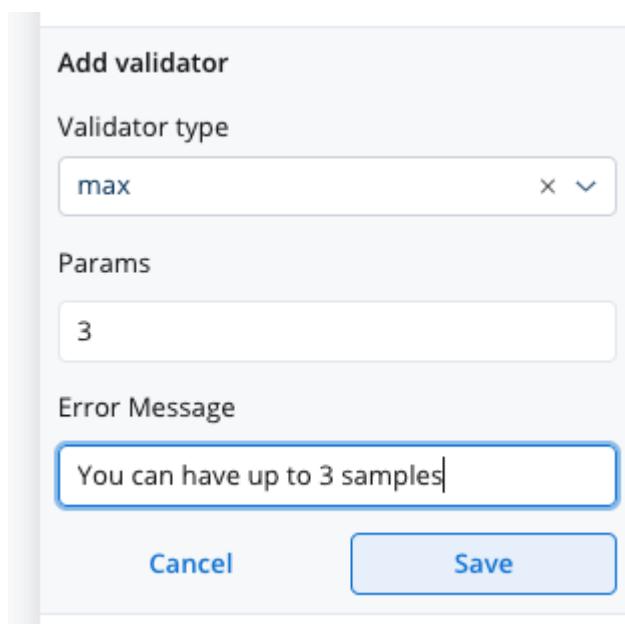
You must be older

Cancel Save



max validator

This validator checks whether a numeric value is larger than the specified value. If there are no characters at all, this validator will not trigger. It is advisable to use this validator with a **required** validator.



email validator

This validator checks whether the input value is a valid email. If there are no characters at all, this validator will not trigger. It is advisable to use this validator with a **required** validator.

Add validator

Validator type

email

Error Message

The entered email is not valid

Cancel Save

pattern validator

This validator checks whether the input value matches the specified pattern (for example, a [regex expression](#)).

Edit validator

Validator type

pattern

Params

`^[0-9]*$`

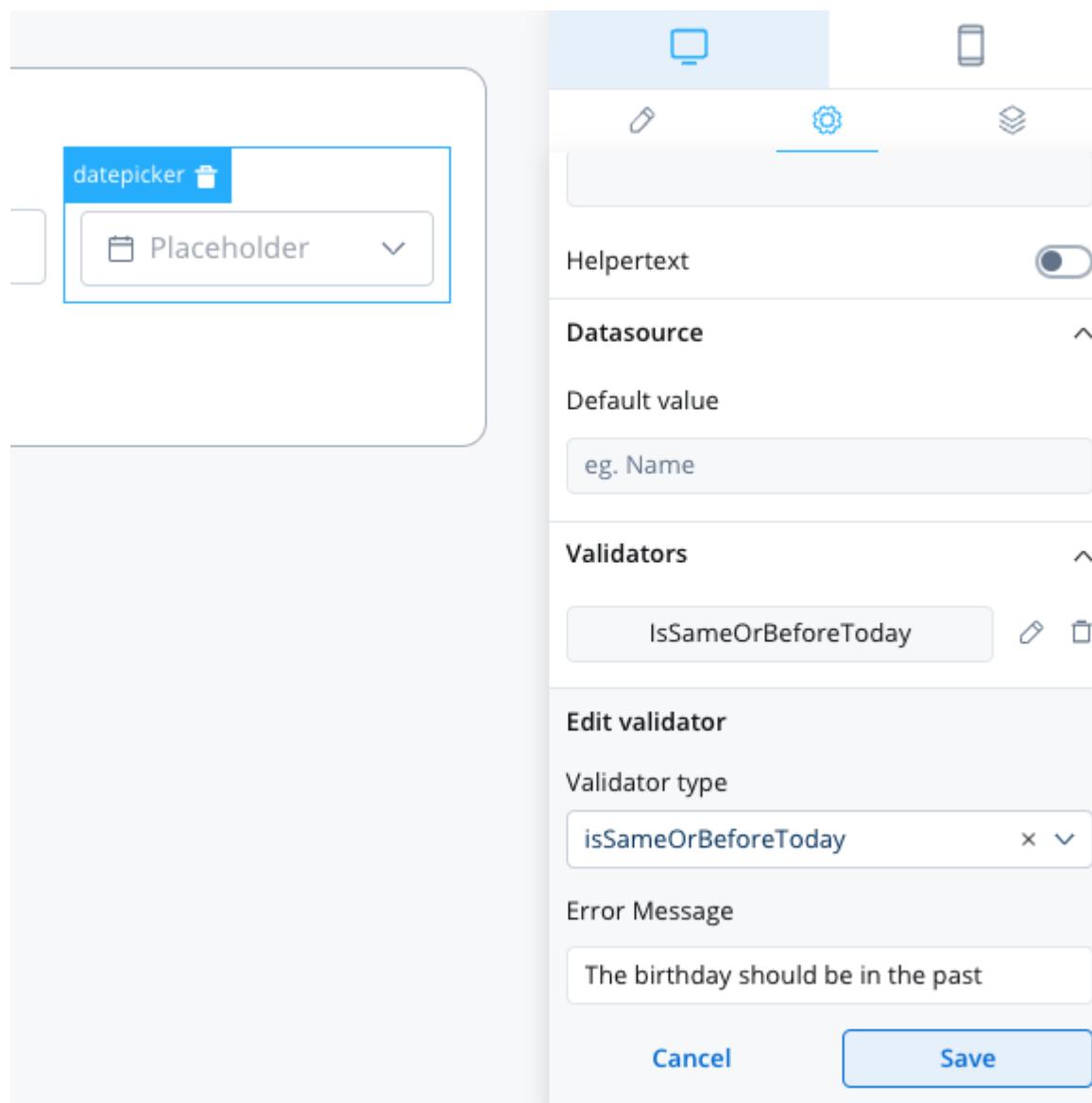
Error Message

Just numbers please

Cancel Save

datepicker - isSameOrBeforeToday

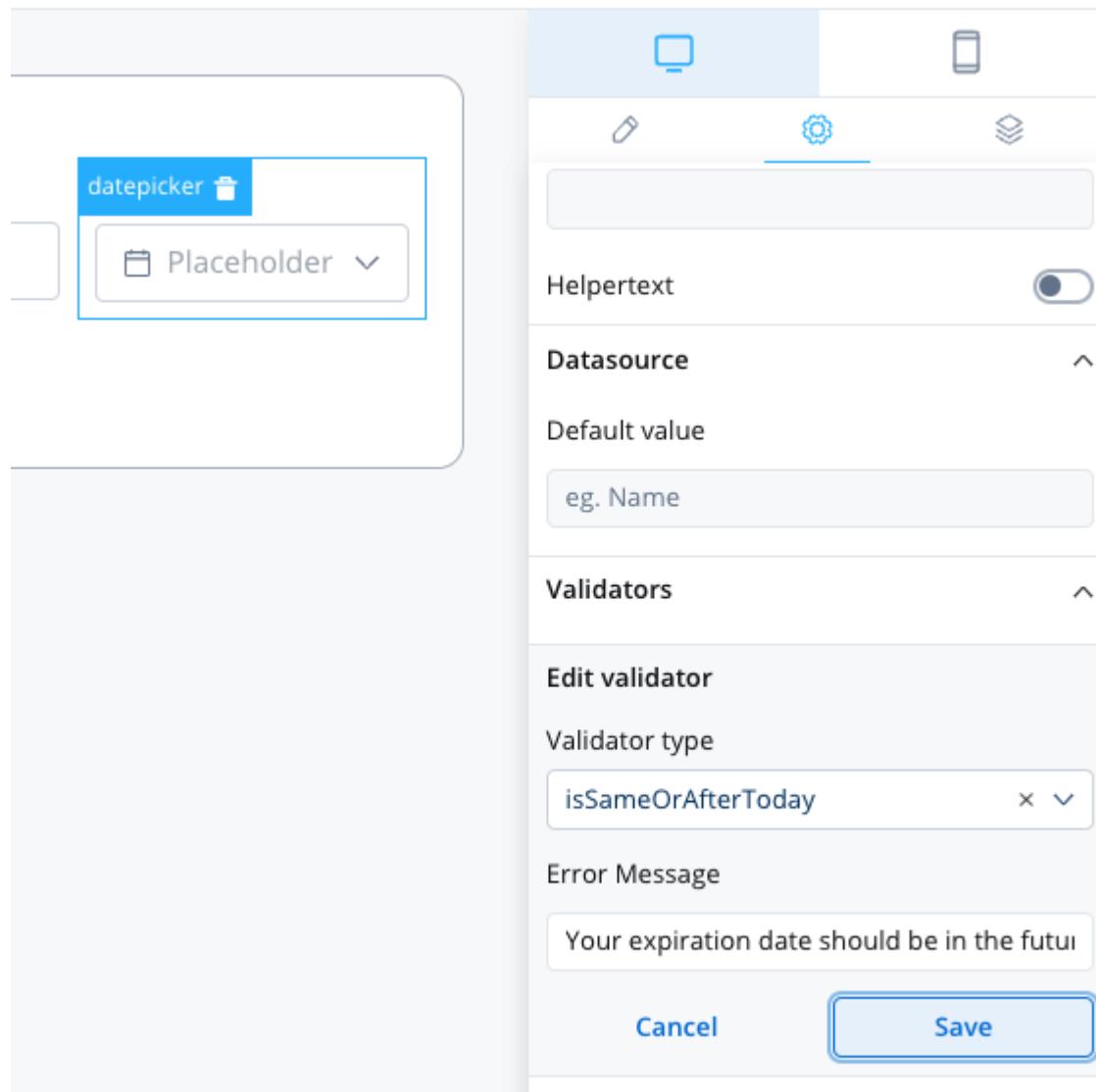
This validator can be used to validate **datepicker** inputs. It checks whether the selected date is today or in the past. If there are no characters at all, this validator will not trigger. It is advisable to use this validator with a **required** validator.



datepicker - isSameOrAfterToday

This validator can be used to validate datepicker inputs. It checks whether the selected date is today or in the future. If there are no characters at all, this

validator will not trigger. It is advisable to use this validator with a **required** validator.



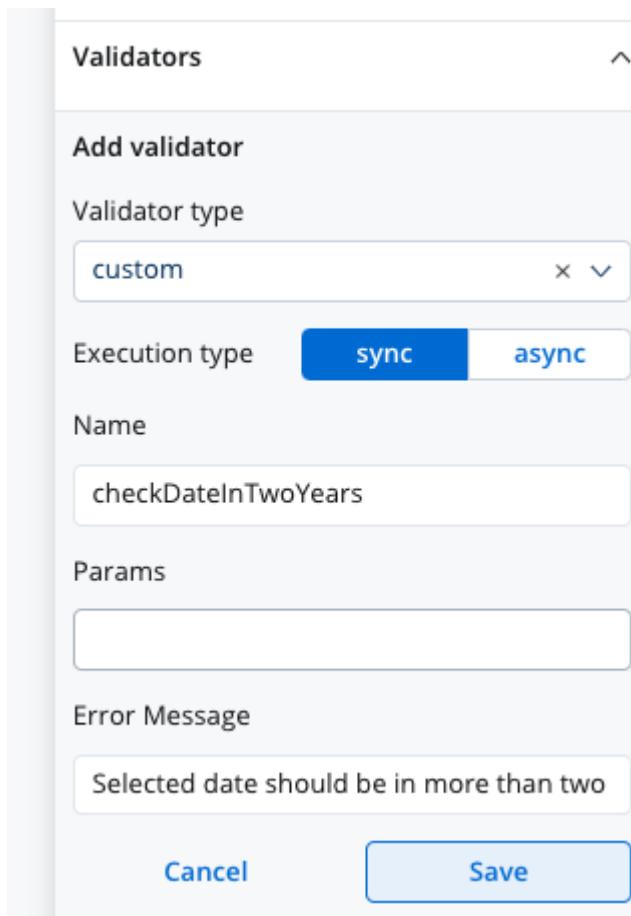
Custom validators

Additionally, custom validators can be created within the web application and referenced by name. These custom validators can have various configurations such as execution type, name, parameters, and error message.

1. **Execution type** - sync/async validator (for more details check [this](#))
2. **Name** - name provided by the developer to uniquely identify the validator
3. **Params** - if the validator needs inputs to decide if the field is valid or not, you can pass them using this list
4. **Error Message** - the message that will be displayed if the field is not valid

INFO

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



The screenshot shows a modal dialog box titled "Validators". It contains fields for "Validator type" (set to "custom"), "Execution type" (set to "sync"), "Name" (set to "checkDateInTwoYears"), and "Error Message" (set to "Selected date should be in more than two"). At the bottom are "Cancel" and "Save" buttons.

Validators	
Add validator	
Validator type	
custom	
Execution type	sync
Name	
checkDateInTwoYears	
Params	
Error Message	
Selected date should be in more than two	
Cancel	Save

Custom validator example

Below you can find an example of a custom validator (`currentOrLastYear`) that restricts data selection to the current or the previous year:

currentOrLastYear

```
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;
}
```

smallerOrEqualsToNumber

Below is another custom validator example that returns `AsyncValidatorFn` param, which is a function that can be used to validate form input asynchronously. The validator is called `smallerOrEqualsToNumber` and takes an array of `params---

sidebar_position: 3

as an input.

ⓘ INFO

For this custom validator the execution type should be marked as `async` using the UI Designer.

```
export function smallerOrEqualsToNumber (params$:  
Observable<any>[]): AsyncValidatorFn {  
    return (AC): Promise<ValidationErrors | null> |  
Observable<ValidationErrors | null> => {  
    return new Observable((observer) => {  
        combineLatest(params$).subscribe(([maximumLoanAmount])  
=> {  
            const validationError =  
                maximumLoanAmount === undefined || !AC.value ||  
Number(AC.value) <= maximumLoanAmount ? null :  
{smallerOrEqualsToNumber: true};  
  
            observer.next(validationError);  
            observer.complete();  
        });  
    });  
};  
}
```

If the input value is undefined or the input value is smaller or equal to the maximum loan amount value, the function returns `null`, indicating that the input is valid. If the input value is greater than the maximum loan amount value, the

function returns a `ValidationErrors` object with a key `smallerOrEqualsToNumber` and a value of true, indicating that the input is invalid.

 **INFO**

For more details about custom validators please check this link.

Using validators in your application can help ensure that the data entered by users is valid, accurate, and consistent, improving the overall quality of your application.

It can also help prevent errors and bugs that may arise due to invalid data, saving time and effort in debugging and fixing issues.

Overall, enforcing data validation using validators is a crucial step in building high-quality, reliable, and user-friendly applications.

[Was this page helpful?](#)

BUILDING BLOCKS / UI Designer / Dynamic & computed values

In modern application development, the ability to create dynamic and interactive user interfaces is essential for delivering personalized and responsive experiences to users. Dynamic values and computed values are powerful features that enable developers to achieve this level of flexibility and interactivity.

Dynamic values

Dynamic values refer to the capability of dynamically populating element properties in the user interface based on process parameters or substitution tags. These values can be customized at runtime, allowing the application to adapt to specific scenarios or user input. With dynamic values, you can personalize labels, placeholders, error messages, and other properties of UI elements, providing a tailored experience for users.

You can now utilize process parameters or **substitution tags** with the following UI elements and their properties:

Element	Property	Accepts Params/Subst Tags
Form Elements	Default Value (except switch)	Yes
	Label, Placeholder	Yes
	Helper Text, Validators	Yes
Document Preview	Title, Subtitle	Yes
Card	Title, Subtitle	Yes
Form	Title	Yes
Message	Message	Yes

Element	Property	Accepts Params/Subst Tags
Buttons	Label	Yes
Select, Checkbox, Radio, Segmented Button (Static)	Label, Value	Subst Tags Only
Text	Text	Yes
Link	Link Text	Yes
Modal	Modal Dismiss Alert, Title, Message, Confirm Label, Cancel Label	Yes
Step	Label	Yes

Example using Substitution tags

The screenshot shows the FLOWX.AI platform interface. On the left, there is a sidebar with various icons and sections: Layout (Container, Card, Custom), Forms (Form, Input, Textarea, Select, Checkbox, Radio, Switch, Datepicker, Segmented Button, Slider), and Collection. In the center, there is a process step titled "input". The step contains the text "This is a substitution tag" and three fields: "prefix_en", "placeholder_en", and "suffix_en", with "helpertext_en" below them. A success message "All data has been validated. Thank you!" is displayed at the bottom. On the right, there is a configuration panel for the "Input" step. It includes tabs for "Input" (selected), "Properties", "Datasource", and "Validators". The "Input" tab shows "Process data key" set to "inputKey". The "Properties" tab shows "Label" set to "@@docs_label", "Placeholder" set to "@@placeholder", "Type" set to "text", "Prefix" set to "@@prefix", "Suffix" set to "@@suffix", and "Has clear" checked. The "Helpertext" section is turned on, with "@@helpertext" as the value. There is also a checkbox for "Hide inside infopoint" which is unchecked. The "Datasource" tab shows "Default value" set to "@@default_value".

Example using process parameters

Business rule example

In the above example, the following MVEL business rule was used to populate the keys with values from the task:

```
//assigning a JSON object containing dynamic values for the
specified keys to the "app" key

output.put("app", {"label": "This is a label",
                  "title": "This is a title",
                  "placeholder": "This is a placeholder",
                  "helpertext": "This is a helper text",
```

```
        "errorM":"This is a error message",
        "prefix":"prx",
        "suffix":"sfx",
        "subtile":"This is a subtitle",
        "message":"This is a message",
        "defaultV":"defaultValue",
        "value":"Value101",
        "value":"Value101",
        "confirmLabel":"This is a confirm
label",
        "cancelLabel":"This is a cancel label",
        "defaultValue":"dfs",
        "defaultDate":"02.02.2025",
        "defaultSlider": 90});
```

⚠ CAUTION

Please note that for releases **<= 3.3.0**, it is not possible to concatenate process parameters with substitution tags when using dynamic values.

Computed values

Computed values take the concept of dynamic values a step further by allowing you to generate values dynamically using JavaScript expressions. Rather than relying solely on predefined values, computed values enable the calculation, transformation, and manipulation of data based on specific rules or conditions.

The screenshot shows the FLOWX.AI application interface. On the left is a sidebar with various UI component icons categorized under 'Layout', 'Forms', and 'Collection'. The main area displays a form titled 'Enter Personal Information' with fields for First Name, Last Name, Date of birth, Employment type (radio buttons for Employed and Pensioner), Income range (a slider from < \$50.000 to \$50.000 - \$100.000), and a Save personal information toggle. Below these are fields for Loan amount (text input) and Down payment (a slider with a value of NaN\$). A modal window titled 'Computed expression' is open over the form, containing a JavaScript code editor with the following code:

```

1 if ( !isNaN(parseInt(${application.client.amount})) ) {
2     return 0.15 * parseInt(${application.client.amount})
3 } else {
4     return 2000
5 }
6

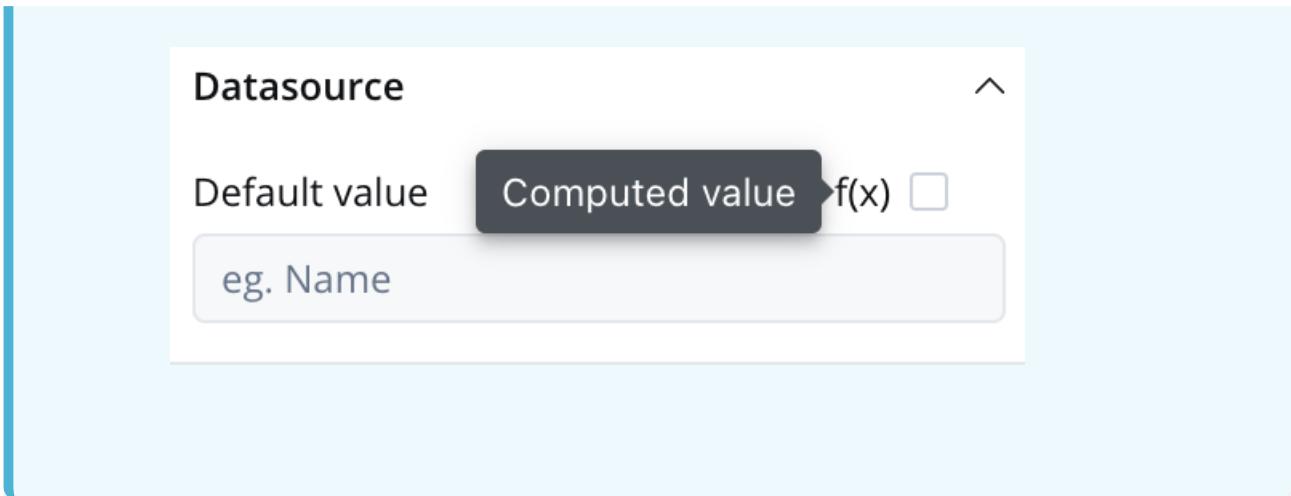
```

The modal also includes tabs for 'Default value' (containing 'eg. Name') and 'Validators' (with a 'Add a validator' button).

Computed values can be created by writing JavaScript expressions that operate on process parameters or other variables within the application.

INFO

To add a computed value, you have to explicitly check “Computed value” option (represented by the **f(x)** icon), which will transform the desired field into a JavaScript editor.



By enabling computed values, the application provides flexibility and the ability to create dynamic and responsive user interfaces.

Slider Properties:

- Process data key: application.client.avans
- General properties:
 - Slider label: Down payment
 - Show value label: checked
- Helper text:
- Min Value: f(x)

```
if ( !isNaN(parseInt(${application.client.amount})) ) {
  return 0.15 * parseInt(${application.client.amount})
} else {
```

- Max Value: f(x)

```
if ( !isNaN(parseInt(${application.client.amount})) ) {
  return 0.35 * parseInt(${application.client.amount})
} else {
```

- Suffix: \$
- Step size: 1000

Datasource:

- Default value: f(x)
- eg. Name

Slider example

The above example demonstrates the usage of computed values for a Slider element, where JavaScript expressions are used to compute the minimum and maximum values based on a value entered in an input UI element (linked by the process key `${application.client.amount}`).

Min Value

```
if ( !isNaN(parseInt(${application.client.amount})) ) {  
    return 0.15 * parseInt(${application.client.amount})  
} else {  
    return 10000  
}
```

Max Value

```
if ( !isNaN(parseInt(${application.client.amount})) ) {  
    return 0.35 * parseInt(${application.client.amount})  
} else {  
    return 20000  
}
```

Example details

The code snippets check whether the value of

`${application.client.amount}` key can be successfully parsed as an integer. Here's a step-by-step explanation:

- The `parseInt()` function is used to attempt to convert `${application.client.amount}` into an integer.

- The `isNaN()` function is then used to check if the result of the conversion is `NaN` (not a number).
- If the value is not `NaN`, it means `${application.client.amount}` is a valid numeric value.
- In that case, the code calculates the computed value by multiplying the parsed integer by `0.15` (the minimum percentage value for the down payment or with 0.35, the maximum percentage value of the down payment). The result is returned as the computed value for the expression.
- If the value is `NaN` (or `${application.client.amount}` couldn't be successfully parsed as an integer), the code executes the `else` block and returns a default value of `10000`.

In summary, the JS expressions demonstrates how a computed value can be derived based on a conditional calculation. It first checks if a specific process parameter (`${application.client.amount}`) is a valid numeric value, and if so, it computes the value by multiplying it by 0.15 or 0.35. Otherwise, it falls back to a default value of `10000` or `20000`.

Usage

The UI Designer now allows JavaScript expressions to create computed values used on the following UI elements with their properties:

Element	Properties
Slider	min Value, max Value, default Value
Input	Default Value

Element	Properties
Any UI Element that accepts validators	min, max, minLength, maxLength
Text	Text
Link	Link Text

- **Slider:** The min value, max value, and default value for sliders can be set using JavaScript expressions applied to process parameters. This allows for dynamic configuration based on numeric values.
- **Any UI Element that accepts validators min, max, minLength, maxLength:** The "params" field for these elements can also accept JavaScript expressions applied to process parameters. This enables flexibility in setting validator parameters dynamically.
- **Default Value:** For input elements like text inputs or number inputs, the default value can be a variable from the process or a computed value determined by JavaScript expressions.
- **Text:** The content of a text element can be set using JavaScript expressions, allowing for dynamic text generation or displaying process-related information.
- **Link:** The link text can also accept JavaScript expressions, enabling dynamic generation of the link text based on process parameters or other conditions.

Please note that these settings are specifically applicable to numeric values and are not intended for date or string values.

CAUTION

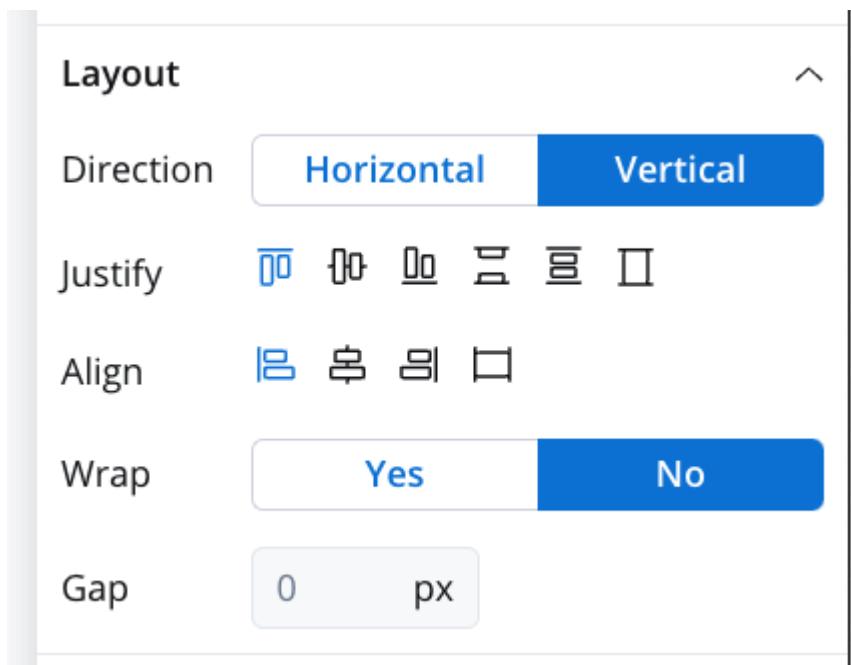
For input elements (e.g., text input), you may require a default value from a process variable, while a number input may need a computed value.

Was this page helpful?

BUILDING BLOCKS / UI Designer / Layout configuration

Layout settings will be available for all components that can group other types of elements (for example, [Container](#) or [Card](#)).

These settings allow users to customize properties as the layout direction, alignment, gap, sizing, and spacing.



These settings can be applied practically in various ways, depending on the context and purpose of the design. For example:

- The layout direction and alignment settings can be used to ensure that the content is displayed in a logical and visually appealing way. For instance, a left-to-right layout direction may be more appropriate for languages that read from left to right, while a center alignment may be better for headings or titles.
- The gap, sizing, and spacing settings can be used to control the distance between elements in a design. This can help create a sense of hierarchy and balance, as well as improve readability and usability. For example, increasing the spacing between paragraphs or sections can make the content easier to scan and read, while reducing the size of certain elements can help prioritize others.
- Customizing these properties can also help ensure that a design is accessible and inclusive. For instance, adjusting the layout direction and alignment settings can make the design more readable for users with certain disabilities or who use assistive technologies. Similarly, increasing the spacing and sizing of interactive elements can make them easier to click or tap for users with motor impairments.

To better understand how these layout configurations work and see real-time feedback on different combinations, please refer to the following link:

» [Layout configuration](#)

Was this page helpful?

BUILDING BLOCKS / UI Designer / Rendering and UI Designer changelog

⚠ CAUTION

This changelog is relevant to the 3.0 release, which introduces significant changes to the UI configuration.

Please be aware that a process (when it comes to UI configuration) may look different from previous releases and that certain updates may not be compatible with older configurations. The migration process may take longer than usual.

Notes for post-migration

After migrating to the 3.0 release, please take in consideration the following changes to ensure smooth operation:

1. Verify font sizes where float values were set.
2. Verify line heights where scale values were set.
3. Verify border radius values where values other than px were set.
4. Review and set padding and margin values where needed. Deleted keys include "margin" : "8px 0" and "padding" : "16px 0 0 16px".
5. Check **radio** and **checkbox** elements and update the new `label` prop that has been added.
6. Configure the new `column` prop for Layout (available for checkbox and radio), which allows for grouping many enumerations.
7. The `height` prop (from **Container**, **Form** and **Card**) has been removed.

8. Update the width prop by configuring the new `Fit W` prop with values such as fill, fixed, or auto. `Min H` and `Max H` props have been removed.
9. The hint and message UI components were combined into a single component called 'message' with the following properties: `type`, `fill`, `border`, and `text`.
10. The `button` element no longer has the `fill`, `border`, and `flat` types. It now has 4 types: `primary`, `secondary`, `ghost`, and `text`.
11. Added Helpertext (to replace Info tooltip) - this new element can be found on [Form elements](#) and provides additional information about each element, which can be hidden within an infopoint.

[Was this page helpful?](#)

BUILDING BLOCKS / Token

Token is the concept that describes the current position in the process flow. When you start the process you have a graph of [nodes](#) and based on the configuration you will go from one to another based on the defined sequence (connection between nodes).

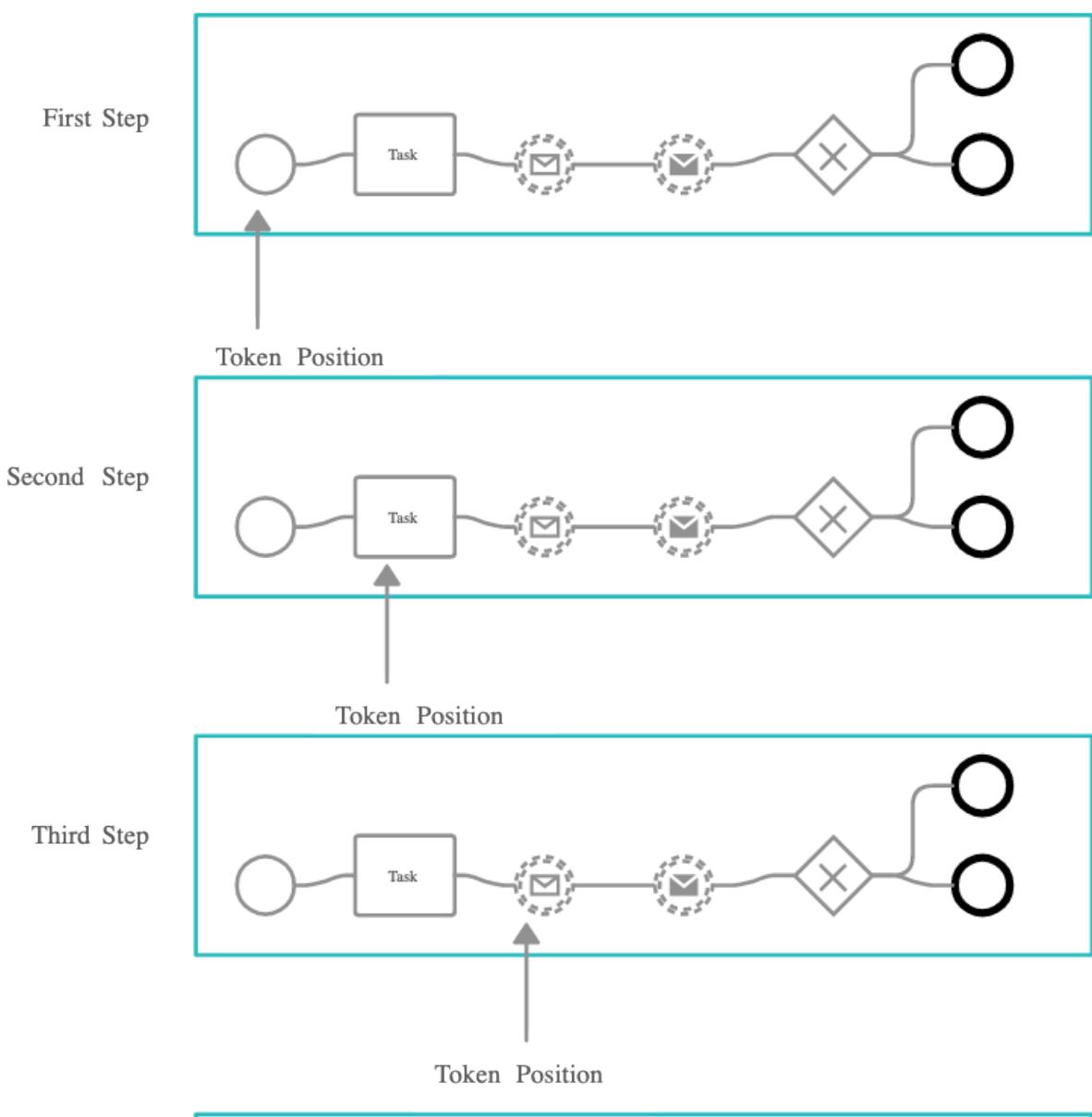
The token is a [BPMN](#) concept that represents a state within a process instance. It keeps track of the current position in the process flow and is used to store data related to the current process instance state.

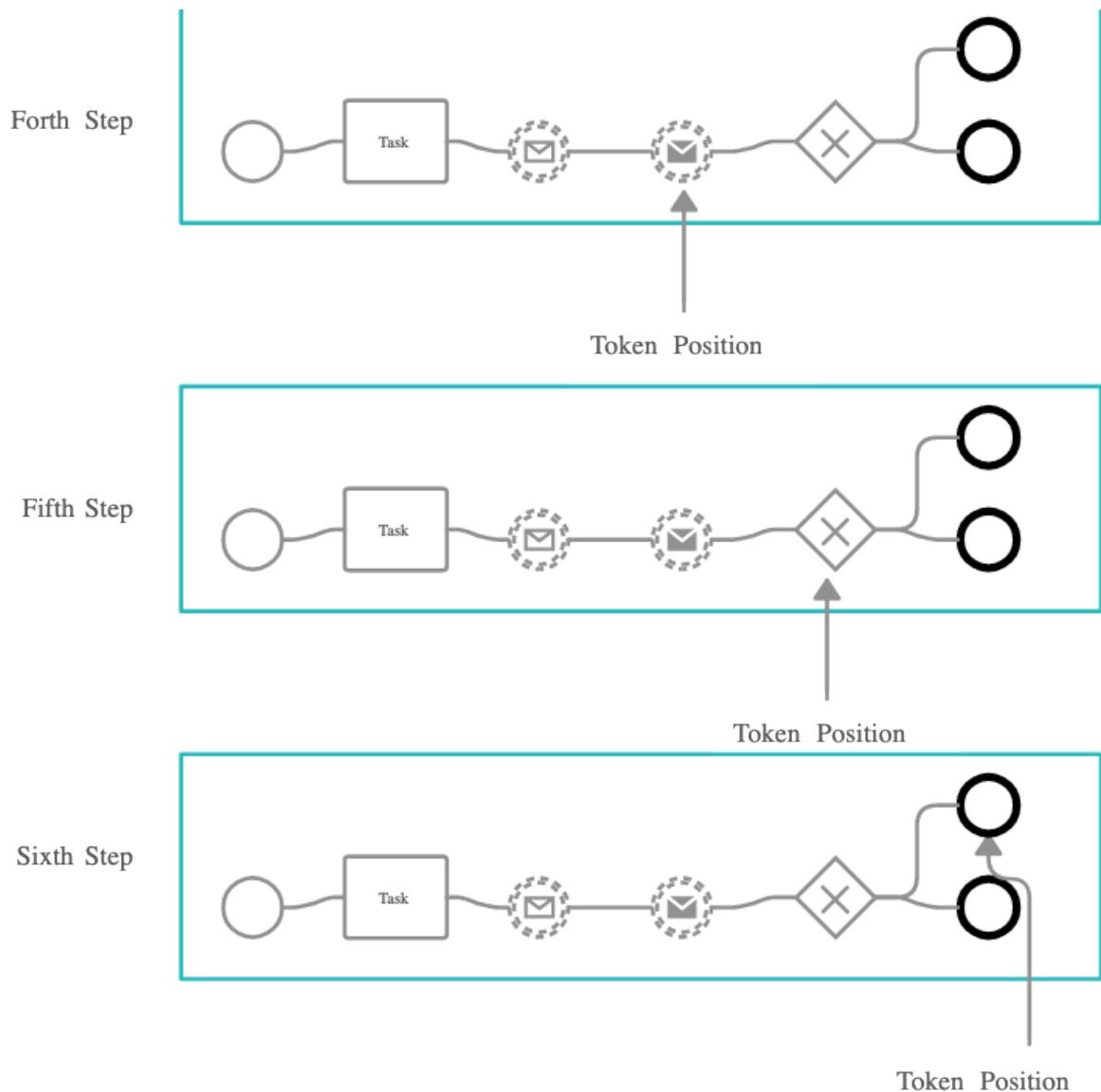
A token is created each time a new process instance is started. As the actions on the process instance are executed, the token advances from one node to the next.

As a node can have several **actions** that need to be executed, the token is also used for keeping track of the actions executed in each node.

In case of **parallel gateways**, child tokens are created for each flow branch. The parent token moves to the gateway sync node and only advances after all the child tokens also reach that node.

The image below shows how a token advances through a process flow:





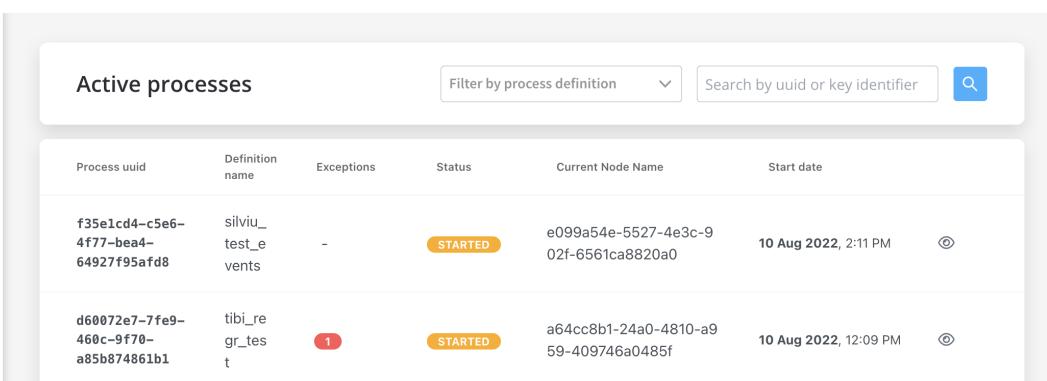
The token will only move to the next node when there are no more mandatory actions from the current node that need to be executed. The token will also wait on a node in case the node is set to receive an event from an external system through Kafka.

There will be cases when the token needs to be stopped in a node until some input is received from the user. If the input from the user is needed for further

advancing in the process, the token should only advance after all data was received. A mandatory manual action can be used in this case and linked to the user action. This way we make sure that the process flow advances only after the user input is received.

Checking the token status

The current process instance status can be retrieved using the FLOWX Designer. It will display some info on the tokens related to that process instance and the current nodes they are in.



The screenshot shows the FLOWX Designer interface with the following details:

- Left sidebar:** Processes Definitions, Active process (selected), Process Instances, Failed process start, Content Management (selected), Enumerations.
- Top bar:** FLOWX.AI logo, Active processes, Filter by process definition, Search by uid or key identifier, magnifying glass icon.
- Table:** Active processes table with columns: Process uid, Definition name, Exceptions, Status, Current Node Name, Start date.
- Data:**
 - Row 1: Process uid f35e1cd4-c5e6-4f77-bea4-64927f95afdb, Definition name silviu_test_events, Exceptions -, Status STARTED, Current Node Name e099a54e-5527-4e3c-902f-6561ca8820a0, Start date 10 Aug 2022, 2:11 PM.
 - Row 2: Process uid d60072e7-7fe9-460c-9f70-a85b874861b1, Definition name tibi_regr_test, Exceptions 1, Status STARTED, Current Node Name a64cc8b1-24a0-4810-a959-409746a0485f, Start date 10 Aug 2022, 12:09 PM.

In case more details are needed about the token, you can click the **Process status** view button, choose a token then click the **view button** again:

The screenshot shows the FLOWX.AI interface. On the left, there's a sidebar with navigation links for Processes (Definitions, Active process, Process Instances, Failed process start), Content Management (Enumerations, Substitution tags, Content models, Languages), and a user profile for Dragos Caravasile. The main area is titled "Process Definitions" and contains two tables:

Name	Version	Edited at	Edited by
AutoTestProcess72860032	1	29 Jun 2022, 1:49 PM	QA FlowX
TA_BackinSteps_Process_1654862288218	10	29 Jun 2022, 11:33 AM	QA FlowX

Name	Version	Published at	Published by
TA_Subprocess_2	4	29 Jun 2022, 11:40 AM	QA FlowX
TA_Subprocess_item_2	4	29 Jun 2022, 11:40 AM	QA FlowX

Token status details

The following token details are available when you access and view the JSON file of a token in FLOWX Designer:

```
id: 492952
version: 31
parentTokenId: null
currentNodeId: 491660
currentnodeName: null
state: "INACTIVE"
statusCurrentNode: "EXECUTED_COMPLETE"
syncNodeTokensCount: 0
syncNodeTokensFinished: 0
dateUpdated: "2022-05-18T09:57:58.639911Z"
paramValues: null
processInstanceId: 492902
currentNode: null
nodesActionStates:
  0: Object {"nodeId":491663,"name":"Start","arrivedDate":"2022-05-18T09:57:58.639911Z","lastTransitionDate":"2022-05-18T09:57:58.639911Z","status":1,"action":{},"value":{},"error":{},"script":{},"scriptResult":{}}
```

```
1: Object {"nodeId":491657,"name":"stepper","arrivedDate":"2023-07-26T09:57:58.085Z","lastExecutionDate":null,"state":"COMPLETED","lastExecutionTime":1684993078000,"actionStateData":null}
2: Object {"nodeId":491656,"name":"step1","arrivedDate":"2023-07-26T09:57:58.085Z","lastExecutionDate":null,"state":"COMPLETED","lastExecutionTime":1684993078000,"actionStateData":null}
3: Object {"nodeId":491662,"name":"Client Form","arrivedDate":null,"lastExecutionDate":null,"state":"PENDING","lastExecutionTime":0,"actionStateData":null}
{"492053": {"name": "saveClient", "state": "COMPLETED", "lastExecutionDate": "2023-07-26T09:57:58.085Z", "lastExecutionTime": 1684993078000, "actionStateData": null}}
4: Object {"nodeId":491664,"name":"end step1","arrivedDate":"2023-07-26T09:57:58.085Z","lastExecutionDate":null,"state":"COMPLETED","lastExecutionTime":1684993078000,"actionStateData":null}
5: Object {"nodeId":491661,"name":"step2","arrivedDate":"2023-07-26T09:57:58.085Z","lastExecutionDate":null,"state":"PENDING","lastExecutionTime":0,"actionStateData":null}
6: Object {"nodeId":491655,"name":"company form","arrivedDate":null,"lastExecutionDate":null,"state":"PENDING","lastExecutionTime":0,"actionStateData":null}
{"492052": {"name": "SaveCompany", "state": "COMPLETED", "lastExecutionDate": "2023-07-26T09:57:58.085Z", "lastExecutionTime": 1684993078000, "actionStateData": null}}
7: Object {"nodeId":491658,"name":"stop step2","arrivedDate":null,"lastExecutionDate":null,"state":"PENDING","lastExecutionTime":0,"actionStateData":null}
8: Object {"nodeId":491659,"name":"stop stepper","arrivedDate":null,"lastExecutionDate":null,"state":"PENDING","lastExecutionTime":0,"actionStateData":null}
9: Object {"nodeId":492452,"name":"CreateDocument","arrivedDate":null,"lastExecutionDate":null,"state":"PENDING","lastExecutionTime":0,"actionStateData":null}
{"492102": {"name": "sendInformation", "state": "COMPLETED", "lastExecutionDate": "2023-07-26T09:57:58.085Z", "lastExecutionTime": 1684993078000, "actionStateData": null}}
10: Object {"nodeId":492453,"name":"ReceiveDocuments","arrivedDate":null,"lastExecutionDate":null,"state":"PENDING","lastExecutionTime":0,"actionStateData":null}
18T09:57:58.085Z", "actionStateData": null}
11: Object {"nodeId":491660,"name":"end_process","arrivedDate":null,"lastExecutionDate":null,"state":"PENDING","lastExecutionTime":0,"actionStateData":null}
backSeq: Object {"nodes": [491663, 491657, 491656, 491662, 491664, 491658, 491659]}
uuid: "794954a7-875f-4508-bbcb-8a11cf7a9b37"
```

Token status details	Execution details
id	492952
version	31
parentTokenId	null

Token status details	Ex
currentNodeId	491660
state	ACTIVE, ON_HOLD, INACTIVE
statusCurrentNode	ARRIVED, EXECUTING, EXECUTED_PARTIAL, EXECUTED, MESSAGE_RESPONSE_TIMED_OUT
syncNodeTokensCount	syncNodeTokensCount: 0

Token status details	
syncNodeTokensFinished	syncNodeTokensFinished: 0
dateUpdated	"2022-05-18T09:53:28.587930Z"

Token status details		Ex
processInstanceId	492902	
nodesActionStates		<pre>0: Object {"nodeId":491663,"name":"Start", "actionState": "PENDING", "lastModified": "2023-07-18T09:56:39.576Z", "actionStateData": null}</pre>
backSeq		<pre>Object {"nodes": [491663, 491657, 491656, 491662, 491664, 491665]}</pre>

Was this page helpful?

BUILDING BLOCKS / Supported scripts

Supported scripts

Scripts are used to define and run **actions** but also properties inside **nodes**. For now, the following script languages are supported:

- Python (Jython)
- DMN
- MVEL
- Groovy
- JavaScript (Nashorn Engine)

Scripting Language	Version
Python (Jython)	2.7.0
DMN	1.3
MVEL	2.4.10
Groovy	3.0.8

Scripting Language	Version
Nashorn engine (JavaScript)	15.4

Python

(!) INFO

We use **Jython**.

Jython is an implementation of the high-level, dynamic, object-oriented language **Python** seamlessly integrated with the **Java** platform. Jython is an open-source solution.

Properties:

- Supports **Python 2.7** most common python libs can be imported, ex: math, time, etc.
- Java libs can also be imported: [details here](#)

Useful links:

» [Python 2.7.18 documentation](#)

» [Jython](#)

» [Jython FAQs](#)

DMN

Decision Model and Notation (DMN) is a standard for Business Decision Management.

FLOWX uses [BPMN.io](#) (based on **camunda-engine-dmn** version **7.14.0**) which is built on [DMN 1.3](#) standards.

Properties:

camunda-engine-dmn supports [DMN 1.3](#), including Decision Tables, Decision Literal Expressions, Decision Requirements Graphs, and the Friendly Enough Expression Language (FEEL)

Useful links:

» [Decision Model and Notation \(DMN\)](#)

» [DMN 1.3 specs](#)

More information:

» [Intro to DMN](#)

» DMN Business Rule Action

MVEL

MVEL is a powerful expression language for Java-based applications. It provides a plethora of features and is suited for everything from the smallest property binding and extraction, to full-blown scripts.

- FLOWX uses **mvel2 - 2.4.10 version**

Useful links:

» Mvel documentation

» Maven repository: Mvel 2.4.0 final

More information:

» Intro to MVEL

Groovy

Groovy is a multi-faceted language for the Java platform. The language can be used to combine Java modules, extend existing Java applications and write new

applications

We use and recommend **Groovy 3.0.8** version, using **groovy-jsr223** engine.

! **INFO**

Groovy has multiple ways of integrating with Java, some of which provide richer options than available with **JSR-223** (e.g. greater configurability and more security control). **JSR-223** is recommended when you need to keep the choice of language used flexible and you don't require integration mechanisms not supported by **JSR-223**.

! **INFO**

JSR-223 (spec) is a **standard scripting API for Java Virtual Machine (JVM) languages**. The JVM languages provide varying levels of support for the JSR-223 API and interoperability with the Java runtime.

Useful links:

» [Groovy Language Documentation](#)

» [\[Java\] Class GroovyScriptEngineImpl](#)

Nashorn Engine (JavaScript)

Nashorn engine is an open source implementation of the [ECMAScript Edition 5.1 Language Specification](#). It also implements many new features introduced in ECMAScript 6 including template strings; `let`, `const`, and block scope; iterators and `for..of` loops; `Map`, `Set`, `WeakMap`, and `WeakSet` data types; symbols; and binary and octal literals. It is written in Java and runs on the Java Virtual Machine.

Latest version of **Nashorn** is **15.4**, available from [Maven Central](#). You can check the [changelog](#) to see what's new.

Useful links:

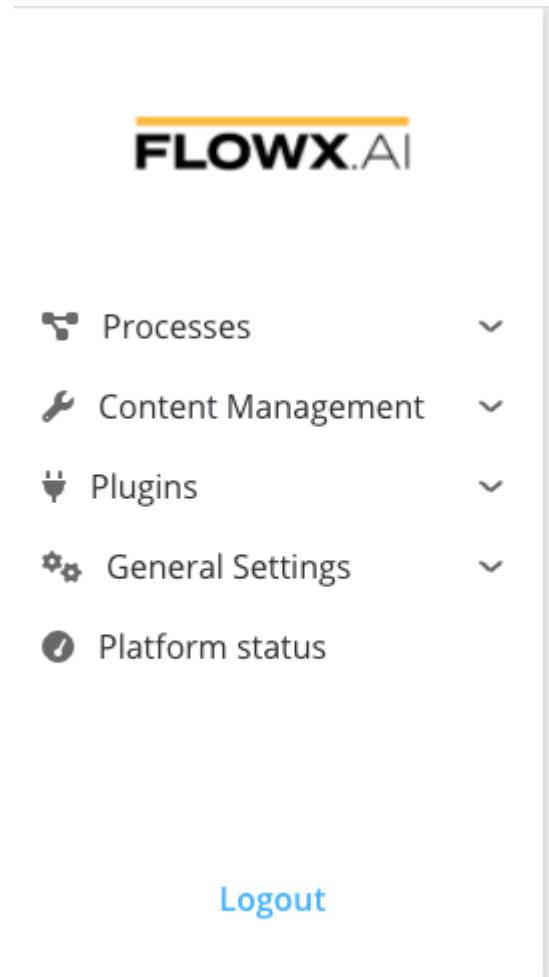
» [GitHub - Nashorn](#)

» [OpenJDK - Nashorn](#)

Was this page helpful?

FLOWX.AI DESIGNER / What is the FLOWX Designer?

FLOWX Designer is the FLOWX.AI no-code visual development environment. It allows users to create web and mobile applications without having to know how to code.



With the FLOWX Designer, you can:

- Develop
The fallback content to display on prerendering
based on **BPMN 2.0**
- Configure interfaces for the processes for both generated and custom screens
- Define
The fallback content to display on prerendering
and validations via **DMN** files or via **MVEL**, or using other supported scripting
languages
- Create integration connectors in a visual manner

- Create
The fallback content to display on prerendering
for your applications
- Adding new capabilities by using [plugins](#)
- Manage users access

This section will go through the steps to set up the

The fallback content to display on prerendering
and how to use it.

» [Overview](#)

» [Managing a process flow](#)

» [Designer setup guide](#)

[Was this page helpful?](#)

FLOWX.AI DESIGNER / Overview

👉 Let's go through the main options available in the

The fallback content to display on prerendering

▶ Processes

▶ Content Management

▶ Plugins

▶ General Settings

▶ Platform status

The screenshot shows a sidebar menu on the left side of a white page. At the top of the sidebar is a yellow horizontal bar. Below it, the FLOWX.AI logo is displayed in a bold, black, sans-serif font. The sidebar contains five items, each with an icon and a label followed by a downward arrow:

- Processes
- Content Management
- Plugins
- General Settings
- Platform status

The "Platform status" item is highlighted with a blue circle around its icon.

INFO

Depending on your access rights, some tabs might not be visible. For more information, check Configuring access rights for Admin section.

Managing process definitions

A

The fallback content to display on prerendering is uniquely identified by its name and version number.

The screenshot shows the FlowX.AI interface with the sidebar collapsed. The main area is titled "Process Definitions".

Drafts / In progress:

Name	Version	Edited at	Edited by	Actions
Test Process	36	26 May 2022, 11:01 AM	QA FlowX	▶ ⚒ ⋮
Test Process 1	1	25 May 2022, 9:13 AM	QA FlowX	▶ ⚒ ⋮

Published:

Name	Version	Published at	Published by	Actions
Happy Process	1	24 May 2022, 2:59 PM	QA FlowX	▶ ⚒ ⋮
Demo Process	1	23 May 2022, 8:53 PM	QA FlowX	▶ ⚒ ⋮

» Process definition

» Managing a process flow

Viewing active process instances

The complete list of active

The fallback content to display on prerendering
is visible from the FLOWX Designer. They can be filtered by
The fallback content to display on prerendering
names and searched by their unique ID. You can also view the current process
instance status and data.

The screenshot shows the FLOWX.AI interface with a sidebar on the left containing navigation links such as Processes, Definitions, Active process, Content Management, Plugins, General Settings, and Platform status. The main content area is titled "Active processes" and displays a table of four completed processes. Each row in the table includes the process UUID, definition name, status (FINISHED), current node name (End), start date (18 May 2022, 5:56 PM), and a refresh icon.

Process uuid	Definition name	Status	Current Node Name	Start date
f6c051ab-f066-4e1a-b98e-5a3bf3b887b5	Client Identification	FINISHED	End	18 May 2022, 5:56 PM
9278863b-6f47-42aa-9e8e-a8b3d059df10	Test Process	FINISHED	End	18 May 2022, 5:53 PM
3d16a65d-7224-4ffa-9f88-17ea3c6fbf50	Test Process 1	FINISHED	End	18 May 2022, 5:53 PM
537c7e51-bc11-4fde-be24-c3e33e686749	Test Process 2	FINISHED	End	18 May 2022, 5:51 PM
a8a5ad8d-277e-46d2-b45e-80dc0728460e	Test Process 3	FINISHED	End	18 May 2022, 5:46 PM
fa273456-bd03-48d1-98fc-710aff4c8794	Test Process 4	FINISHED	End	18 May 2022, 5:44 PM

» Process instance

Managing CMS

Using the content management feature you can perform multiple actions that enable manipulation of the content and simplification of it. You need first to deploy the CMS service in your infrastructure, so you can start defining and using the custom content types described in the **Content Management** tab above.

The screenshot shows the FLOWX.AI platform interface. On the left is a sidebar with navigation links: Processes, Content Management, Plugins, General Settings, and Platform status. A 'Logout' link is also present. The main area is titled 'Process Definitions' and contains two tables: 'Drafts / In progress' and 'Published'.
Drafts / In progress:

Name	Version	Edited at	Edited by
1		25 May 2022, 4:40 PM	QA FlowX

Published:

Name	Version	Published at	Published by
4		26 May 2022, 12:49 PM	John Doe

» Headless CMS

» CMS Setup Guide

Managing tasks

The Task Manager

The fallback content to display on prerendering has the scope to show a process that you defined in Designer, offering a more business-oriented view. It also offers interactions at the assignment level.

The screenshot shows the FLOWX.AI web application. On the left is a sidebar with the following navigation:

- Processes
- Content Management
 - Enumerations
 - Substitution tags
 - Content models
- Languages
- Source systems
- Plugins
 - Task Manager
 - All tasks
 - Hooks
 - Stages
 - Notification templates
 - Document templates
- General Settings
- Platform status

The main content area is titled "Activities". It features a header with buttons for "Status", "Stages · 2", and a search bar. Below is a table with the following data:

Title	Stage	Assignee	Status	Priority	Last updated
Task 1	Onboarding	silviu@flowx.ai	STARTED	3	25.5.2022, 12:45
Task 2	Onboarding	front@flowx.ai	EXPIRED	3	8.3.2022, 16:21
Task 3	Onboarding	silviu@flowx.ai	EXPIRED	3	27.1.2022, 17:50
Task 4	Onboarding	silviu@flowx.ai	FAILED	3	15.12.2021, 2:40
Task 5	Onboarding	silviu@flowx.ai	FAILED	3	15.12.2021, 2:40
Task 6	Onboarding	silviu@flowx.ai	FAILED	3	15.12.2021, 2:40
Task 7	Onboarding	silviu@flowx.ai	FAILED	3	15.12.2021, 2:40
Task 8	Onboarding	silviu@flowx.ai	FAILED	3	15.12.2021, 2:40
Task 9	Onboarding	silviu@flowx.ai	FAILED	3	15.12.2021, 2:40

Below the table, a callout box contains the text: "» Task Management".

Managing notification templates

The notification templates plugin can be viewed, edited, and activated/inactivated from the

The fallback content to display on prerendering

The screenshot shows the FLOWX.AI platform's Notifications Template editor. On the left, a sidebar navigation includes sections for Processes, Content Management (with sub-options like Enumerations, Substitution tags, Content models, Languages, and Source systems), Plugins (Task Manager, All tasks, Hooks, Stages), Notification templates (selected), Document templates, General Settings (Generic Parameters), and Platform status. At the bottom of the sidebar are Logout and Test links. The main workspace is titled "Notifications Template - ExampleTemplate". It has tabs for "Body" (selected) and "Data model". Under "Body", settings include "Type: MAIL", "Forward on Kafka" (unchecked), "Language: Romanian (Romania)-ro-RO", and "Subject: Contract [\${firstInput}]". Below these are "Source" and "Format" buttons. The central area contains a rich-text editor toolbar and a preview of the email template. The template itself features a header with the FLOWX.AI logo, a greeting "Salut", a message "Salut #firstInput! #secondInput, ne bucurăm sa te avem alături!", and a note "Găsești toate detailele în brosura atașata acestui email." At the bottom are three orange placeholder boxes. At the very bottom of the workspace are Save and Publish buttons.

» Notifications

Managing 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, there is a sidebar with the following navigation items:

- 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
- Platform status

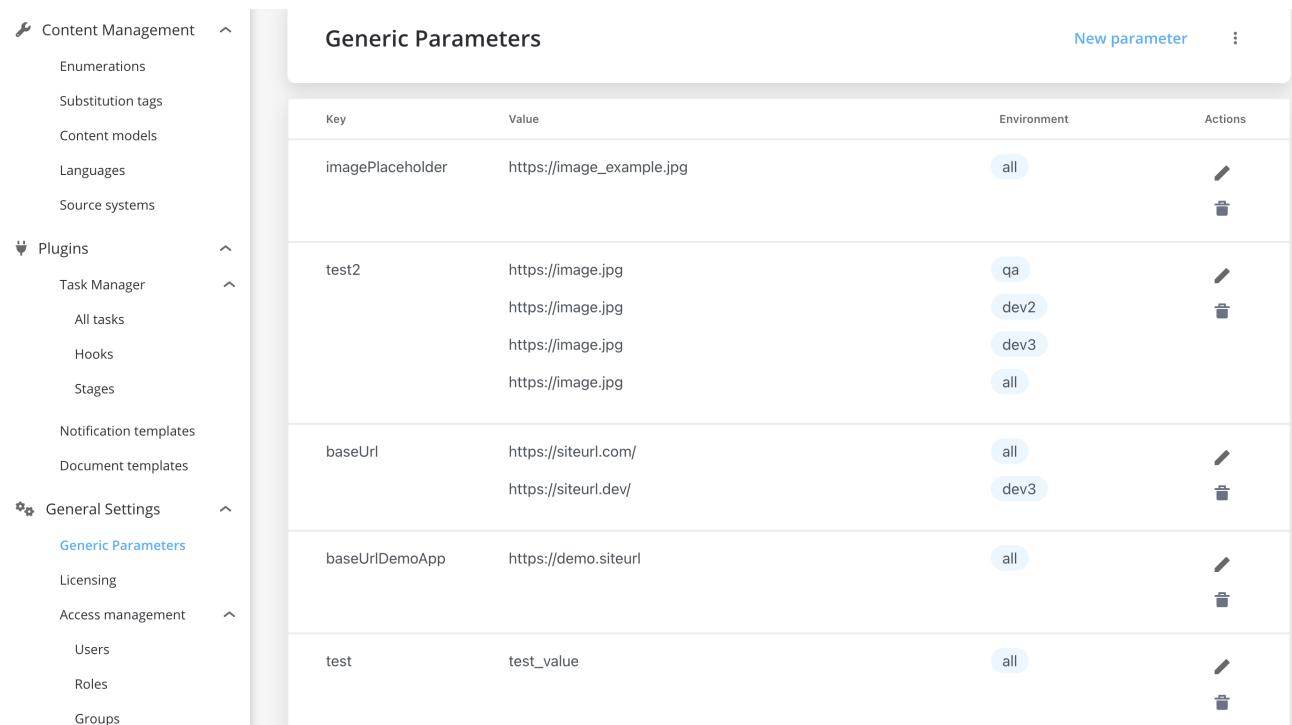
On the right, the main area is titled "Documents Templates - ExampleTemplate". It has tabs for "Body" (selected) and "Data model". The "Body" tab shows a rich text editor toolbar and a preview area containing placeholder text and two input fields labeled "#firstInput" and "#secondInput". The preview area also contains some sample text. At the bottom of the preview area, there are "body p" and "body" buttons. Below the preview area are three buttons: "Test", "Save", and "Publish".

» Document management

Managing generic parameters

Through the

The fallback content to display on prerendering, you can edit generic parameters, and import or export them. You can set generic parameters and assign the environment(s) where they should apply.



The screenshot shows the FLOWX Designer interface with the 'Content Management' section expanded. Under 'General Settings', the 'Generic Parameters' option is selected. The main area displays a table titled 'Generic Parameters' with columns: Key, Value, Environment, and Actions. The table contains several entries:

Key	Value	Environment	Actions
imagePlaceholder	https://image_example.jpg	all	 
test2	https://image.jpg	qa	 
	https://image.jpg	dev2	 
	https://image.jpg	dev3	 
	https://image.jpg	all	 
baseUrl	https://siteurl.com/	all	 
	https://siteurl.dev/	dev3	 
baseUrlDemoApp	https://demo.siteurl	all	 
test	test_value	all	 

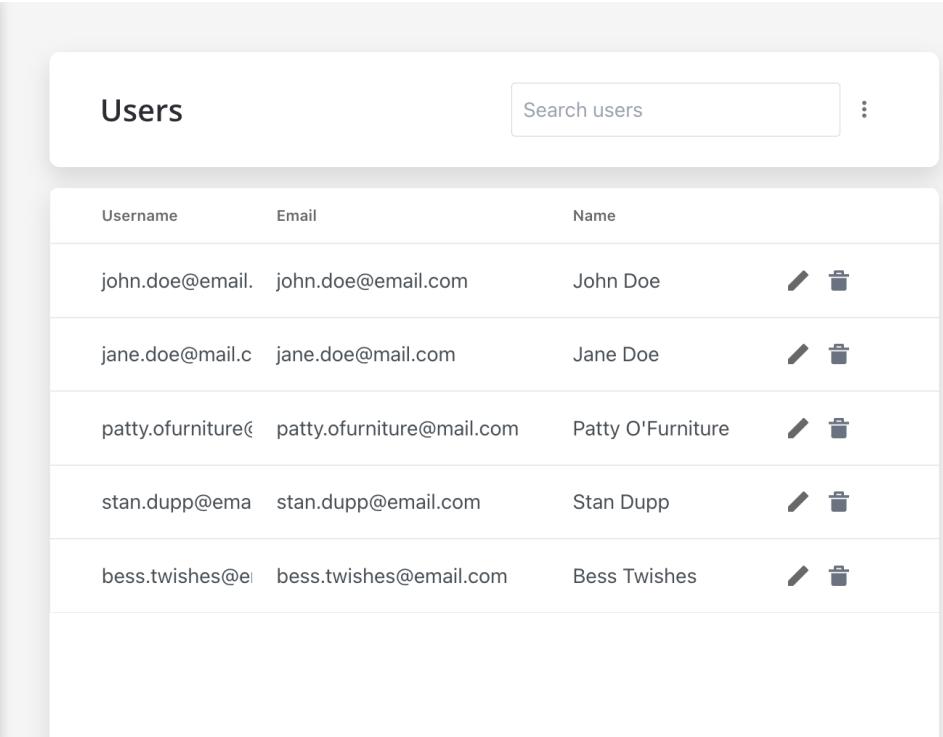
⚠ CAUTION

The maximum length of an input value is 255 characters.

Managing users access

Access Management is used to administrate users, roles and groups, directly in FLOWX Designer. Access Management helps you to access the identity management solution (keycloak/RH-SSO) through its API, extracting all the

necessary details. Access Management is based on user roles that need to be configured in the identity management solution.



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

- Processes
- Content Management
- Plugins
- General Settings
 - Generic Parameters
 - Licensing
- Access management
 - Users** (this item is currently selected)
 - Roles
 - Groups
- Platform status

The main content area is titled "Users" and contains a table with the following data:

Username	Email	Name	Actions
john.doe@email.com	john.doe@email.com	John Doe	
jane.doe@mail.com	jane.doe@mail.com	Jane Doe	
patty.ofurniture@mail.com	patty.ofurniture@mail.com	Patty O'Furniture	
stan.dupp@email.com	stan.dupp@email.com	Stan Dupp	
bess.twishes@email.com	bess.twishes@email.com	Bess Twishes	

» [Configuring access rights for admin](#)

Managing integrations

Integration management enables you to keep track of each integration and its correspondent component and different scenarios used: creating an OTP, document generation, notifications, etc.

The screenshot shows the FlowX.AI platform's left sidebar with a tree view of management categories: Content management, Plugins, General Settings, Integration management (which is expanded), Licensing, and Access management. A user profile for 'John Doe' is at the bottom of the sidebar. The main area is titled 'Integrations' and lists five entries:

	Name	Identifier	System name	Actions
>	Integration 1	Intgr	01	
>	Document Pl...	documentPlu...	FlowX	
>	Notification Pl...	notificationPl...	FlowX	
>	Integration Test	test	System	
>	test_integrati...	integrationtest	Flowx	

Checking platform status

You can quickly check the health status of all the

The fallback content to display on prerendering
and all of your custom connectors.

The screenshot shows the FLOWX.AI Platform Status 2.7.0 interface. On the left, there is a sidebar with the following navigation items:

- Content models
- Languages
- Source systems
- Plugins
 - Task Manager
 - All tasks
 - Hooks
 - Stages
 - Notification templates
 - Document templates
- General Settings
 - Generic Parameters
 - Licensing
 - Access management
 - Users
 - Roles
 - Groups
- Platform status

The main content area is titled "Platform Status 2.7.0" and contains two tables:

Flowx Components

Component name	Status	Installed version	Suggested version
Component 1	UP	0.3.40	0.3.40
Component 2	UP	0.2.23	0.2.23
Component 3	UP	1.0.37	1.0.37

Other Components

Component name	Status	Installed version
Custom Component	UP	0.1.22
Custom component 1	DOWN	-

Check the next section to learn how to create and manage a process from scratch:

» [Managing a process flow](#)

Was this page helpful?

FLOWX.AI DESIGNER / Managing a process flow / Creating a new process definition

The first step of defining your business process in the

The fallback content to display on prerendering
is adding a new **process definition** for it.

This should include at least one **START** and **END** node.

Steps for creating a new process definition

To create a new

The fallback content to display on prerendering

:

1. Open
The fallback content to display on prerendering
and go to the **Definitions** tab.
2. Click the **New process** button, using the **breadcrumbs** from the top-right corner.
3. Enter a unique name for your process and click **Create**.
4. You're automatically taken to the **FLOWX Designer** editor where you can start building your process.

The screenshot shows the FLOWX.AI platform interface for managing process definitions. On the left, there is a sidebar with various navigation options under categories like Processes, Content Management, Plugins, General Settings, and Platform status. The main area is titled "Process Definitions" and contains two tables:

- Drafts / In progress:** This section lists five process definitions:

Name	Version	Edited at	Edited by
demo_3	1	19 May 2022, 4:03 PM	Dragos Caravasile
tibi_test_gap	1	19 May 2022, 3:59 PM	QA FlowX
tibi_testMandatoryKey	1	19 May 2022, 12:19 PM	QA FlowX
ccc_mex_proces_principal_1652945...	1	19 May 2022, 10:32 AM	serban chiricescu
ccc_mex_proces_principal_1652878...	1	18 May 2022, 3:52 PM	serban chiricescu
- Published:** This section lists four published process definitions:

Name	Version	Published at	Published by
ccc_mex_proces_principal	27	18 May 2022, 2:35 PM	serban chiricescu
testVlad6	1	09 May 2022, 5:06 PM	QA FlowX
Vlad_Process	1	02 May 2022, 12:20 PM	QA FlowX
silviu_update_user	1	08 Apr 2022, 10:40 AM	silviu grigore
silviu_client_identification	1	08 Apr 2022, 9:43 AM	silviu grigore

At the bottom left of the main area, there is a small watermark that says "MADE WITH GIFOX.GIF".

In the following section, you will learn how to add a new node to your newly created process.

Was this page helpful?

FLOWX.AI DESIGNER / Managing a process flow / Adding a new node

Once you create a new

The fallback content to display on prerendering

, you can start configuring it by adding new

The fallback content to display on prerendering

You can choose between a series of available node types below. For an overview of what each node represents, see [BPMN 2.0 basic concepts](#):

- [start event](#)
- [end event](#)
- [service task](#)
- [user task](#)
- [parallel gateway](#)
- [exclusive gateway](#)
- [message send event](#)
- [message receive event](#)
- [start milestone](#)
- [end milestone](#)

Steps for creating a new node

To create a new node on an existing process:

1. Open

The fallback content to display on prerendering

and from the **Processes** tab select **Definitions**.

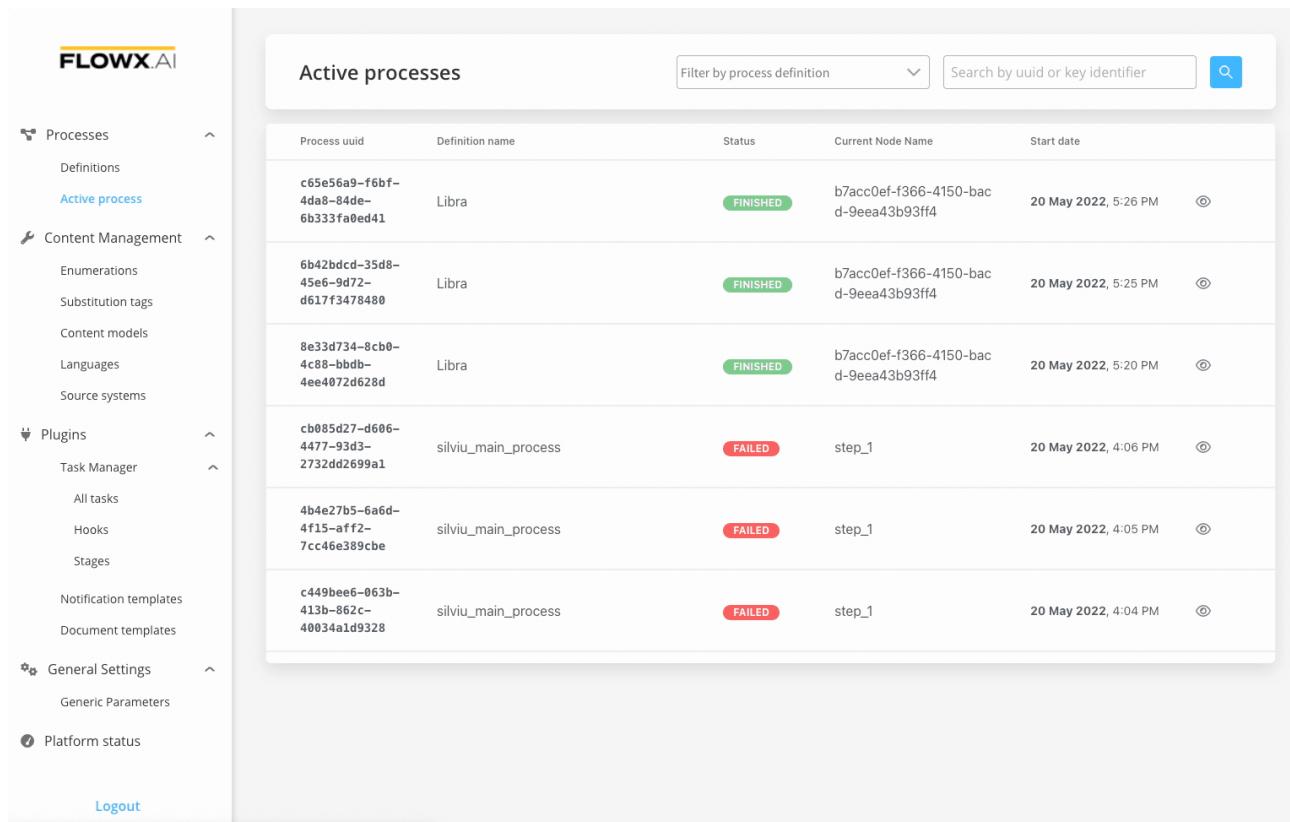
2. Open your **process**.

3. Click the **Edit process** button from the process definition.

4. Drag and drop one **node element**.

5. To connect the node that you just created:

- Click the node, select the **arrow** command
- Click the node that you wish to link to the newly added node



The screenshot shows the FLOWX.AI platform's user 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), General Settings (Notification templates, Document templates), and Platform status. At the bottom of the sidebar are Logout and Help links. The main area is titled "Active processes" and contains a table with the following data:

Process uid	Definition name	Status	Current Node Name	Start date	Action
c65e56a9-f6bf-4da8-84de-6b333fa0ed41	Libra	FINISHED	b7acc0ef-f366-4150-bacd-9eea43b93ff4	20 May 2022, 5:26 PM	⋮
6b42bdcf-35d8-45e6-9d72-d617f3478480	Libra	FINISHED	b7acc0ef-f366-4150-bacd-9eea43b93ff4	20 May 2022, 5:25 PM	⋮
8e33d734-8cb0-4c88-bbdb-4ee4072d628d	Libra	FINISHED	b7acc0ef-f366-4150-bacd-9eea43b93ff4	20 May 2022, 5:20 PM	⋮
cb085d27-d606-4477-93d3-2732dd2699a1	silviu_main_process	FAILED	step_1	20 May 2022, 4:06 PM	⋮
4b4e27b5-6a6d-4f15-aff2-7cc46e389cbe	silviu_main_process	FAILED	step_1	20 May 2022, 4:05 PM	⋮
c449bee6-063b-413b-862c-4003a1d9328	silviu_main_process	FAILED	step_1	20 May 2022, 4:04 PM	⋮

For each new

The fallback content to display on prerendering

, you can set its name, and a set of values (timeout, topic name, key name) and you can also add various **actions** to it.

» Node

Now, check the next section to learn how to add an action to a node.

[Was this page helpful?](#)

FLOWX.AI DESIGNER / Managing a process flow / Adding an action to a node

We use actions to add business decisions to the

The fallback content to display on prerendering

or link the process to custom integrations and

The fallback content to display on prerendering

For more information about actions, check the following section:

» Actions

Steps for creating an action

To create an action:

1. Open **FLOWX Designer** and from the **Processes** tab select **Definitions**.
2. Select your **process**.
3. Click the **Edit process** button.
4. Add a new **node**/ edit an existing one.
5. **!** A few **action parameters** will need to be filled in depending on the selected action type.
6. Add **an action** to the **task node**, for example **sending a Kafka event** to an integration:
 - set **Address** (topic) name
 - set message content to be sent
 - `{"test" : ${processInstanceId}, "title": "Test title"}`
 - a custom header is always set by default to
 - `{"processInstanceId": ${processInstanceId}}`
 - if you want some values to be replaced before sending them to Kafka, they should be marked as such in the action params

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), General Settings (Generic Parameters), Platform status, and Logout. The main area is titled "Process Definitions" and contains two tables:

Drafts / In progress			
Name	Version	Edited at	Edited by
demo_5	1	19 May 2022, 5:16 PM	Dragos Caravasile
deleteActionsParentBug	1	19 May 2022, 4:31 PM	Ciprian Forcos
tibi_test_gap	1	19 May 2022, 3:59 PM	QA FlowX
tibi_testMandatoryKey	1	19 May 2022, 12:19 PM	QA FlowX
ccc_mex_proces_principal...	1	19 May 2022, 10:32 AM	serban chiricescu

Published			
Name	Version	Published at	Published by
client_identification_refactor	4	17 Jan 2022, 1:06 PM	silviu grigore
ccc_mex_proces_principal	27	18 May 2022, 2:35 PM	serban chiricescu
testVlad6	1	09 May 2022, 5:06 PM	QA FlowX
Vlad_Process	1	02 May 2022, 12:20 PM	QA FlowX
silviu_update_user	1	08 Apr 2022, 10:40 AM	silviu grigore

!(INFO)

The nodes that support actions are task nodes, user task nodes, and message send nodes.

Was this page helpful?

FLOWX.AI DESIGNER / Managing a process flow / Handling decisions in the flow

To add business decisions in the flow and use them to pick between a flow branch or another, we can use **exclusive gateways**.



Steps for creating a flow with exclusive branches

To create flow with exclusive branches:

1. Open

The fallback content to display on prerendering and go to the **Definitions** tab.

2. Click on the **New process** button, using the **breadcrumbs** from the top-right corner.

3. Add a **start node** and an **exclusive gateway node**.

4. Add two different **task nodes** and link them after the **exclusive gateway node**.

5. Add a new **exclusive gateway** to merge the two flow branches back into one branch.

6. Add a **new rule** to a node to add a **business decision**, for example:

- select a **scripting language** from the dropdown
- `input.get("application.client.creditScore") >= 700` ← proceed to node for premium credit card
- `input.get("application.client.creditScore") < 700` ← proceed to node for standard credit card

7. Add a **closing exclusive gateway** to continue the flow.

8. Add and **end node**.

For **business rules**, you need to check certain values from the process and pick an outgoing node in case the condition is met. The gateway node must be connected to the next nodes before configuring the rule.

Process Definitions

Search by process definition name

Name	Version	Edited at	Edited by
gateway_decisions_docs	1	23 May 2022, 11:11 AM	Dragos Caravasile
demo_5	1	19 May 2022, 5:16 PM	Dragos Caravasile
deleteActionsParentBug	1	19 May 2022, 4:31 PM	Ciprian Forcos
tibi_test_gap	1	19 May 2022, 3:59 PM	QA FlowX
tibi_testMandatoryKey	1	19 May 2022, 12:19 PM	QA FlowX

Name	Version	Published at	Published by
client_identification_refactor	4	17 Jan 2022, 1:06 PM	silviu grigore
ccc_mex_proces_principal	27	18 May 2022, 2:35 PM	serban chiricescu
testVlad6	1	09 May 2022, 5:06 PM	QA FlowX
Vlad_Process	1	02 May 2022, 12:20 PM	QA FlowX
silviu_update_user	1	08 Apr 2022, 10:40 AM	silviu grigore

[Logout](#)

» Exclusive Gateway Node

Was this page helpful?

FLOWX.AI DESIGNER / Managing a process flow / Adding more flow

branches

To split the

The fallback content to display on prerendering into more steps, you just need to use a **parallel gateway** node type.



Steps for creating a flow with two branches

To create a flow with two branches:

1. Open

The fallback content to display on prerendering and go to the **Definitions** tab.

2. Click on the **New process** button, using the **breadcrumbs** from the top-right corner.

3. Add a **start node** and a **parallel gateway node**.

4. Add two different **task nodes** and link them after the **parallel gateway node**.

5. Add a **parallel gateway** to merge the two flow branches back into one branch.

6. Add an **end node**.

The screenshot shows the FLOWX.AI platform's process management interface. On the left, a sidebar navigation includes sections for Processes (Definitions, Active process), Content Management (Enumerations, Substitution tags, Content models, Languages, Source systems), Plugins (Task Manager, All tasks, Hooks, Stages), General Settings (Generic Parameters), and Platform status. A Logout link is also present. The main content area is titled "Process Definitions" and contains two tables: "Drafts / In progress" and "Published".

Drafts / In progress

Name	Version	Edited at	Edited by	Actions
AutoTestProcess834506323	1	23 May 2022, 1:41 PM	QA FlowX	
TA_AccessRights	1	23 May 2022, 1:38 PM	QA FlowX	
AutoTestProcess144847353	1	23 May 2022, 12:53 PM	QA FlowX	
AutoTestProcess-10960716...	1	23 May 2022, 12:52 PM	QA FlowX	
AutoTestProcess1277519091	1	23 May 2022, 12:51 PM	QA FlowX	

Published

Name	Version	Published at	Published by	Actions
AutoTestProcess-857687723	1	23 May 2022, 1:40 PM	QA FlowX	
AutoTestProcess95481841	1	23 May 2022, 12:49 PM	QA FlowX	
silviu_update_user	3	23 May 2022, 11:46 AM	silviu grigore	
client_identification_refactor	4	17 Jan 2022, 1:06 PM	silviu grigore	
ccc_mex_proces_principal	27	18 May 2022, 2:35 PM	serban chiricescu	

» Parallel Gateway

Was this page helpful?

FLOWX.AI DESIGNER / Managing a process flow / Creating a user

interface

Creating a stepper structure

To create a stepper structure:

1. Go to

The fallback content to display on prerendering
and go to the **Definitions** tab.

2. Click on the **New process** button, using the **breadcrumbs** from the top-right corner.
3. Add a **start node**.
4. Add a new **milestone** to describe all the user tasks.
5. Add a **user task** that will represent the first card of our step.
6. Add an **end milestone** for both the step and the stepper.
7. End your process with an **end node**.

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

- Processes
 - Definitions (selected)
 - 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

The main content area is titled "Process Definitions". It contains two sections: "Drafts / In progress" and "Published".

Drafts / In progress:

Name	Version	Edited at	Edited by	Actions
Stepper Structure	1	24 May 2022, 11:41 AM	Dragos Caravasile	
TA_ActionProcess_v1	23	24 May 2022, 8:59 AM	QA FlowX	
Andratestgateway	1	23 May 2022, 11:32 AM	andra popazu	

Published:

Name	Version	Published at	Published by	Actions
AutoTestProcess-147...	1	23 May 2022, 8:53 PM	QA FlowX	
AutoTestProcess145...	1	23 May 2022, 8:49 PM	QA FlowX	
AutoTestProcess212...	1	23 May 2022, 3:10 PM	QA FlowX	

Configuring the UI

- All the UI elements are configured via the
The fallback content to display on prerendering
- Starting from our basic process, we need to set the start of the stepper
template config
- It is enough to open the drag and drop menu and add a Stepper template
config for now
- For the first step, it will be useful to add also a label
- Clear the cache to test what we have until now

The screenshot shows the FLOWX.AI platform's process management interface. On the left, a sidebar navigation includes categories like 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), and General Settings (Generic Parameters, Licensing, Access management). The main area is titled "Process Definitions" and contains two tables:

Drafts / In progress				
Name	Version	Edited at	Edited by	Actions
stepper structure	1	24 May 2022, 12:13 PM	Dragos Caravasile	▶ ⚒ ⋮
swimlane_drag_docs	1	24 May 2022, 10:24 AM	Dragos Caravasile	▶ ⚒ ⋮
AutoTestProcess-801144746	1	23 May 2022, 9:00 PM	QA FlowX	▶ ⚒ ⋮
AutoTestProcess1667954254	1	23 May 2022, 8:58 PM	QA FlowX	▶ ⚒ ⋮

Published				
Name	Version	Published at	Published by	Actions
AutoTestProcess273667156	1	24 May 2022, 12:07 PM	QA FlowX	▶ ⚒ ⋮
AutoTestProcess-116652463	1	23 May 2022, 8:59 PM	QA FlowX	▶ ⚒ ⋮
AutoTestProcess-1433671545	1	23 May 2022, 8:56 PM	QA FlowX	▶ ⚒ ⋮
AutoTestProcess-1599143931	1	23 May 2022, 7:24 PM	QA FlowX	▶ ⚒ ⋮

Testing the flow that we have

1. From the **process definition**, click the **Start process** button.
2. We will not pass any data to this process so an empty object `{}`.
3. Click **Start Process** and you will see the first step.

The screenshot shows the FLOWX.AI interface for managing process definitions. On the left, there is a sidebar with navigation links for Processes (Definitions, Active process), Content Management (Enumerations, Substitution tags, Content models, Languages, Source systems), Plugins (Task Manager, All tasks, Hooks, Stages), and Notification templates. The main area is titled "Process Definitions" and contains two tables:

- Drafts / In progress**:

Name	Version	Edited at	Edited by
Test Process	5	26 May 2022, 4:49 PM	John Doe
Test Process 1	1	25 May 2022, 4:40 PM	QA FlowX
- Published**:

Name	Version	Published at	Published by
Test Process 3	4	26 May 2022, 12:49 PM	Jane Doe
stepper structure	1	24 May 2022, 12:13 PM	John Doe

Each row in the tables includes edit and delete icons.

Adding a card with one input

1. Go to your **user task** (this will represent the **first card** of your step).
2. Add a **CARD** (this is the UI card element).
3. Add a **Form** to it (Form elements group inputs together).
4. Add an **input** into the form:
 - o Configure the **key** - you can use the key to retrieve the data form saved on that element
 - o Configure a **label**

The screenshot shows the FLOWX.AI platform's process management interface. On the left, there is a sidebar with the FLOWX.AI logo at the top, followed by a tree view of categories: Processes (Definitions, Active process), Content Management (Enumerations, Substitution tags, Content models, Languages, Source systems), Plugins (Task Manager), and All tasks. The main area is titled "Process Definitions" and contains two sections: "Drafts / In progress" and "Published".

Drafts / In progress

Name	Version	Edited at	Edited by	Actions
AutoTestProc...	1	24 May 2022, 12:26 PM	QA FlowX	
AutoTestProc...	1	24 May 2022, 12:26 PM	QA FlowX	

Published

Name	Version	Published at	Published by	Actions
stepper struct...	1	24 May 2022, 12:13 PM	Dragos Caravasile	
AutoTestProc...	1	24 May 2022, 12:07 PM	QA FlowX	

Testing our first input

1. Start one more time the process that you just configured.
2. The input is displayed.
3. Test the input.

The screenshot shows the FLOWX.AI platform's process management interface. On the left, there is a sidebar with navigation links: Processes (Definitions, Active process), Content Management (Enumerations, Substitution tags, Content models, Languages, Source systems), and Plugins (Task Manager, All tasks, Hooks, Stages). The main area is titled "Process Definitions" and contains two sections: "Drafts / In progress" and "Published".

Drafts / In progress:

Name	Version	Edited at	Edited by	Actions
AutoTestProcess1667954254	1	23 May 2022, 8:58 PM	QA FlowX	▶ ⚒ ⋮
AutoTestProcess-1295228711	1	23 May 2022, 8:57 PM	QA FlowX	▶ ⚒ ⋮

Published:

Name	Version	Published at	Published by	Actions
stepper structure	1	24 May 2022, 12:13 PM	Dragos Caravasile	▶ ⚒ ⋮
AutoTestProcess273667156	1	24 May 2022, 12:07 PM	QA FlowX	▶ ⚒ ⋮

Adding second input and a submit action

1. Go to your **user task** node and add a new input via **Node UI designer**.
2. Now go back to the process and add a new **action rule**:
 - first we need to configure the action - the action is called when the button is pressed - the action should be **Manual** (not automatic because it is triggered by a user)
 - we need to keep in mind the name of the action - `saveDataFirstStep`
3. Go back to the Node UI designer and add a button (we need to link the **button** to the **action** based on the name).

The screenshot shows the FLOWX.AI interface with the sidebar collapsed. The main area is titled "Process Definitions". It contains a search bar and a table for "Drafts / In progress". The table has columns for Name, Version, Edited at, and Edited by. Two entries are listed:

Name	Version	Edited at	Edited by
AutoTestProcess-115163...	1	24 May 2022, 12:26 PM	QA FlowX
AutoTestProcess-410150...	1	24 May 2022, 12:13 PM	QA FlowX

Below this is a section titled "Published" with a similar table structure, showing one entry:

Name	Version	Published at	Published by
stepper structure	1	24 May 2022, 12:13 PM	Dragos Caravasile

This screenshot is identical to the one above, but the cursor is hovering over the "Edited at" link for the first entry in the "Drafts / In progress" table.

Was this page helpful?

FLOWX.AI DESIGNER / Managing a process flow / Moving a token backwards in a process

ⓘ INFO

What is it? Back in process is a functionality that allows you to go back in a business process redo a series of previous

The fallback content to display on prerendering in the process.

Why is it useful? Brings a whole new level of flexibility in using a business flow or journey, allowing the user to go back a step without losing all the data inputted so far.

In most cases, the

The fallback content to display on prerendering instance will just need to advance forward in the process as the actions on the process are completed.

» Token

But there might be cases when the token will need to be moved backward in order to redo a series of previous actions in the process.

We will call this behavior **resetting the token**.

The token can only be reset to certain actions on certain process nodes. These actions will be marked accordingly with a flag `allowTokenReset`. When such an action is triggered from the application, the current process token will be marked as aborted and a new one will be created and placed on the node that contains the action that was executed. If any sub-processes were started between the two token positions, they will also be aborted when the token is reset.

The newly created token will copy from the initial token all the information regarding the actions that were performed before the reset point.

There are a few configuration options available in order to decide which of the data to keep when resetting the token:

- `restartFromSnapshot` - if set to true, the process parameter values will be reset to the values they had before the first execution of that action
- `keysForRestart` - an array of process parameter values keys to be copied from the initial execution of that process part

Allow BACK on this action?

Back in steps

When reset from this action, recreate state?

Copy from current state this objects

Was this page helpful?

FLOWX.AI DESIGNER / Managing a process flow / Exporting / importing a process definition

To copy

The fallback content to display on prerendering and move them between different environments, you can use the export/import feature.

Export a process definition

You can export a process definition as a JSON file directly from the process definitions list in the

The fallback content to display on prerendering

:

The screenshot shows the FLOWX.AI interface with a sidebar on the left containing navigation links for Processes, Content Management, Plugins, and Task Manager. The main area is titled 'Process Definitions' and displays two entries under 'Drafts / In progress': 'AutoTest...' (Version 1) and 'AutoTest...' (Version 2). Both entries were edited on 23 May 2022, 8:57 PM by QA FlowX. Below this, there is a section titled 'Published' which contains one entry: 'AutoTest...' (Version 1), published on 23 May 2022, 8:59 PM by QA FlowX. A search bar and a more options button are located at the top right of the main content area.

Name	Version	Edited at	Edited by
AutoTest...	1	23 May 2022, 8:57 PM	QA FlowX
AutoTest...	2	23 May 2022, 8:57 PM	QA FlowX

Name	Version	Published at	Published by
AutoTest...	1	23 May 2022, 8:59 PM	QA FlowX

Import a process definition

You can then import the process definition from the FLOWX Designer. This will add a new definition to your database and set it as a draft. You can then check the imported definition and publish the process to be able to use it from the

The fallback content to display on prerendering

The screenshot shows the FLOWX.AI interface. On the left is a sidebar with categories: Processes (Definitions, Active process), Content Management (Enumerations, Substitution tags, Content models, Languages, Source systems), Plugins (Task Manager, All tasks, Hooks). The main area is titled "Process Definitions" and contains two tables:

Drafts / In progress			
Name	Version	Edited at	Edited by
AutoTestPro...	1	27 May 2022, 11:42 AM	QA FlowX
TA_ActionP...	39	27 May 2022, 9:23 AM	QA FlowX

Published			
Name	Version	Published at	Published by
AutoTestPro...	1	24 May 2022, 2:59 PM	QA FlowX
AutoTestPro...	1	23 May 2022, 8:53 PM	QA FlowX

In case a definition with the same name and version already exists in the environment, the version of the imported process will be incremented.

⚠ CAUTION

In case the process definition was exported using an incompatible FLOWX Designer version, you will receive an error and not be able to import it. It will first need to be adjusted to match the format needed by your current FLOWX Designer version.

Was this page helpful?

FLOWX.AI DESIGNER / Managing a process flow / Process definition states & versioning

The fallback content to display on prerendering
in

The fallback content to display on prerendering
can be edited with ease. To keep track of updates and changes, each definition is assigned a version number and state.

Process Definitions

Search by process definition name:

Drafts / In progress				
Name	Version	Edited at	Edited by	
Test 1	1	27 May 2022, 12:03 PM	John Doe	
Test 2	5	26 May 2022, 4:49 PM	Jane Doe	

Published				
Name	Version	Published at	Published by	
Test 3	4	26 May 2022, 12:49 PM	John Doe	
Test 4	1	24 May 2022, 12:13 PM	Jane Doe	

A Process Definition is unique and can be identified by its name and version number. It can have one of the following states:

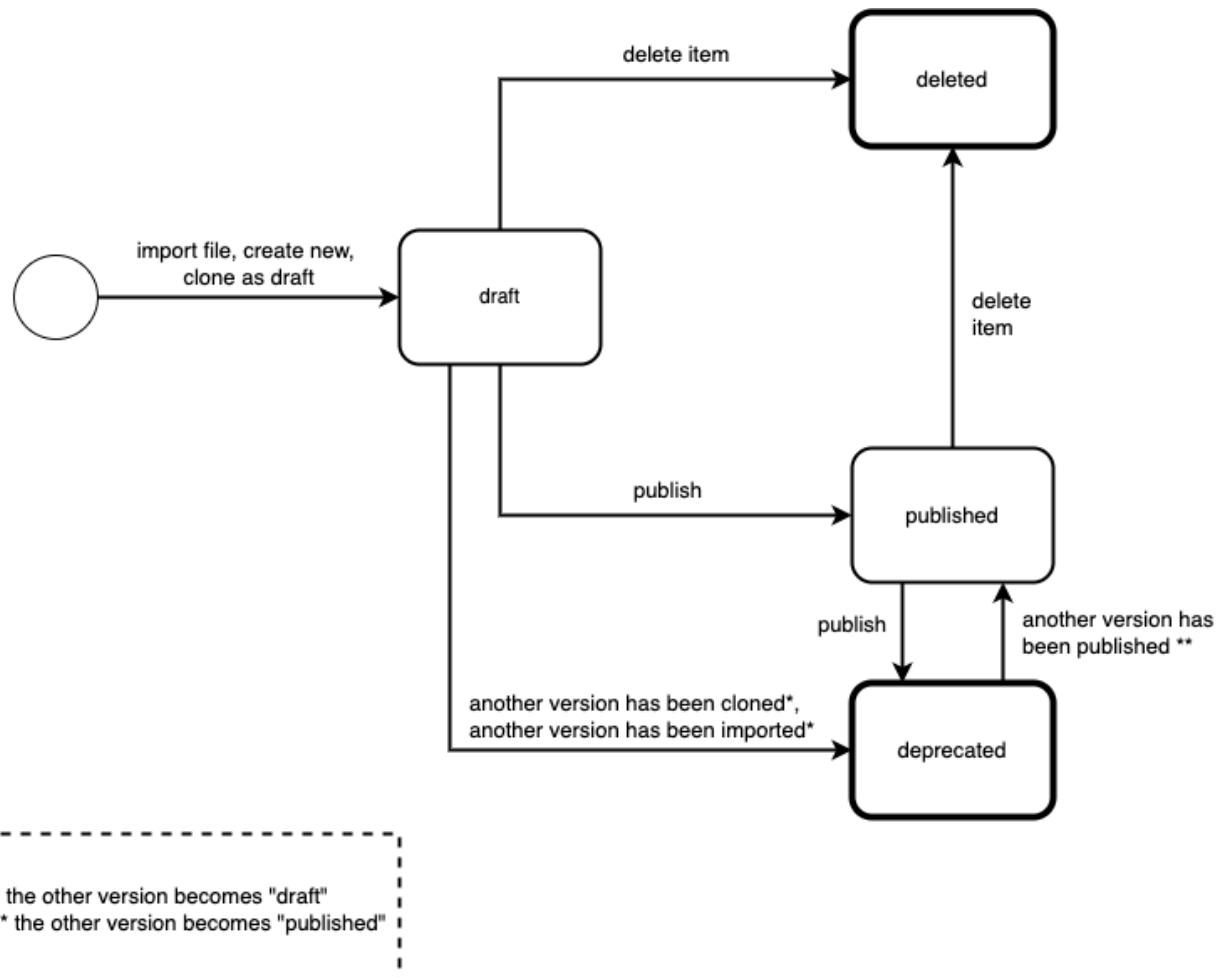
- **draft** - in progress and can't be used to create
The fallback content to display on prerendering
, can only be started in test mode
- **published** - only one version can be published at a time, and it will be used to
instantiate new processes; published versions can be edited
- **deleted** - not visible or usable in the platform
- **deprecated** - old versions

At any time, there can only be one draft and one published version for a specific process definition. A new

The fallback content to display on prerendering
can be created by adding it in FLOWX.AI Designer, importing an existing one, or
cloning an existing definition. A newly created definition will be in the draft state.

Deleting a draft or published version will automatically delete all versions of that definition. When a new version is set to published, the previous published version becomes deprecated. A published version cannot be reset to draft.

When a new version is obtained through cloning or importing, the old versions become deprecated. Importing or cloning a definition increments the maximum version number. If importing/cloning for the first time in a new environment, the definition will be saved with the version number in the imported content. If no version number is present, it will be automatically set to 1.



Testing draft versions

A separate REST endpoint is available for starting process instances from draft process definitions:

▶ **GET** `ENGINE_URL/api/internal/process/processDefinitionId/start`

Was this page helpful?

FLOWX.AI DESIGNER / Designer setup guide / Configuring access rights for Admin

Granular access rights can be configured for restricting access to the Admin component.

Access authorizations are provided, each with specified access scopes:

1. **Manage-platform** - for configuring access for managing platform details

Available scopes:

- **read** - users are able to view platform status
- **admin** - users are able to force health check scan

2. **Manage-processes** - for configuring access for managing process definitions

Available scopes:

- **import** - users are able to import process definitions and process stages
- **read** - users are able to view process definitions and stages
- **edit** - users are able to edit process definitions
- **admin** - users are able to publish and delete process definitions, delete stages, edit sensitive data for process definitions

3. **Manage-configurations** - for configuring access for managing generic parameters

Available scopes:

- **import** - users are able to import generic parameters
- **read** - users are able to view generic parameters
- **edit** - users are able to edit generic parameters
- **admin** - users are able to delete generic parameters

4. **Manage-users** - for configuring access for access management

Available scopes:

- **read** - users are able to read all users, groups and roles
- **edit** - users are able to create/update any user group or roles
- **admin** - users are able to delete users, groups or roles

5. **Manage-integrations** - for configuring integrations with adapters

Available scopes:

- **import** - users are able to import integrations
- **read** - users are able to view all the integrations, scenarios and scenarios configuration(topics/ input model/ output model/ headers)
- **edit** - users are able to create/update/delete any values for integrations/scenarios and also scenarios configuration (topics/input model/ output model/ headers)
- **admin** - users are able to delete integrations/scenarios with all children

The Admin service is configured with the following default users roles for each of the access scopes mentioned above:

- **manage-platform**
 - read:
 - ROLE_ADMIN_MANAGE_PLATFORM_READ
 - ROLE_ADMIN_MANAGE_PLATFORM_ADMIN
 - admin:
 - ROLE_ADMIN_MANAGE_PLATFORM_ADMIN
- **manage-processes**
 - import:
 - ROLE_ADMIN_MANAGE_PROCESS_IMPORT
 - ROLE_ADMIN_MANAGE_PROCESS_EDIT
 - ROLE_ADMIN_MANAGE_PROCESS_ADMIN
 - read:
 - ROLE_ADMIN_MANAGE_PROCESS_READ
 - ROLE_ADMIN_MANAGE_PROCESS_IMPORT
 - ROLE_ADMIN_MANAGE_PROCESS_EDIT
 - ROLE_ADMIN_MANAGE_PROCESS_ADMIN
 - edit:
 - ROLE_ADMIN_MANAGE_PROCESS_EDIT
 - ROLE_ADMIN_MANAGE_PROCESS_ADMIN
 - admin:
 - ROLE_ADMIN_MANAGE_PROCESS_ADMIN
- **manage-configurations**
 - import:
 - ROLE_ADMIN_MANAGE_CONFIG_IMPORT
 - ROLE_ADMIN_MANAGE_CONFIG_EDIT
 - ROLE_ADMIN_MANAGE_CONFIG_ADMIN

- read:
 - ROLE_ADMIN_MANAGE_CONFIG_READ
 - ROLE_ADMIN_MANAGE_CONFIG_IMPORT
 - ROLE_ADMIN_MANAGE_CONFIG_EDIT
 - ROLE_ADMIN_MANAGE_CONFIG_ADMIN
- edit:
 - ROLE_ADMIN_MANAGE_CONFIG_EDIT
 - ROLE_ADMIN_MANAGE_CONFIG_ADMIN
- admin:
 - ROLE_ADMIN_MANAGE_CONFIG_ADMIN
- **manage-users**
 - read:
 - ROLE_ADMIN_MANAGE_USERS_READ
 - ROLE_ADMIN_MANAGE_USERS_EDIT
 - ROLE_ADMIN_MANAGE_USERS_ADMIN
 - edit:
 - ROLE_ADMIN_MANAGE_USERS_EDIT
 - ROLE_ADMIN_MANAGE_USERS_ADMIN
 - admin:
 - ROLE_ADMIN_MANAGE_USERS_ADMIN
- **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 other 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`: `MANAGEPLATFORM`,
`MANAGEPROCESSES`, `MANAGECONFIGURATIONS`, `MANAGEUSERS`.

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

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

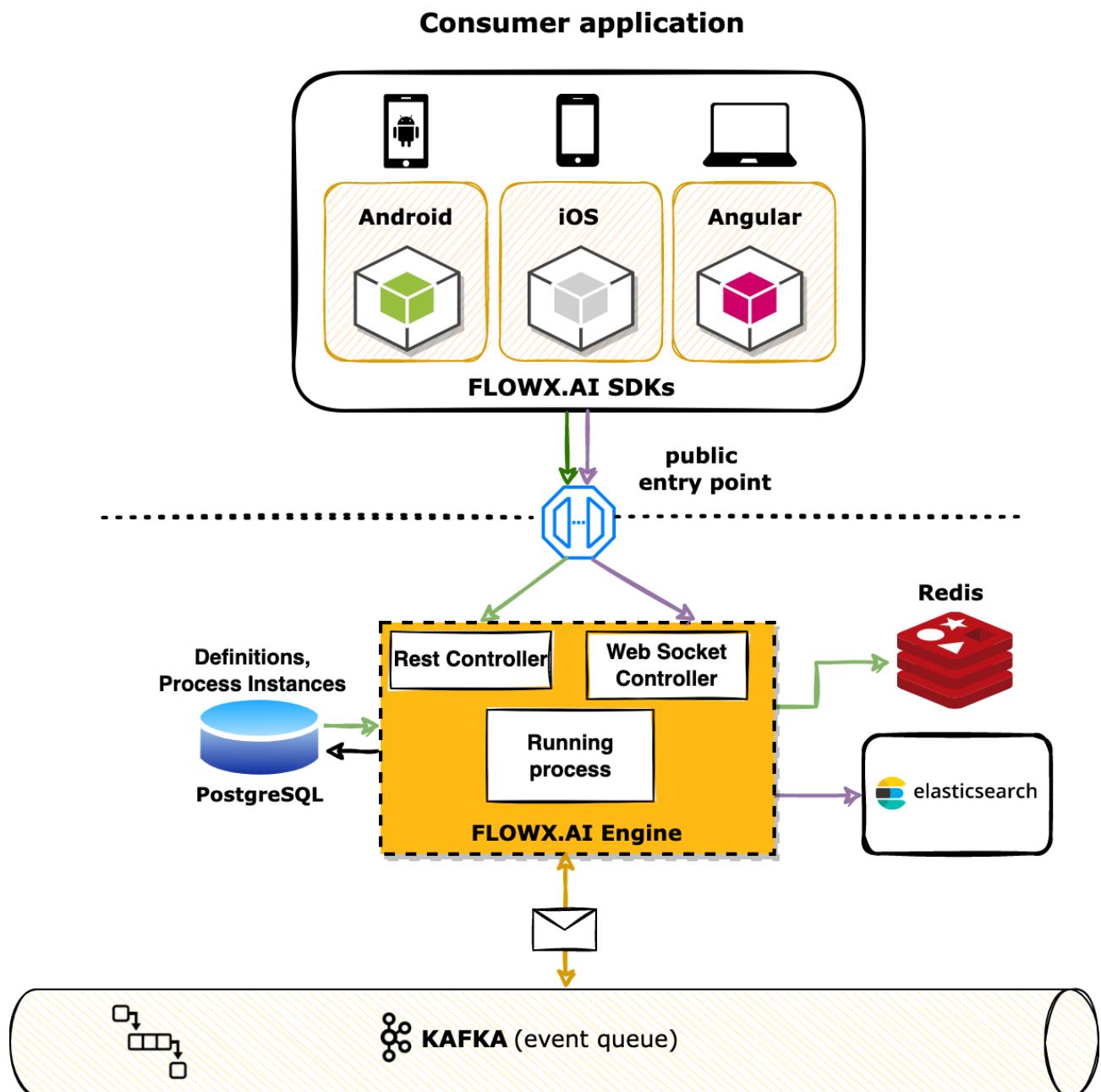
SECURITY_ACCESSAUTHORIZATIONS_MANAGEPROCESSES_SCOPES_READ_ROLES
ROLE_NAME_TEST

Was this page helpful?

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

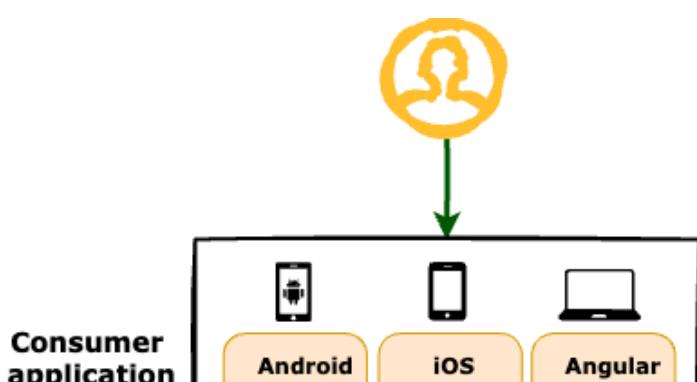
For detailed instructions on how to set up the Advancing Controller microservice, refer to the following guide:

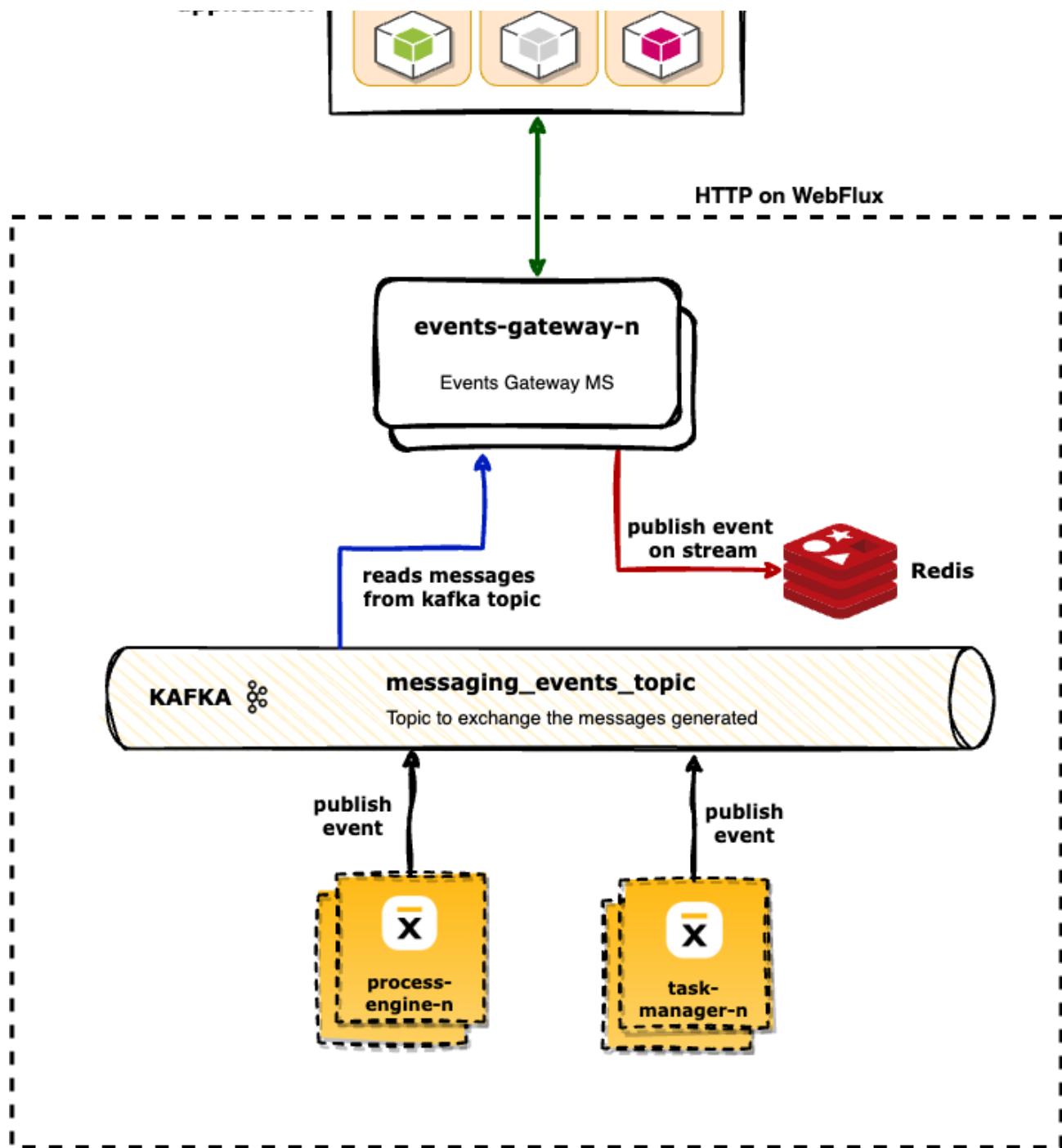
» [Advancing controller setup guide](#)

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**.

The screenshot shows the FLOWX.AI interface with a sidebar on the left and a main content area on the right.

Left Sidebar:

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

Main Content Area:

Enumerations

New enumeration ⋮

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

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...	-	-	-	»
B	Extractiv...	Extractiv...	-	-	-	»
C	Industria ...	Manufact...	-	-	-	»
D	Productio...	Productio...	-	-	-	»

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 list of enumerations on the left and an 'Add new enumeration' form on the right.

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

The 'Add new enumeration' form has a 'Name' input field and a large blue 'Add' button.

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

The screenshot shows the FLOWX.AI interface with a sidebar on the left containing navigation links: Processes (Definitions, Active process), Content Management (Enumerations, Substitution tags, Content models, Languages, Source systems), and Plugins (Task Manager, All tasks, Hooks, Stages). The main content area is titled 'Values for tesEnumerationDocs' and contains a table with the following data:

Code	ro	en	de	FLEX	FLOWX	
test_enumer...	-	-	-	-	-	>> edit delete

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 web application. 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). The main area is titled 'Enumerations' and lists several entries:

Name	Version	Draft	Last upd	Actions
Activity Domain Companies	4	<input type="checkbox"/>	24 Jun	» Delete
Test Enumerations	3	<input type="checkbox"/>	23 Jun	» Delete
Example	4	<input type="checkbox"/>	24 Jun	» Delete
County	5	<input type="checkbox"/>	24 Jun 2022, 5:47 PM	» Delete
Test	3	<input type="checkbox"/>	24 Jun 2022, 5:47 PM	» Delete
Test Monthly Income	3	<input type="checkbox"/>	24 Jun 2022, 5:47 PM	» Delete
Test Activity Domains	3	<input type="checkbox"/>	24 Jun 2022, 5:47 PM	» Delete
Monthly Income	3	<input type="checkbox"/>	24 Jun 2022, 5:47 PM	» Delete

A context menu is open over the 'Activity Domain Companies' entry, showing options: Import (from JSON, from CSV) and 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 type="checkbox"/>	27 Jun 2022, 6:00 PM	»	trash
ActivityDomain_Companies-A-01-01	3	<input type="checkbox"/>	23 Jun 2022, 4:21 PM	»	trash
ActivityDomain_Companies-B-05-062	6	<input type="checkbox"/>	27 Jun 2022, 6:00 PM	»	trash
County	7	<input type="checkbox"/>	27 Jun 2022, 6:00 PM	»	trash
Enumeration_Tesy	1	<input type="checkbox"/>	27 Jun 2022, 1:09 PM	»	trash
StatusFATCA_test	5	<input type="checkbox"/>	27 Jun 2022, 6:00 PM	»	trash
Tara_Test	5	<input type="checkbox"/>	27 Jun 2022, 6:00 PM	»	trash
Test_Andrei	5	<input type="checkbox"/>	27 Jun 2022, 6:00 PM	»	trash
VenitLunar_test	5	<input 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.

Substitution tags		Search by keyword						New value	⋮
	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

INFO

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 interface with a sidebar on the left and a main content area on the right.

Left Sidebar:

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

Main Content Area:

Content Models

New content model

Model name	Edited at	Edited by	Actions
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 dialog box for editing content model attributes. At the top left is the heading "General settings". Below it is a "Name" input field containing "Property name". Under "Attributes", there is a table with columns: Name, Type, Mandatory, and Actions. In the "Name" column, there is an input field with "Property name". In the "Type" column, a dropdown menu is open, showing "String" (which is selected), "Number", and "Boolean". To the right of the dropdown are checkboxes for "Mandatory" (unchecked) and "Actions" (checked, indicated by a blue circle with a checkmark). A blue "+" button is also visible. At the bottom left is a "Close" button, and at the bottom right is a large blue "Save" button.

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 interface. On the left is a sidebar with the following navigation items:

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

The main content area is titled "Content Models" and contains a table with three rows of data:

Model name	Edited at	Edited by	
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.

The screenshot shows the FLOWX.AI interface with a sidebar on the left containing navigation links such as Processes, Content Management, and Languages. The main area is titled "Languages" and displays a table with three rows of language data. The columns are "Code", "Name", and "Default". The data is as follows:

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

Each row has edit and delete icons.

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

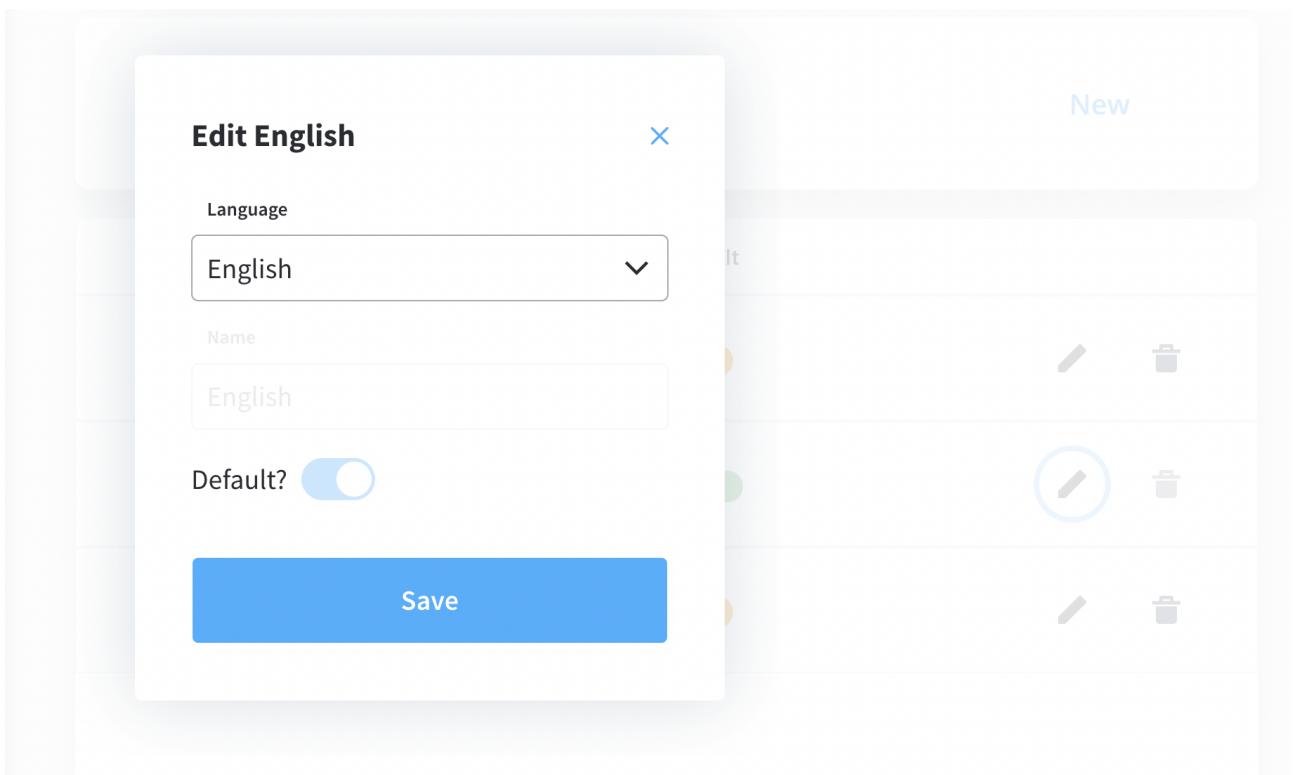
- **Code** - not displayed in the end-user interface, but used to assure value uniqueness
- **Name** - the name of the language
- **Default** - you can set a language as **Default** (default values can't be deleted)

⚠ 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.

en	English	yes	 
de	German	no	  Can't delete default value

- **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 Content Management interface. On the left, there is a sidebar with three main sections: 'Content Management' (expanded), 'Plugins' (expanded), and 'General Settings' (collapsed). Under 'Content Management', the 'Languages' option is highlighted in blue. The main area is titled 'Languages' and contains a table with three rows. The columns are 'Code', 'Name', and 'Default'. The rows are: 1) Code: ro, Name: Romanian, Default: no (indicated by an orange button). 2) Code: en, Name: English, Default: yes (indicated by a green button). 3) Code: de, Name: German, Default: no (indicated by an orange button). Each row has edit and delete icons on the right.

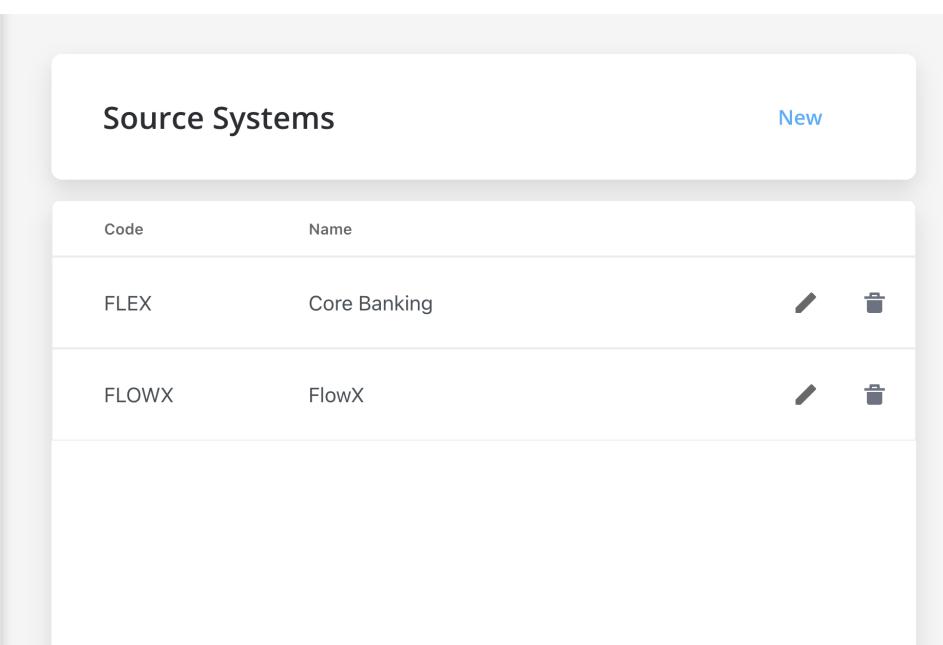
Code	Name	Default		
ro	Romanian	no		
en	English	yes		
de	German	no		

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 platform interface. On the left, there is a sidebar with navigation links: 'Processes' (Definitions, Active process), 'Content Management' (Enumerations, Substitution tags, Content models, Languages, Source systems), and a 'New' button. The main area is titled 'Source Systems' and contains a table with two rows:

Code	Name	Action
FLEX	Core Banking	
FLOWX	FlowX	

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.

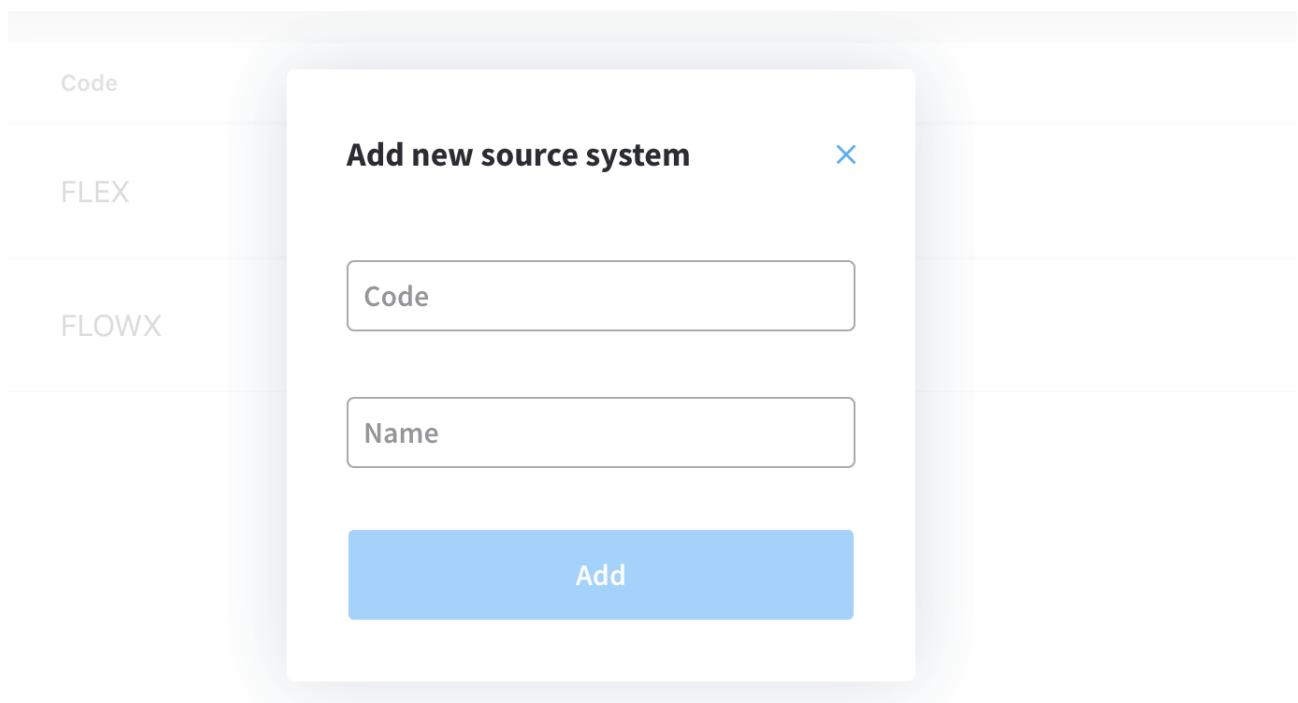
2. Select **Source systems** from the list.

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.

Preview	Key	Format	Size	Edited at	Edited by
	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:

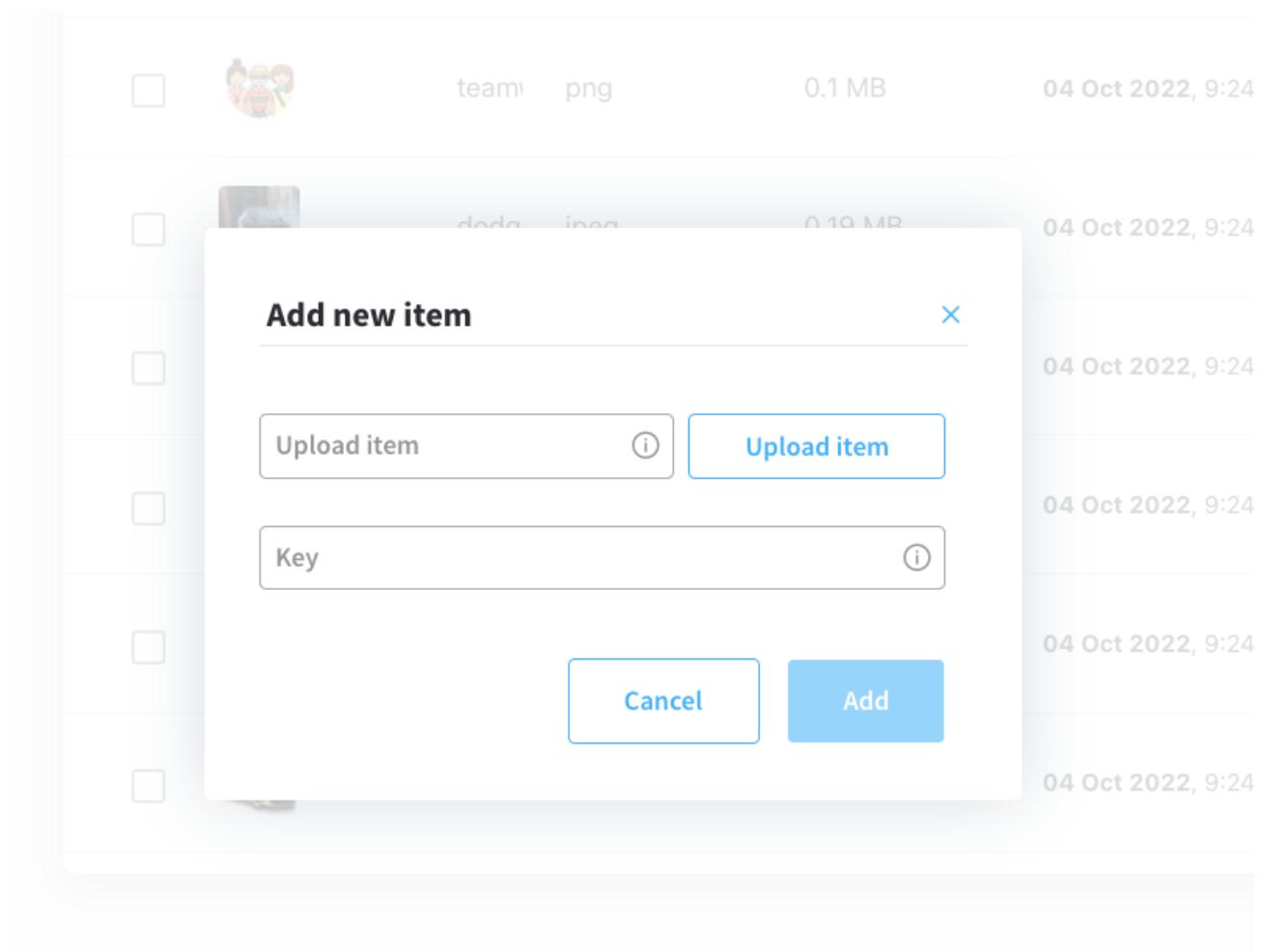
1. Open

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						<input type="text" value="Search item by key"/>	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	 	

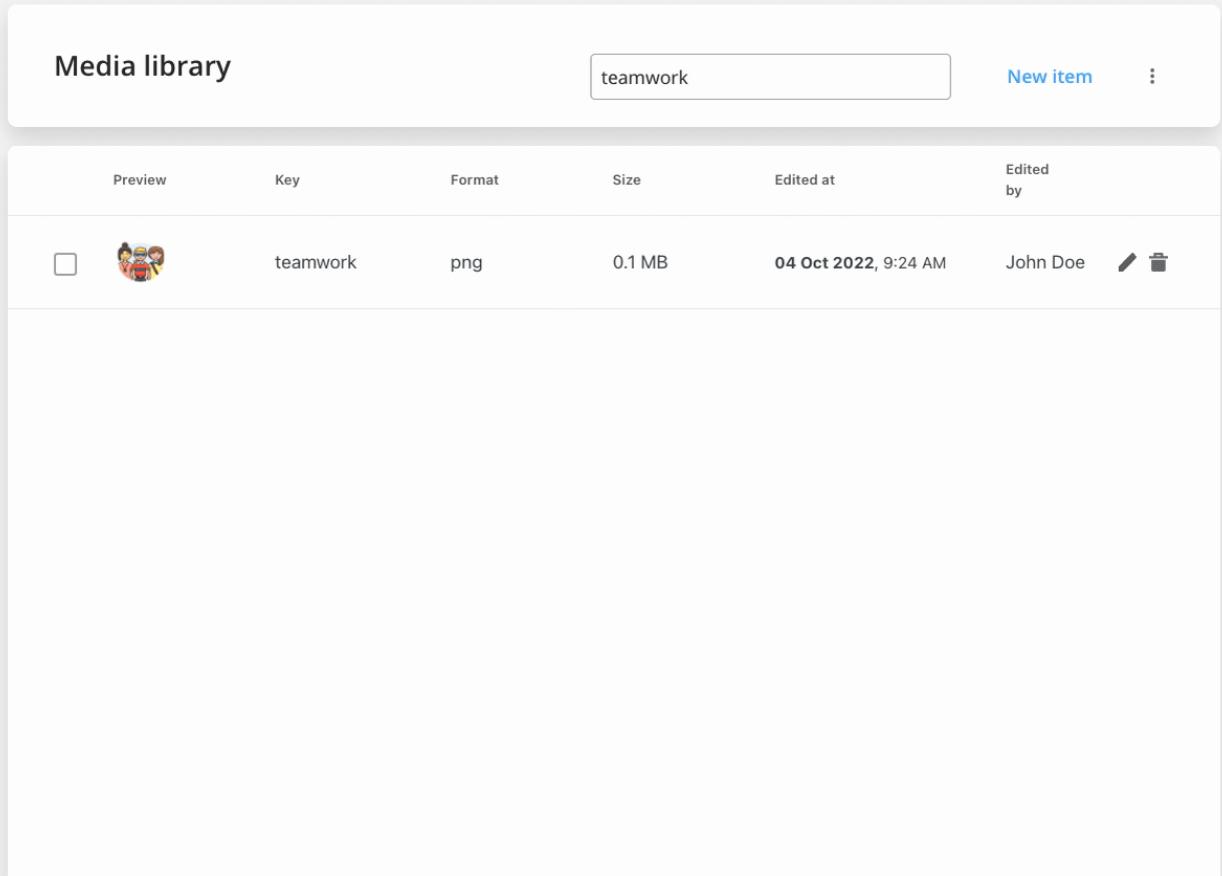
Searching assets

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

Media library						<input type="text" value="teamwork"/>	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:

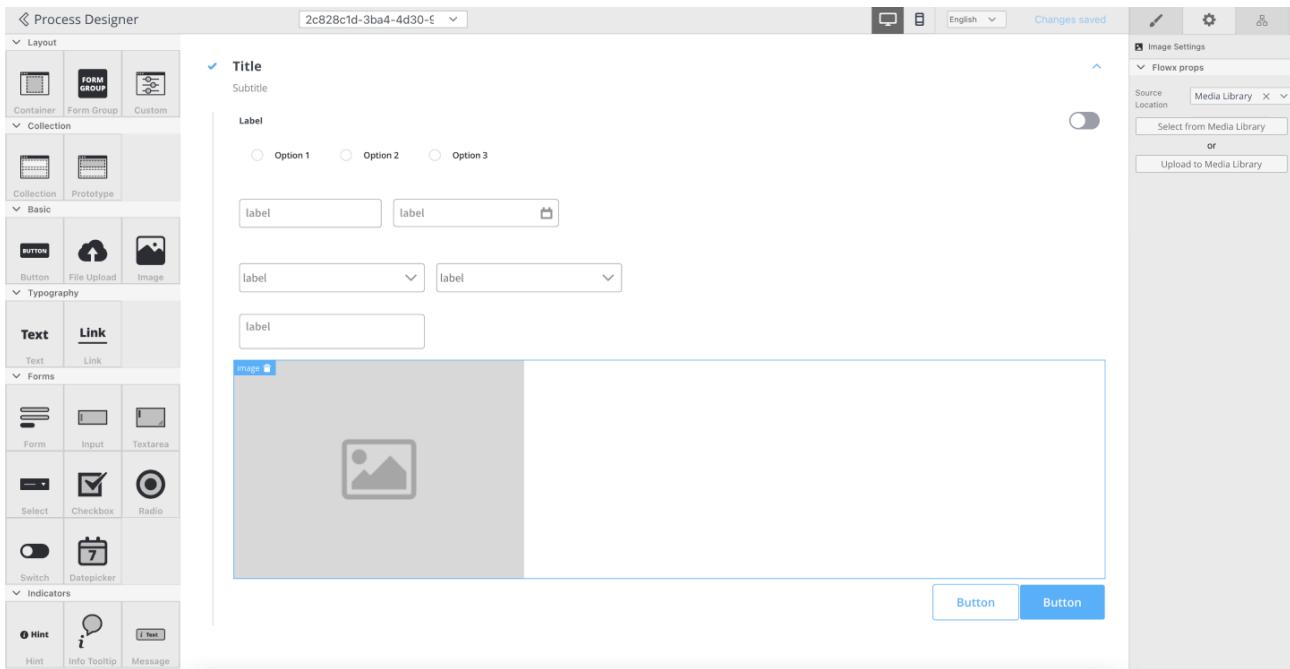
Preview	Key	Format	Size	Edited at	Edited by
<input type="checkbox"/> 	teamwork	png	0.1 MB	04 Oct 2022, 9:24 AM	John Doe  

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

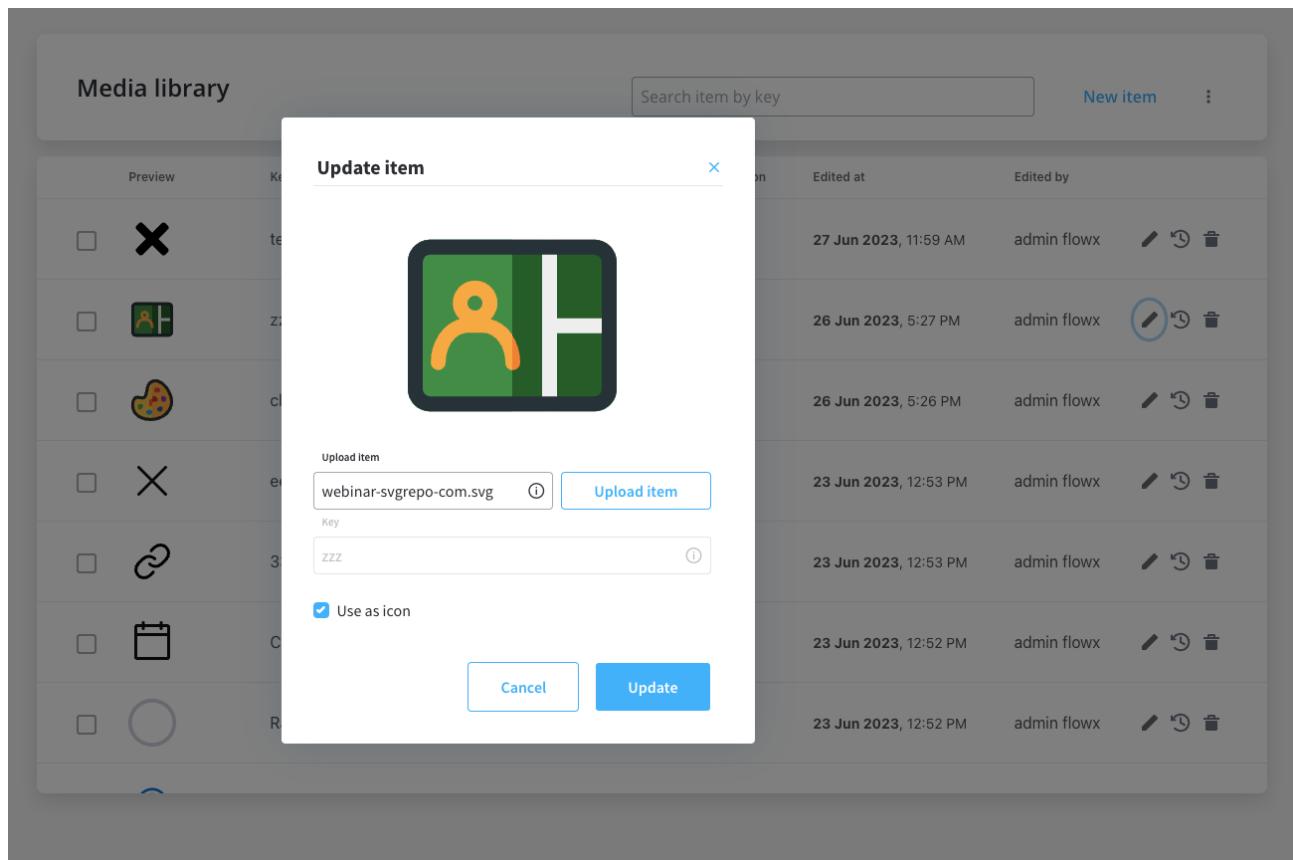


!(INFO)

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

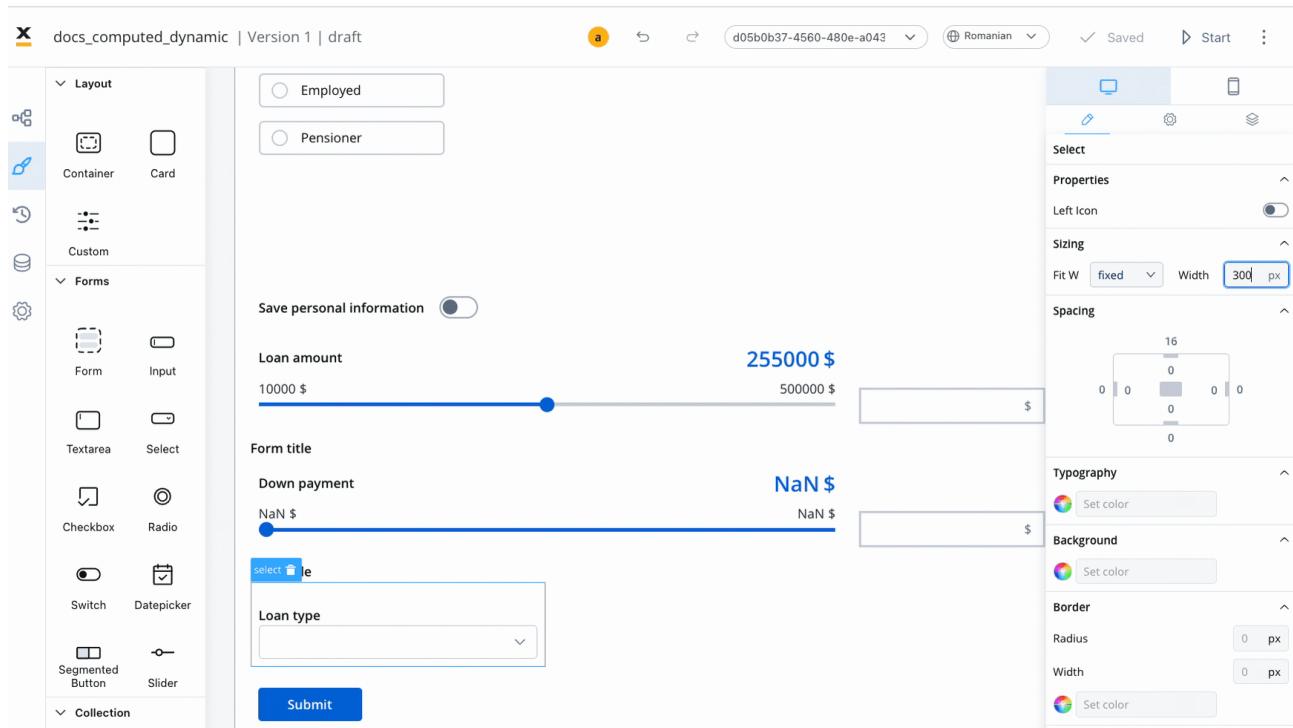
from icon sets that provide these size options or to resize icons proportionally to fit within these dimensions.

⚠ 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.

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/1681945723855566474/maxresdefault.jpg",
      "thumbnailStoragePath":
      "https://d22tnnndi9lo60.cloudfront.net/devmain/flowx/cart/1681945723855566474/thumbnail/maxresdefault.jpg"
    },
    {
      "key": "pizza",
      "application": "flowx",
      "filename": "pizza.jpeg",
      "format": "jpeg",
      "contentType": "image/jpeg",
      "size": 22845,
      "storagePath":
      "https://d22tnnndi9lo60.cloudfront.net/devmain/flowx/pizza/1681945723855566474/pizza.jpeg",
      "thumbnailStoragePath":
      "https://d22tnnndi9lo60.cloudfront.net/devmain/flowx/pizza/1681945723855566474/thumbnail/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 process definitions interface. On the left, there 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".

Drafts / In progress:

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

Published:

Name	Version	Published at	Published by	Actions
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 generic parameters configuration page. On the left, there is a sidebar with navigation links: Notification templates, Document templates, General Settings (expanded, showing Generic Parameters), Integration management, Licensing, Access management (expanded), Users, Roles, Groups, Audit Log, Platform status, and Admin Flowx (with a yellow badge). The main area displays a table of generic parameters:

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

Using generic parameters

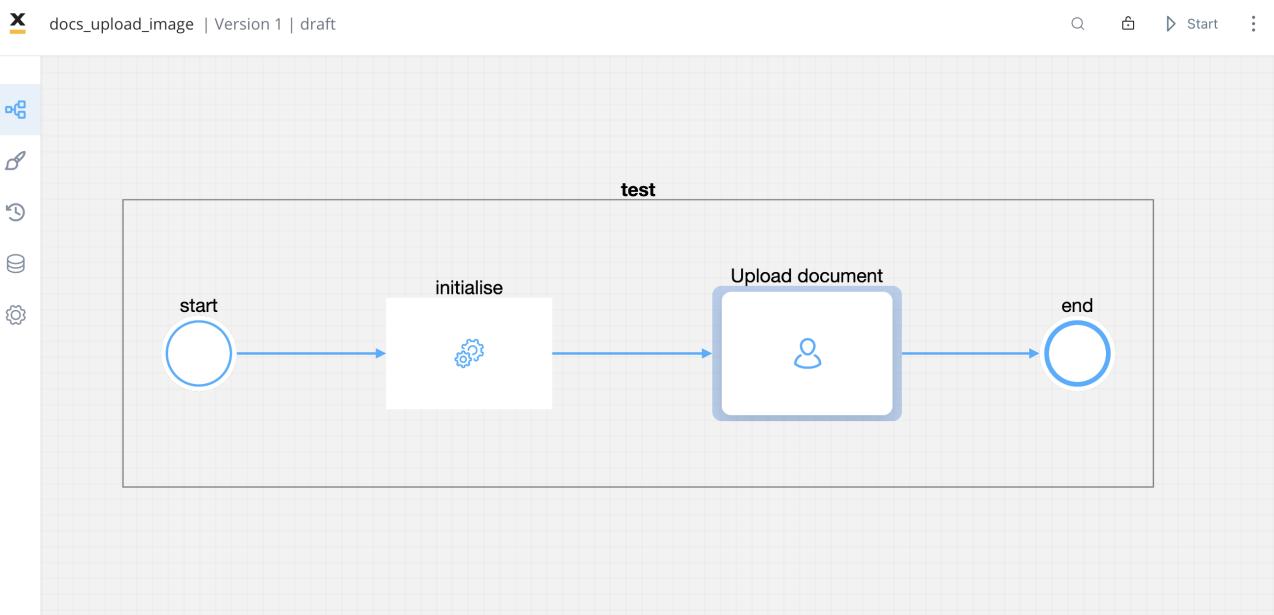
Use case

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

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

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

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



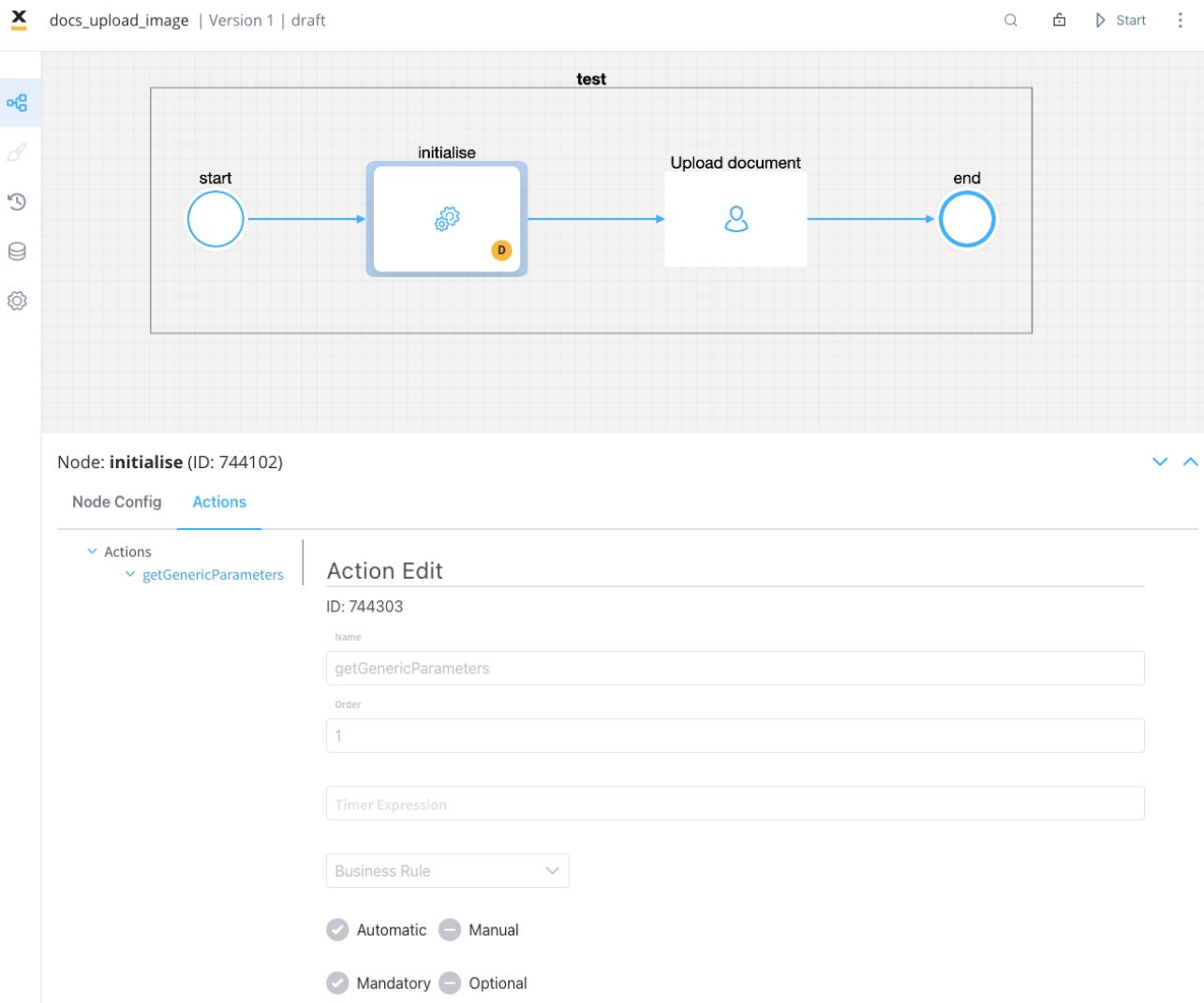
Configuring the task node

Set a

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

Action Edit

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



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 trash

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".

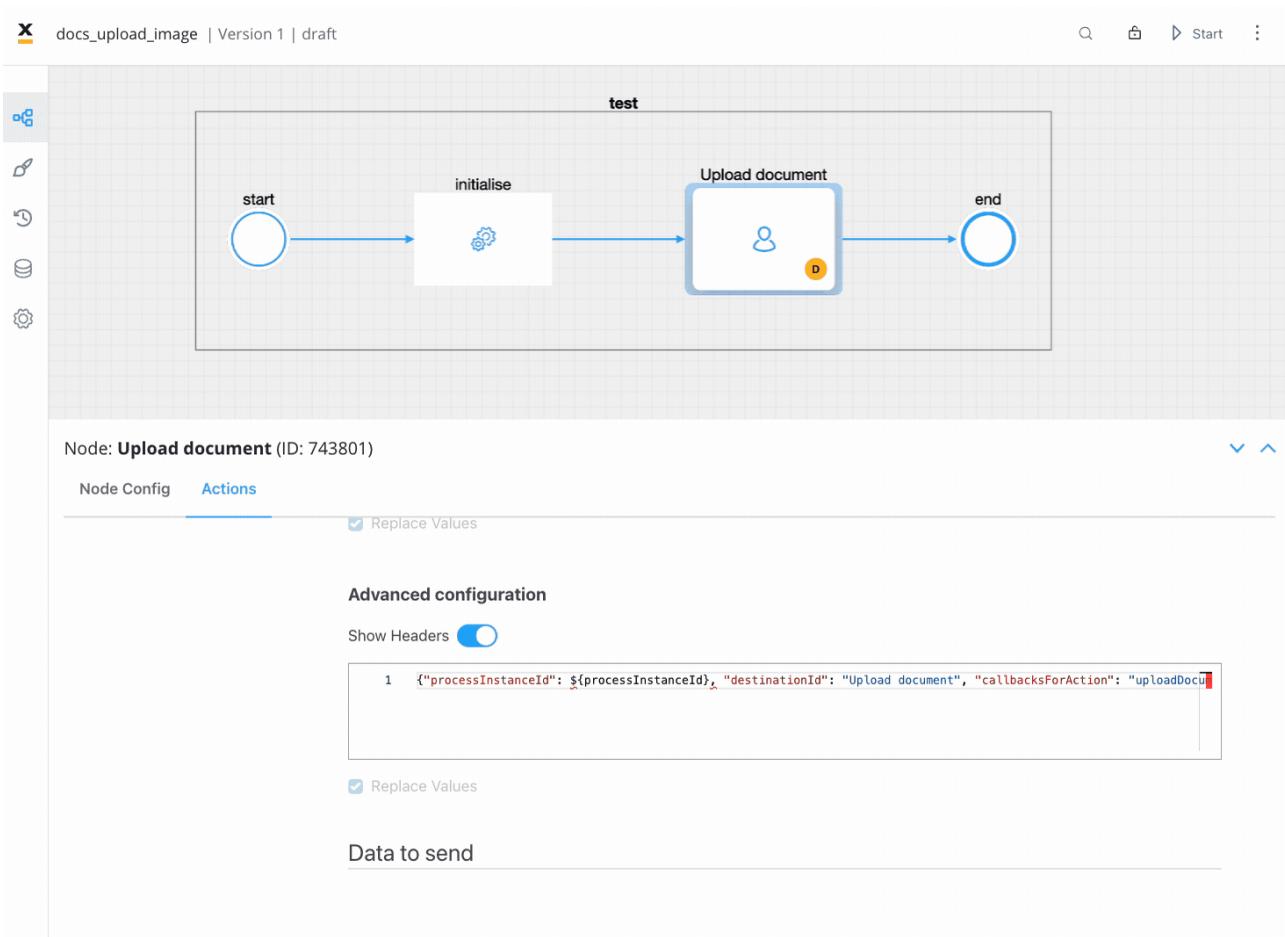
In summary, the code seems to be processing an uploaded document by checking its download path, constructing a full file path, and updating the document object with the new file path. Finally, it outputs the updated document object.

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	
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-ede16ed5c8f5	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	Versioning	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			
Today Clear									

- 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

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.

🔥 DANGER

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"  
}
```

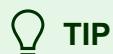
These fields, `applicationName` and `applicationId` are used to uniquely identify a scheduled message.

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:

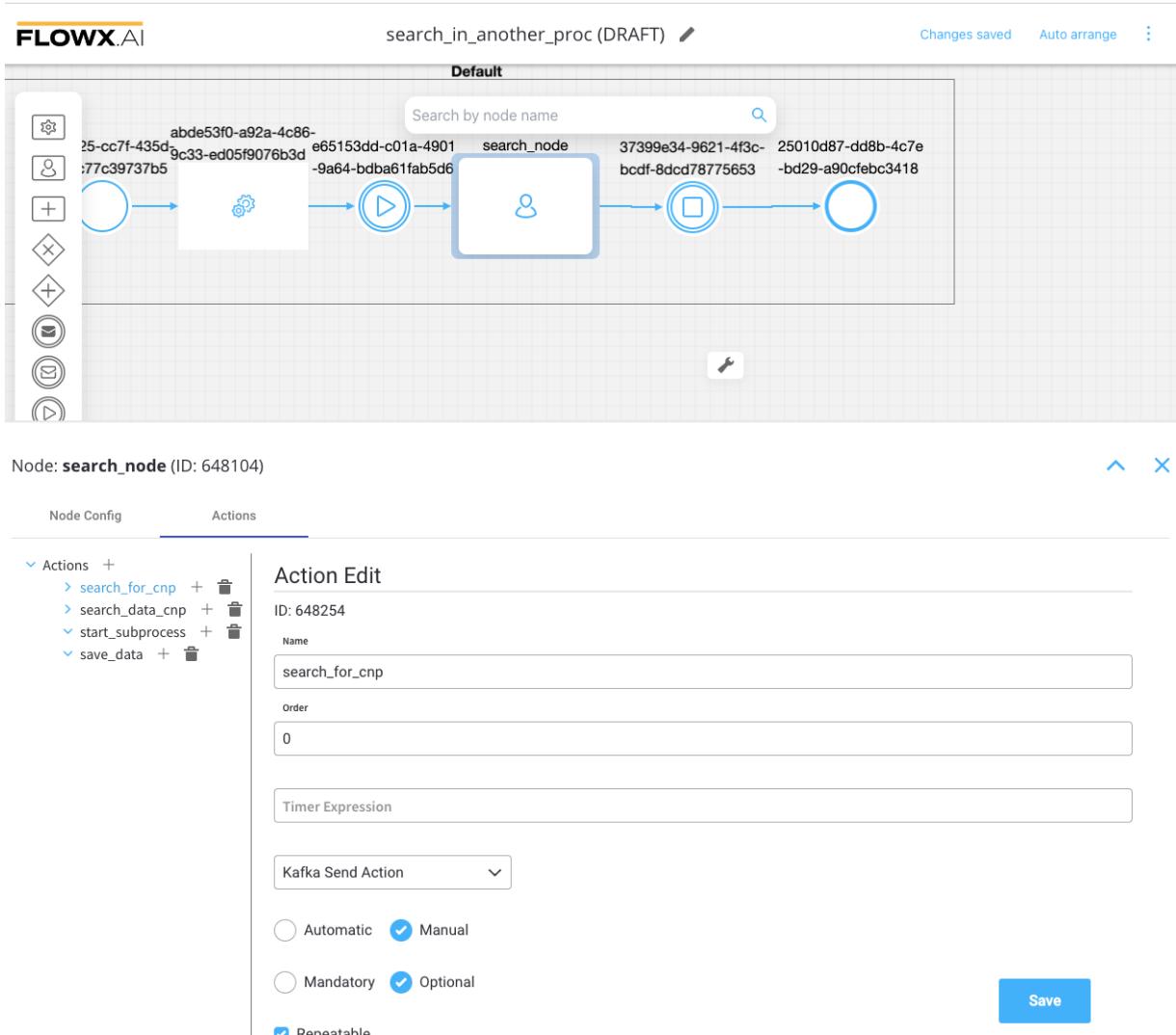
- search for data in other processes
- display results about other processes where the search key was found

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

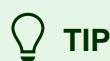
```
1  {
2    "searchKey": "application.client.identificationData.lastName",
3    "value": "${searchValue2}",
4    "processDefinitionNames": ["silviu_add_data_process"],
5    "processStartDateAfter": "2022-08-24T13:31:47.912524Z"
6 }
```

Advanced configurationShow Headers

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

Data to send **Add Key****Save**

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"
  processInstanceUuid: "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": "00000000000000000000000000000000"}, "identificationData": {"personalIdentificationNumber": "00000000000000000000000000000000"}, "identificationData": {"personalIdentificationNumber": "00000000000000000000000000000000"}}, "processEndDate": "2022-09-13T13:59:56.131Z", "status": "OK", "error": null}
    1: Object {"processInstanceUUID": "c8716ea6-0b3b-448d-a814-c84db6347ff7", "state": "FINISHED", "processStartDate": "2022-09-13T14:28:48.988Z", "data": {"application": {"client": "1871201460000", "identificationData": {"personalIdentificationNumber": "00000000000000000000000000000000"}, "identificationData": {"personalIdentificationNumber": "00000000000000000000000000000000"}, "identificationData": {"personalIdentificationNumber": "00000000000000000000000000000000"}}, "processEndDate": "2022-09-13T14:28:48.988Z", "status": "OK", "error": null}
    2: Object {"processInstanceUUID": "11f0503b-fd25-4e38-8744-42cef8838fbc", "state": "FINISHED", "processStartDate": "2022-09-13T14:44:59.061Z", "data": {"application": {"client": "1871201460000", "identificationData": {"personalIdentificationNumber": "00000000000000000000000000000000"}, "identificationData": {"personalIdentificationNumber": "00000000000000000000000000000000"}, "identificationData": {"personalIdentificationNumber": "00000000000000000000000000000000"}}, "processEndDate": "2022-09-13T14:44:59.061Z", "status": "OK", "error": null}
  }
  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-87d5-72eaab51928f"
webSocketAddress: "wss://public.qa.flowxai.dev/5929721e-1a5a-4ebd-87d5-72eaab51928f"
```

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

Parameters:

Name	Description	Type	Mandatory	Default value	
baseApiUrl	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; }
}
```

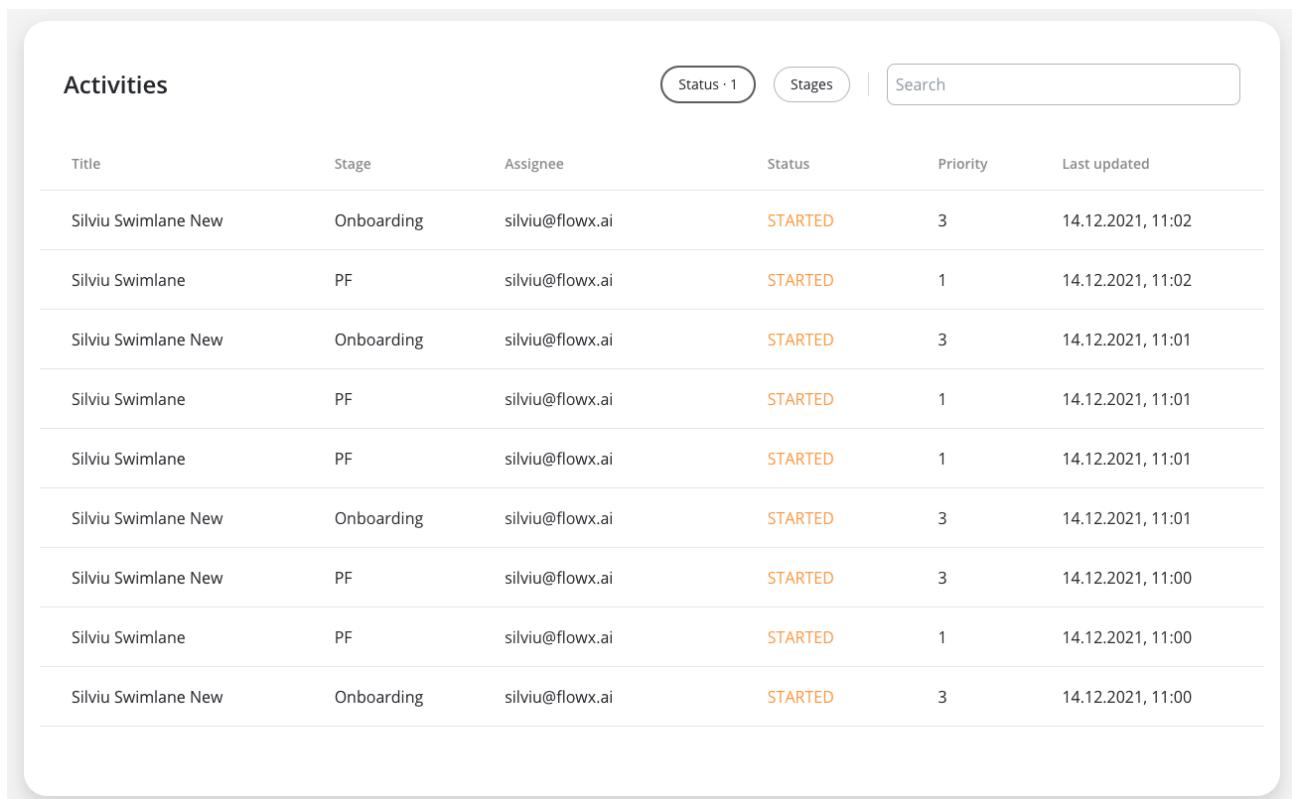
Interacting with the process

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 navigation bar with tabs for 'Status' (containing '1'), 'Stages', and a search bar. Below the navigation, there is a table titled 'Activities' with columns: Title, Stage, Assignee, Status, Priority, and Last updated. The table contains ten rows of data, each representing a task entry. The tasks are all labeled 'Silviu Swimlane' or 'Silviu Swimlane New' and are in the 'Onboarding' stage. The assignee for all tasks is 'silviu@flowx.ai'. The status for all tasks is 'STARTED'. The priority is set to 3 for most tasks and 1 for two specific ones. The last updated time for all tasks is '14.12.2021, 11:00'.

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 for managing process definitions. On the left, there is a sidebar with navigation links for Processes, Content Management, Plugins, and a user profile for John Doe. The main area is titled "Process Definitions" and contains 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

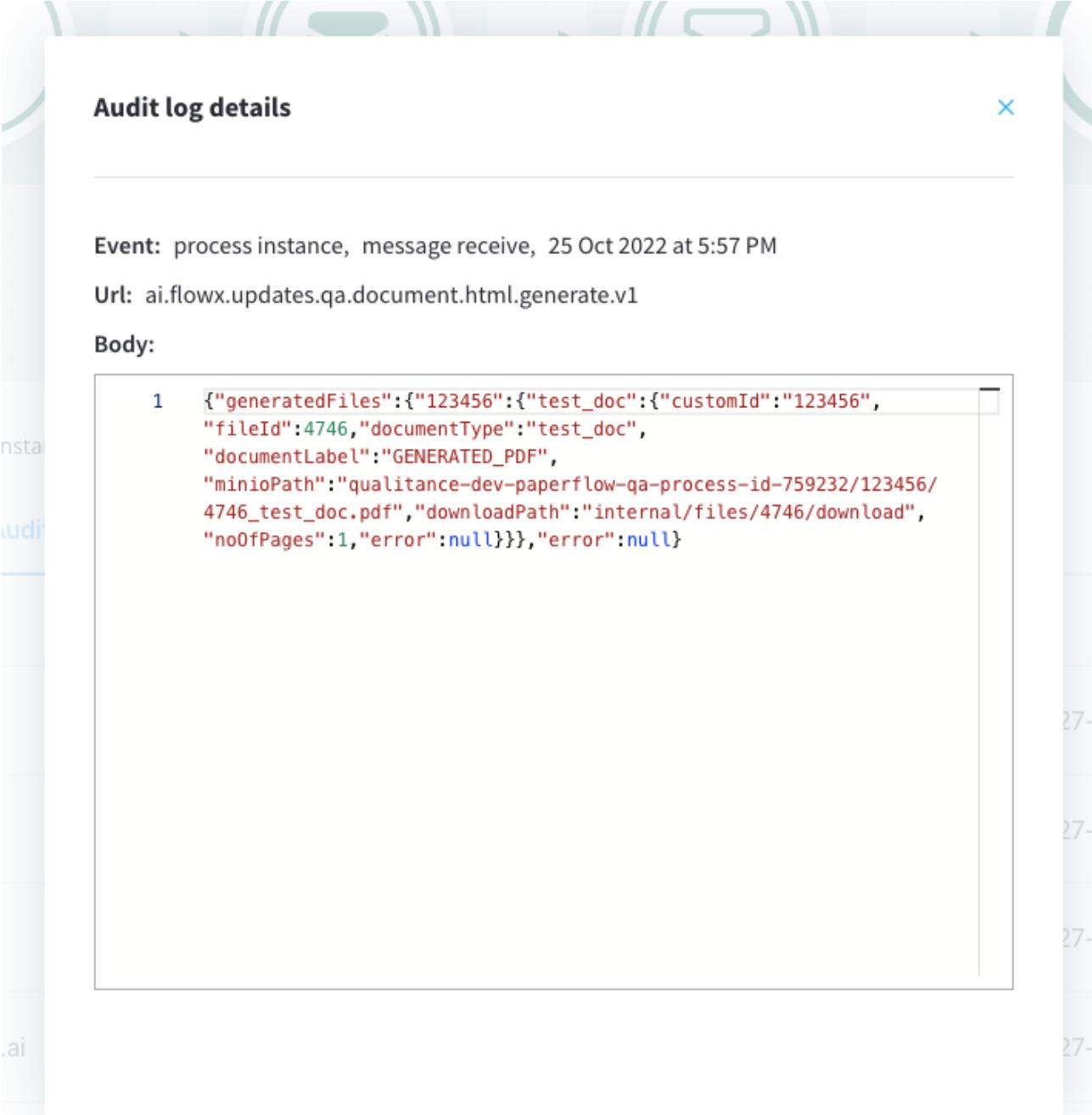
The engine listens for messages on topics with specific naming patterns. Make sure to use an outgoing topic name that matches the pattern configured in the engine.

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 Kafka message viewer titled "Audit log details". The message contains the following information:

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 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

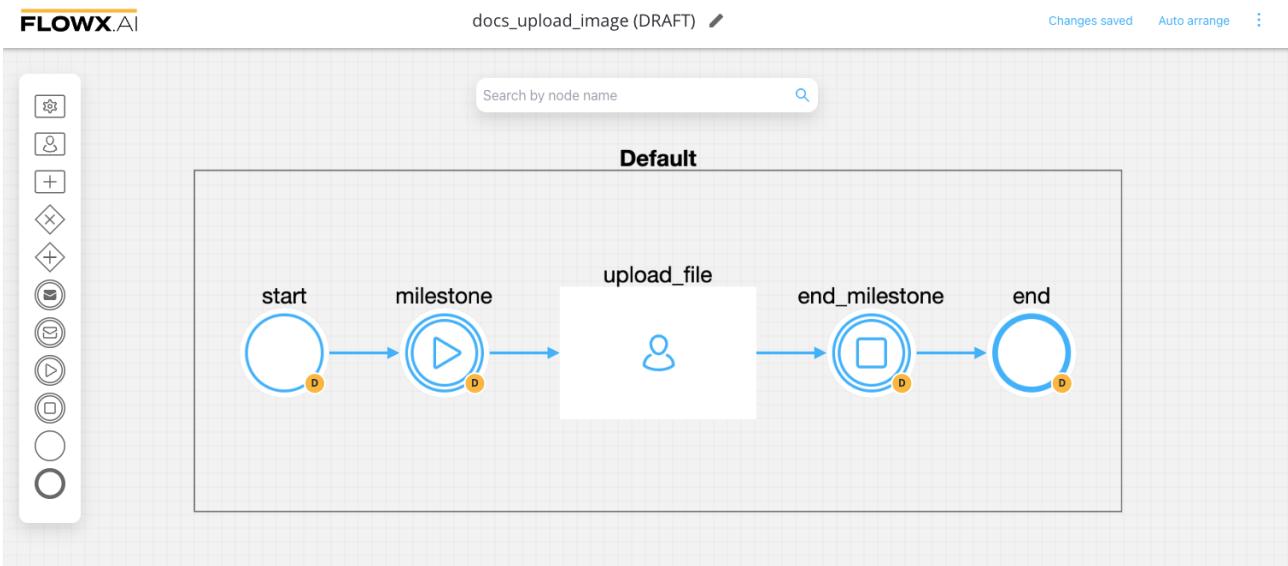
You can integrate document upload into a

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

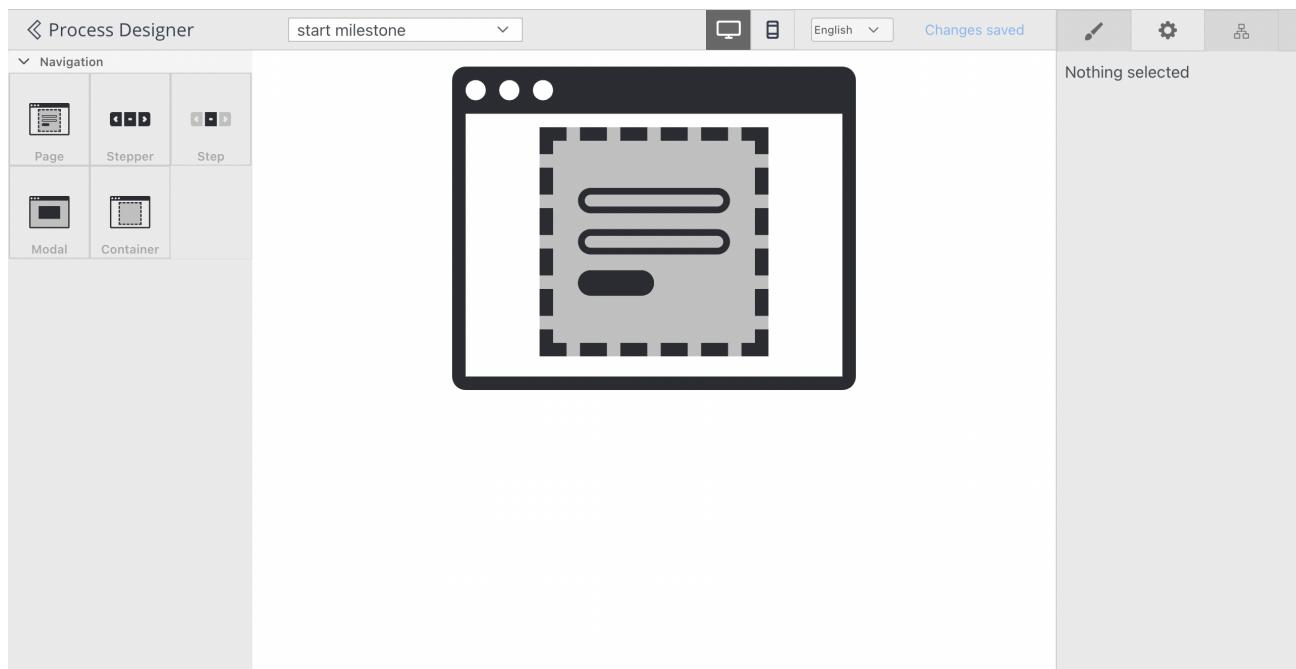
Show Headers

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 workflow editor interface. At the top, it displays the project name "pdf_to_jpeg (DRAFT)" and various status indicators. On the left, there is a sidebar with icons for creating new nodes and managing existing ones. The main workspace shows a state machine diagram titled "Default". The diagram consists of four states: "start", "convert_request", "convert_reply", and "end". Transitions between states are labeled "convert_request" and "convert_reply". The "convert_request" transition is highlighted with a yellow circle containing a double-headed arrow icon. Below the diagram, a search bar says "Search by node name".

Node: **convert_request** (ID: 725262)

Actions

Action Edit

ID: 725652

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

Order: 1

Timer Expression

Kafka Send Action

Automatic Manual

Mandatory Optional

Save

The "Actions" tab is selected in the node configuration dialog. The "Kafka Send Action" dropdown is open. Under "Automatic" and "Mandatory" options, there are checkboxes that are checked.

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-abv/-4e/a-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 process editor interface. At the top, it displays the project name "split_docs (DRAFT)" and various status indicators. On the left, there's a sidebar with icons for settings, users, and other process elements. The main workspace shows a workflow diagram titled "Default" with four states: "start", "split_request", "split_reply", and "end". Arrows connect "start" to "split_request", "split_request" to "split_reply", and "split_reply" to "end". The "split_request" state contains a yellow Kafka send event node icon. Below the workspace, a modal window is open for the "split_request" node, specifically for editing its "Actions". The modal has tabs for "Node Config" and "Actions", with "Actions" being the active tab. Under "Actions", there's a list of existing actions: "split_request" (selected). The "Action Edit" section shows the ID "728253" and fields for "Name" (set to "split_request") and "Order" (set to "1"). There's also a "Timer Expression" field which is currently empty. A dropdown menu shows "Kafka Send Action" is selected. Below the dropdown are two radio buttons: "Automatic" (checked) and "Manual". At the bottom of the modal are two checkboxes: "Mandatory" (checked) and "Optional". A blue "Save" button is located at the bottom right of the modal.

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

split_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.split.v1

Key Name

receiveReply



Add stream

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.

Audit log details

Event: process instance, message receive, 25 Oct 2022 at 2:36 PM

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

Body:

```
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

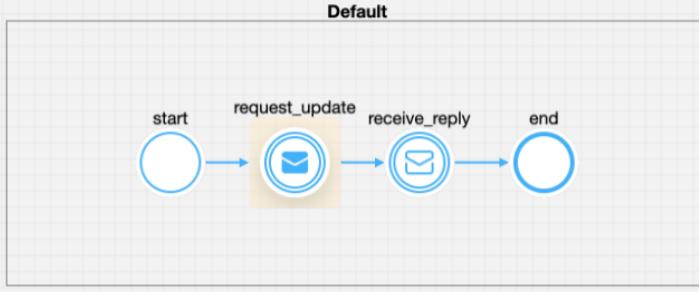
Sending the request

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

Default

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

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

Node Config	Actions
<p>Actions + + deleteDocument + </p>	<h3>Action Edit</h3> <p>ID: 765241</p> <p>Name <input type="text" value="deleteDocument"/></p> <p>Order <input type="text" value="1"/></p> <p>Timer Expression <input type="text"/></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}
```

instance

Audi

nx.ai

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

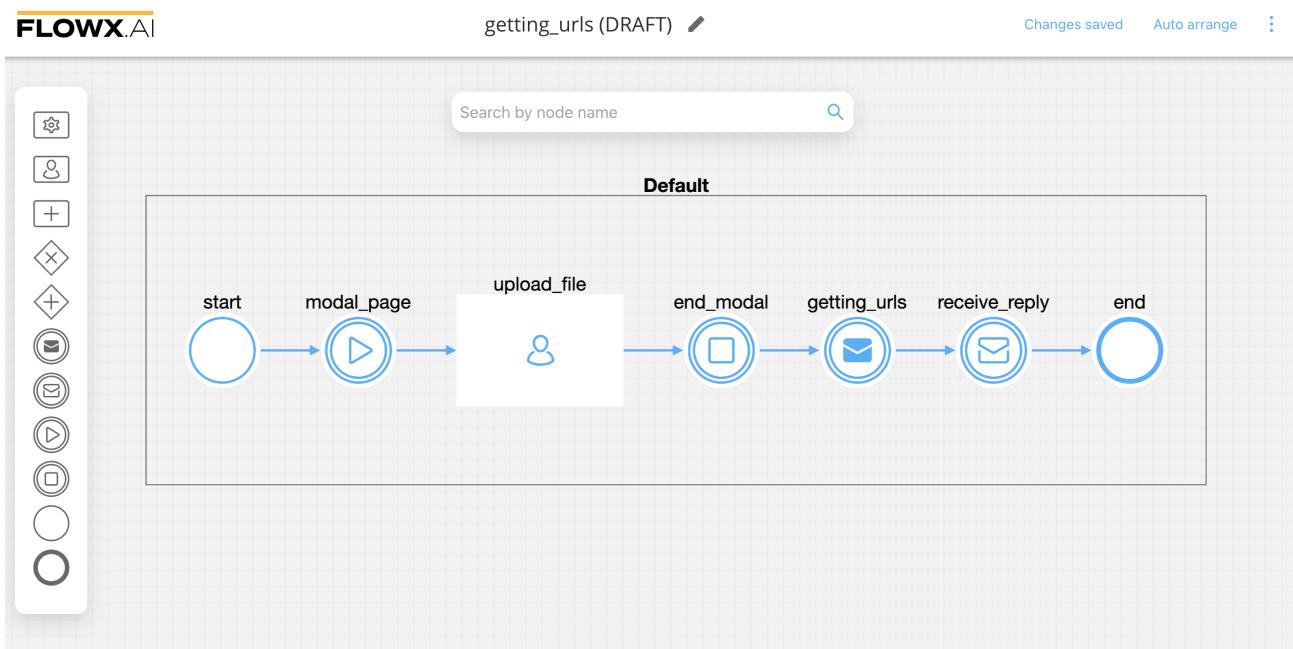
In certain scenarios, you may need to obtain URLs that point to uploaded documents to be used by other integrations. This requires adding a custom action to your process that requests the URLs from the Documents Plugin.

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 ▼

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.

The screenshot shows the FLOWX.AI application's sidebar navigation on the left and a main content area on the right. The sidebar includes sections for Media Library, Plugins (selected), Task Manager, All tasks, Hooks, Stages, Allocation rules, Out of office, Notification templates (selected), Document templates, General Settings (selected), Generic Parameters, Integration management, Licensing, Access management (selected), Users, Roles, Groups, and Audit Log. A user profile for 'John Doe' is at the bottom of the sidebar. The main content area has a header 'Notification Templates' with a search bar and a three-dot menu. It contains two sections: 'Drafts / In progress' and 'Published'. The 'Drafts / In progress' section lists three templates: 'Stunning Template' (version 2, edited 18 Oct 2022, 3:26 PM by John Doe), 'Amazing Template' (version 1, edited 18 Oct 2022, 3:21 PM by Jane Doe), and 'Exquisite Template' (version 1, edited 18 Oct 2022, 3:18 PM by Bess Twishes). The 'Published' section lists three templates: 'Cool Template' (version 21, published 20 Sep 2022, 4:30 PM by John Doe), 'Awesome Template' (version 1, published 03 Aug 2022, 11:48 AM by Jane Doe), and 'Creative Template' (version 1, published 08 Jun 2022, 11:11 AM by Bess Twishes). Each template row includes edit and more options icons.

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

Notifications Template - ExampleTemplate

Body Data model

Type: MAIL

Forward on Kafka

Language: Romanian (Romania)-ro-RO

Contract [\${firstInput}]

Source

Format toolbar

Flowx logo

Salut

Salut #firstInput #secondInput, ne bucurăm sa te avem alături!

Găsești toate detailele în brosura atașată acestui email.

Logout Test Save Publish

The screenshot shows the FLOWX.AI interface for creating a notification template. On the left is a sidebar with navigation links: Processes, Definitions, Active process, Content Management (with sub-links: Enumerations, Substitution tags, Content models, Languages, Source systems), Plugins (Task Manager, All tasks, Hooks, Stages), Notification templates (selected), Document templates, General Settings (Generic Parameters), Platform status, and Logout. The main area is titled 'Notifications Template - ExampleTemplate'. It has tabs for 'Body' (selected) and 'Data model'. Under 'Body', there are fields for 'Type' (MAIL), a checked 'Forward on Kafka' checkbox, 'Language' (Romanian (Romania)-ro-RO), and a 'Contract' placeholder with the value '[\${firstInput}]'. Below these are a 'Source' button and a rich-text editor toolbar. The main content area displays an email template with a header containing the Flowx logo, a greeting 'Salut', a body text 'Salut #firstInput #secondInput, ne bucurăm sa te avem alături!', and a footer with the message 'Găsești toate detailele în brosura atașată acestui email.' At the bottom are 'Test', '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 the FLOWX.AI interface for creating a document template named "ExampleTemplate". The main title "Documents Templates - ExampleTemplate" is at the top left, with a three-dot menu icon to its right. Below the title, there are two tabs: "Body" and "Data model", with "Data model" being the active tab. The "Data model" tab displays a configuration form for an attribute named "firstInput".
The attribute configuration includes:

- Name:** firstInput
- Item type:** STRING (selected from a dropdown menu)
- Mandatory:** A toggle switch is turned on.

At the bottom of the configuration form are two buttons: "Close" and "Save". To the right of the configuration form, a preview pane shows a table row with the header "New value" and a single data entry "firstInput". The preview also includes columns for "Mandatory" and "Actions".

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 FLOWX.AI Notifications Templates interface. On the left, there is a sidebar with the following navigation items:

- Enumerations
- Substitution tags
- Content models
- Languages
- Source systems
- Plugins
 - Task Manager
 - All tasks
 - Hooks
 - Stages
- Notification templates** (highlighted in blue)
- Document templates
- General Settings
 - Generic Parameters
 - Licensing
 - Access management
 - Users
 - Roles
- Groups

The main area is titled "Notifications Templates - Test_notification_template". It contains the following fields:

- Body** (selected tab) and **Data model**
- Type**: MAIL
- Forward on Kafka
- Language**: Romanian-ro
- Romanian-ro subject**
- Source** button
- WYSIWYG editor toolbar with various icons for text formatting and media.

» [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.

Notifications Templates - Test_notification_template

⋮

Body Data model

Type
MAIL

Forward on Kafka

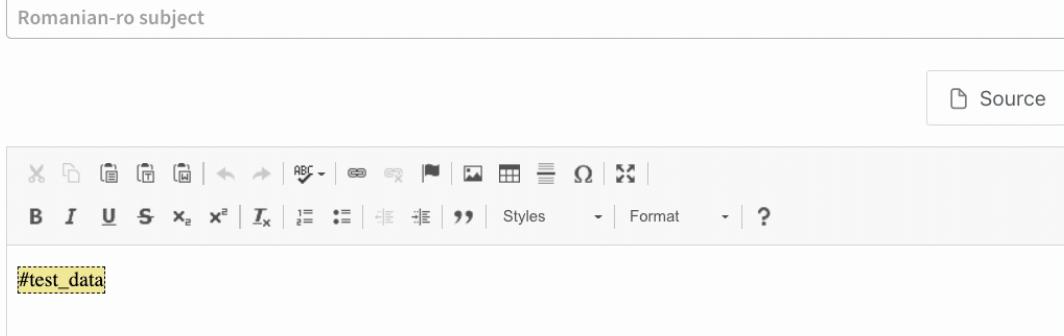
Language
Romanian-ro

Romanian-ro subject

Source

X **I** U **S** **x_e** **x^a** | **T_x** | **:=** **:=** | **;** **,** **;** **?** | Styles | Format | ?

#test_data



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 notifications in a table format. The columns are 'Edited at', 'Edited by', 'Published at', and 'Published by'. A context menu is open over the second notification (Edited by John Doe). The menu options are: Publish template, Export template, Show history, and Delete (highlighted in red).

Edited at	Edited by	Published at	Published by
18 Oct 2022, 3:26 PM	John Doe	20 Sep 2022, 4:30 PM	Jane Doe
18 Oct 2022, 3:21 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](#)

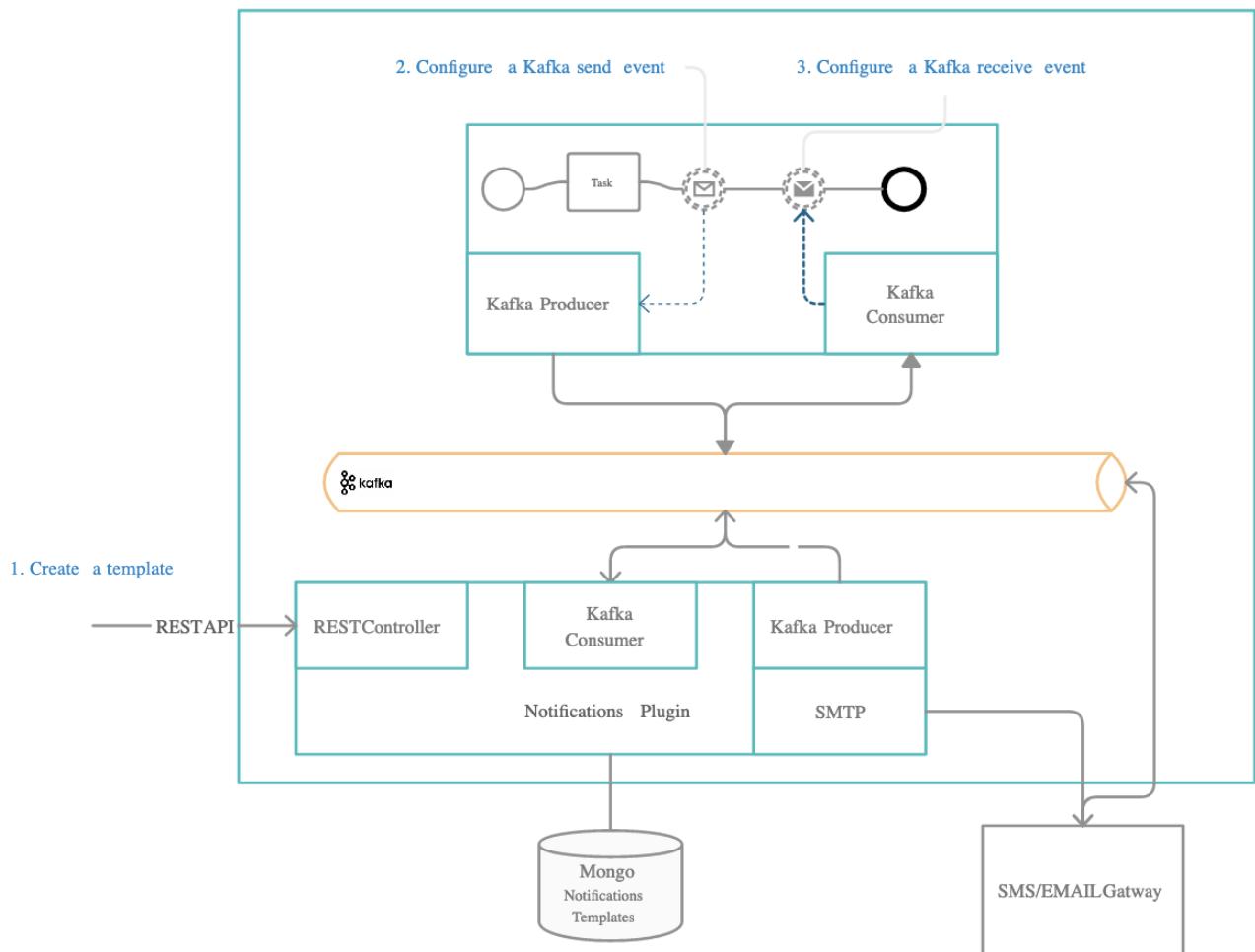
The following values are expected in the request body:

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.

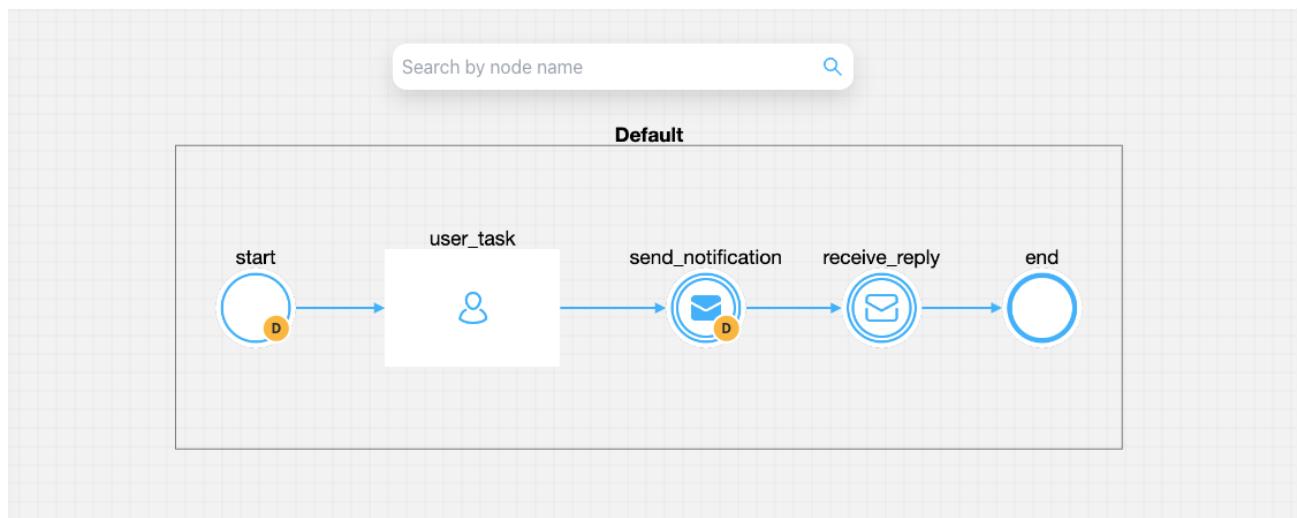


Define needed Kafka topics

Kafka topic names can be set by using environment variables:

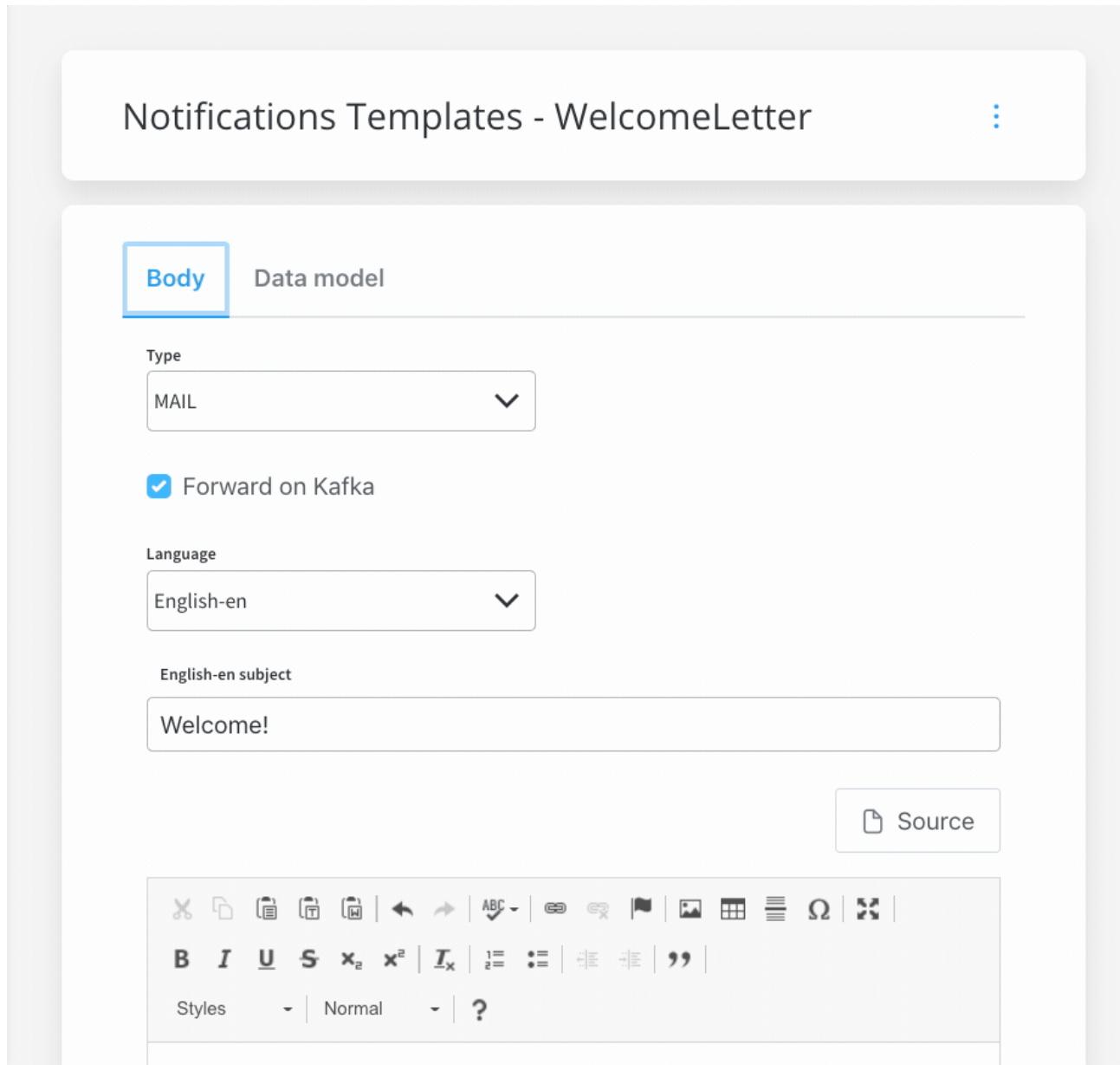
- **KAFKA_TOPIC_NOTIFICATION_INTERNAL_IN** - topic used to trigger the request to send a notification
- **KAFKA_TOPIC_NOTIFICATION_INTERNAL_OUT** - topic used for sending replies after sending the notification

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 X

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}
```

7b89870...
dit Log
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

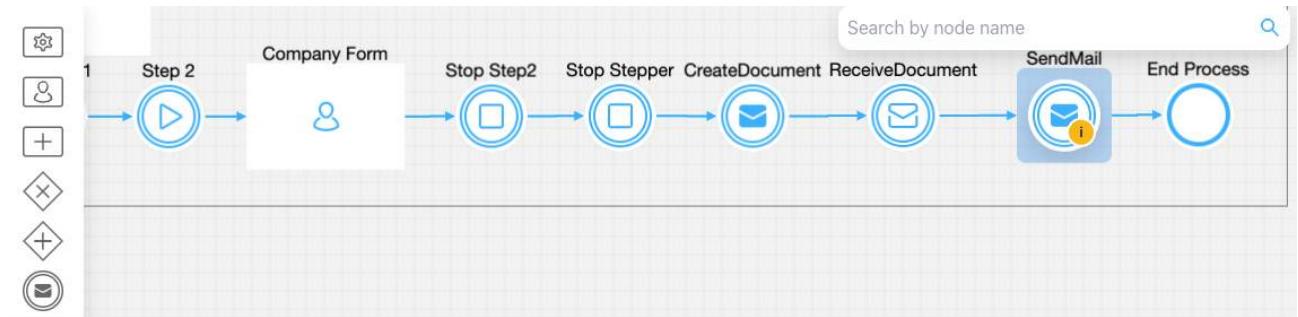
To use the notification plugin for sending emails with attachments, you must define the same topic configuration as for sending regular notifications. A notification template must be created, and the corresponding Kafka topics must be defined.

» [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 } 17 }</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!

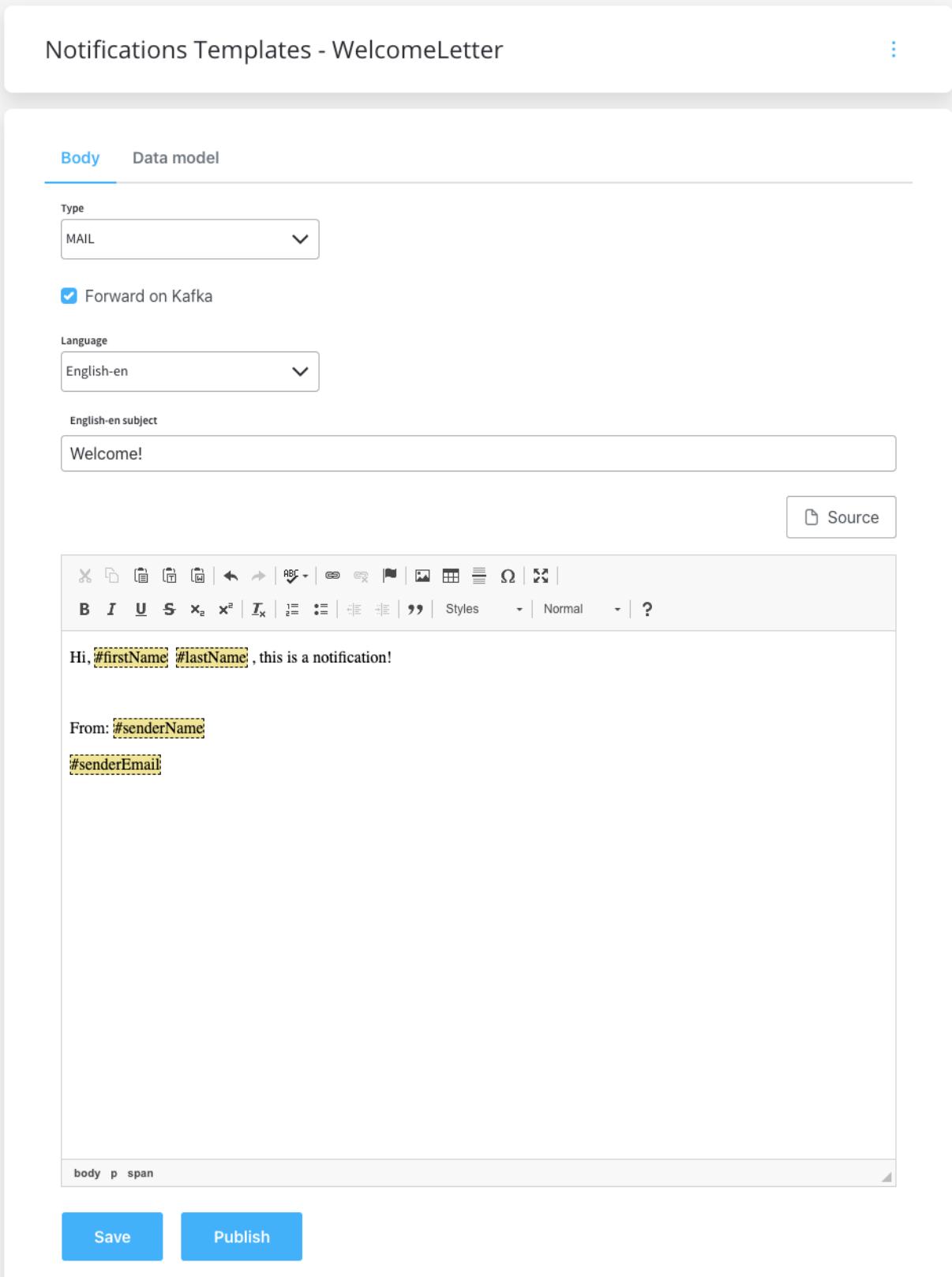
[Source](#)

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}}
```

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 log entries, with the first one being "7b89870...".

Response example at `KAFKA_TOPIC_NOTIFICATION_INTERNAL_OUT`:

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

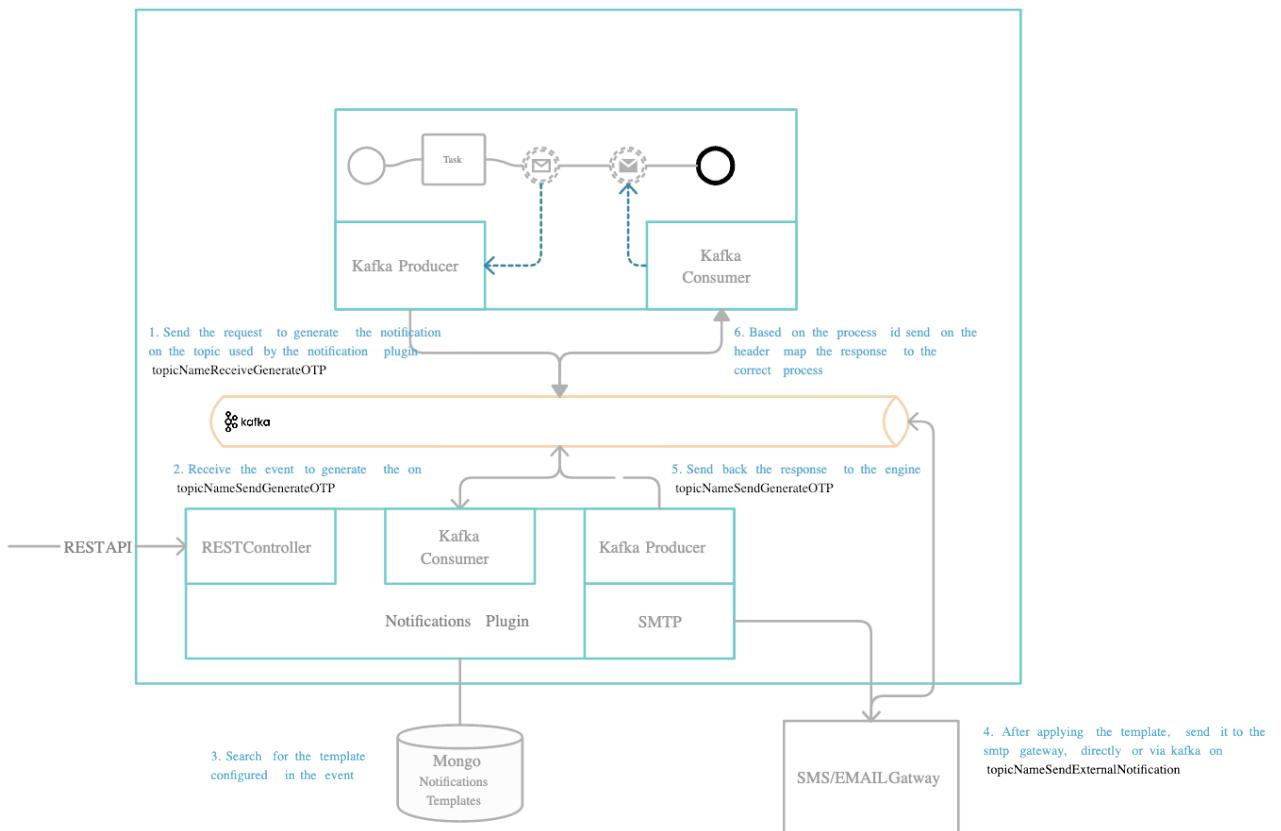
```
    "john.doe@mail.com"  
],  
"channel": "MAIL",  
"language": "en"  
}
```

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, there's a summary of an event: "Event: process instance, message receive, 18 Oct 2022 at 4:12 PM" and "Url: ai.flowx.updates.qa.notification.otp.generate.v1". Below this, under "Body:", a JSON object is displayed:

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

On the left side of the main interface, there's a vertical sidebar with tabs: "Active", "Audit" (which is selected), and "flowx.ai" (repeated twice). At the bottom of the screen, there are four status indicators: "flowx.ai", "Process Instance", "Execute action", and "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). The modal displays the following information:

Event: process instance, message receive, 18 Oct 2022 at 5:33 PM

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

Body:

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

The background of the application shows a list of audit logs with columns for Identifier, Version, and Date.

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.

The screenshot shows the 'Allocation rules' page. On the left, there is a sidebar with the following navigation items:

- Content models
- Languages
- Source systems
- Plugins** (selected)
- Task Manager
- All tasks
- Hooks
- Stages
- Allocation rules** (selected)
- Out of office
- Notification templates
- Document templates
- General Settings**
- Generic Parameters
- Integration management

On the right, the main area is titled 'Allocation rules'. It has a search bar 'Search by process name' and buttons 'Add process' and '⋮'. Below the title, it says 'Process' and lists two processes:

- Parallel Swimlanes Process**: Contains four swimlane allocations for 'OFFICEADMIN', 'SUPERVISOR', 'BACKOFFICE', and 'FRONTOFFICE'.
- Parallel Gateways Process**: Contains no allocations.

Each allocation row includes columns for 'Swimlane', 'Allocation', 'Edited at', and 'Edited by' (John Doe). There are edit and delete icons for each row.

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](#).

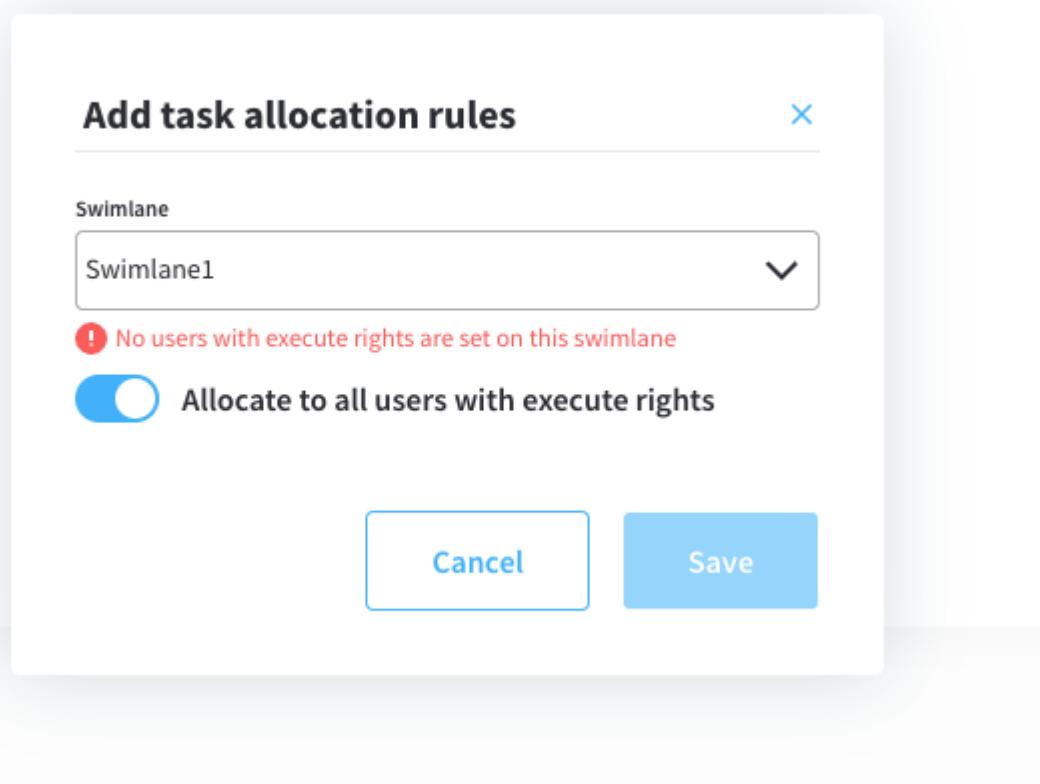
This screenshot shows the same 'Allocation rules' page as above, but with the 'Add process' button highlighted with a green border. The rest of the interface is identical to the first screenshot.

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.

Allocation rules

Search by process name

Process

doc

clie



silvi

silvi

Add task allocation rules



Swimlane

Swimlane1

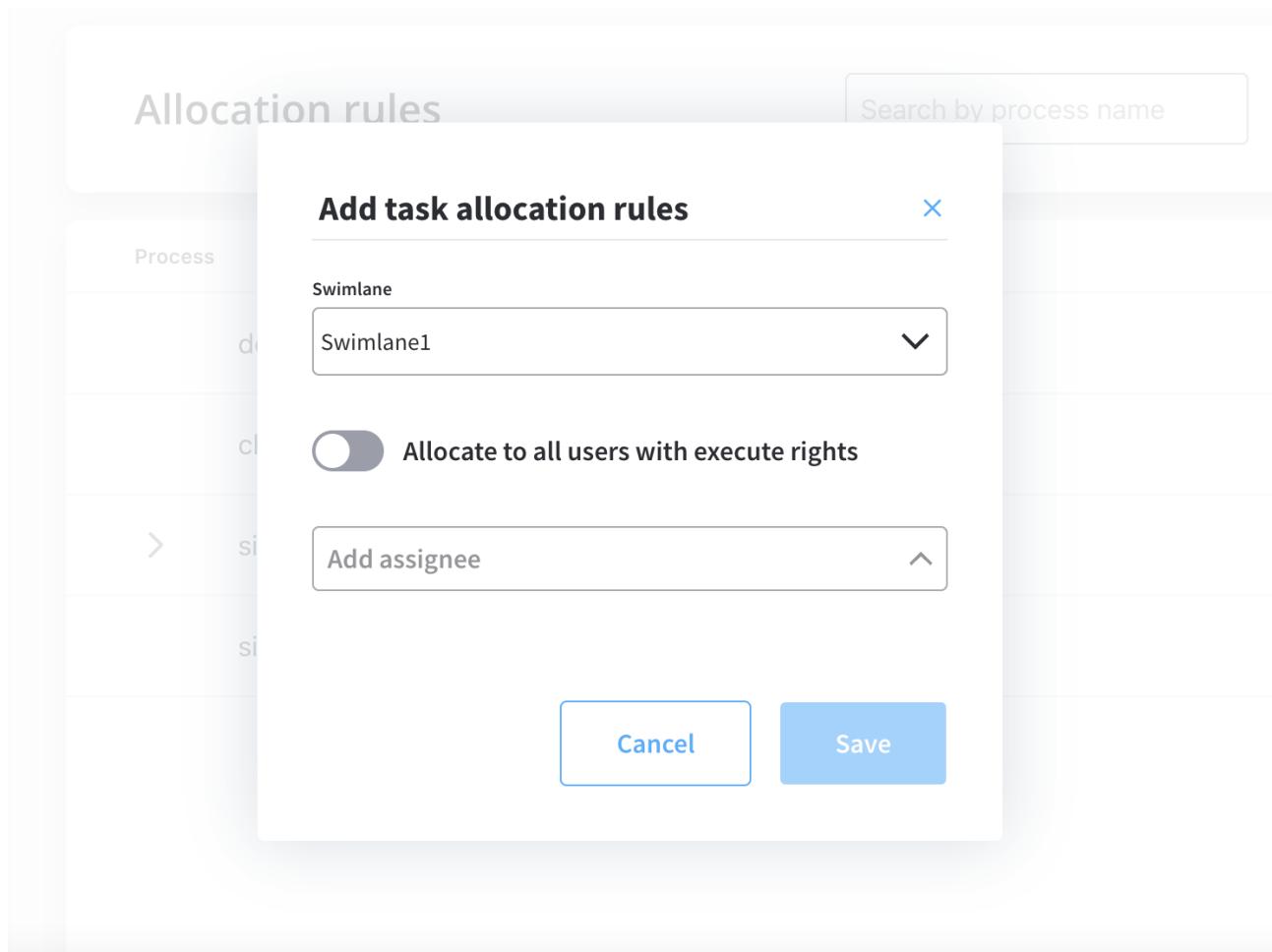


Allocate to all users with execute rights

Cancel

Save

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

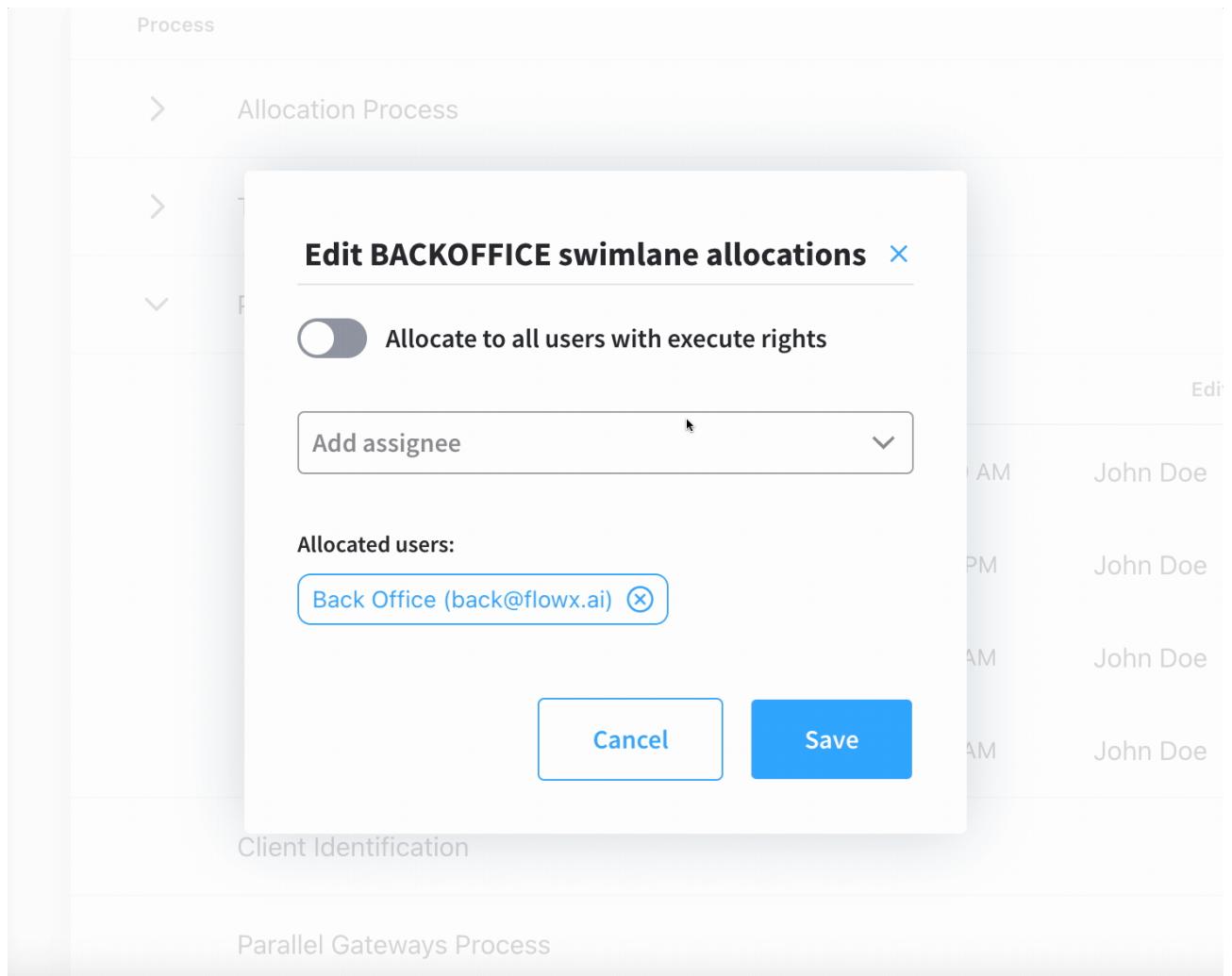
To edit allocation rules, follow the next steps:

1. Click **Edit** button.

The screenshot shows a user interface for managing allocation rules. At the top, there is a search bar labeled "Search by process name" and buttons for "Add process" and more options. Below this, a section titled "Allocation rules" lists three processes: "Allocation Process", "Test Process", and "Parallel Swimlanes Process". Each process has a plus sign and a more options menu icon. The "Parallel Swimlanes Process" is expanded, showing a table of allocations across four swimlanes: BACKOFFICE, OFFICEADMIN, SUPERVISOR, and FRONTOFFICE. The columns are "Swimlane", "Allocation", "Edited at", and "Edited by". Each row contains the swimlane name, the allocation date and time (e.g., 09 Aug 2022, 10:30 AM), the editor (John Doe), and edit/delete icons. The "Edit" icon for the OFFICEADMIN allocation is highlighted with a red box.

Swimlane	Allocation	Edited at	Edited by
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	

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 labeled "Search by process name", a "Add process" button, and a more options button. The table displays allocation rules for the "Swimlanes 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 application 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	Status	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.

The screenshot shows a 'Hooks' section on the left with three items: 'Hook Name', 'Hook1', 'Hook2', and 'Hook3'. A central modal window titled 'Add new hook' is open, containing fields for 'Name', 'Parent process', 'Type', 'Trigger', 'Triggered Process', and an 'Active' checkbox. The '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.

The screenshot shows a user interface for managing out-of-office records. At the top, there is a search bar labeled "Search by user" and a blue "Add out-of-office" button. Below this is a table with columns: User, Start Date, End Date, Edited at, and Edited by. The table contains three rows of data:

User	Start Date	End Date	Edited at	Edited by
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 application interface. On the left, there is a sidebar with a tree-like navigation menu. The 'Task Management' section is expanded, showing 'All tasks', 'Hooks', 'Stages', 'Allocation rules', and 'Out of office' (which is currently selected and highlighted in blue). Other sections like 'Notification templates' and 'Document templates' are also listed. Below 'Task Management' is another section 'General Settings' with 'Generic Parameters', 'Integration management', 'Licensing', and 'Access management'. At the bottom of the sidebar, there is a user profile for 'John Doe' and a three-dot menu icon. The main content area is titled 'Out of office' and contains a table with three rows of data. The columns are 'User', 'Start Date', 'End Date', 'Edited at', and 'Edited by'. The first row shows 'Jane Smith (jane....)' with dates '03 Aug 2022' and '07 Aug 2022', edited at '03 Aug 2022, 3:24 PM' by 'John Doe'. The second row shows 'Bess Twishes (be...' with dates '03 Aug 2022' and '31 Dec 2046', edited at '03 Aug 2022, 3:16 PM' by 'John Doe'. The third row shows 'Gene Eva (gene....)' with dates '08 Aug 2022' and '27 Aug 2022', edited at '03 Aug 2022, 9:25 AM' by 'John Doe'. Each row has edit and delete icons at the end. A search bar labeled 'Search by user' is located above the table, and a 'Add out-of-office' button is on the right side of the table header.

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)

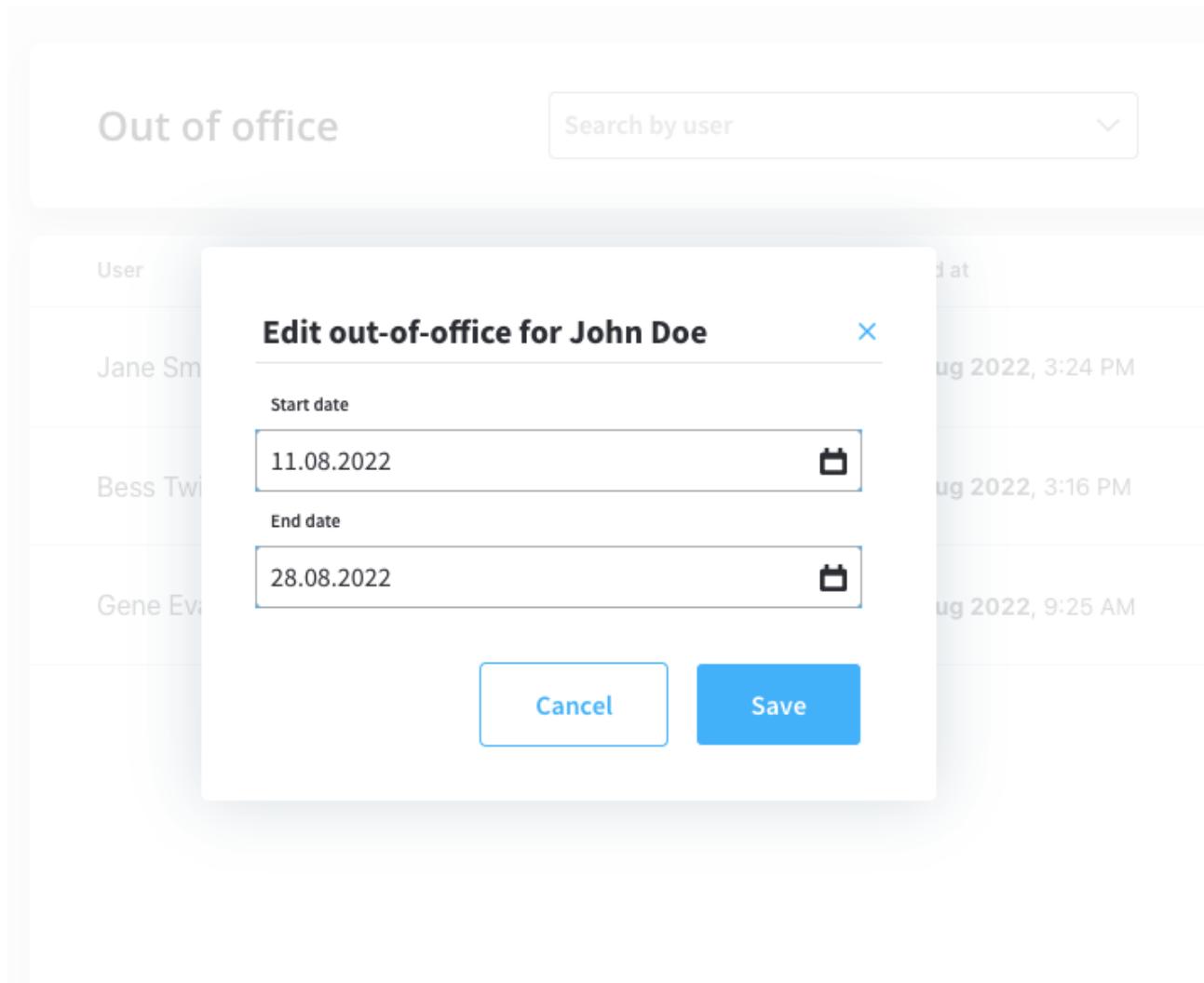
The screenshot shows the Flowx AI application interface. At the top, there's a header with the Flowx logo and a search bar labeled "Search by user". Below the header, a sidebar lists users: Jane Smith, Bess Twink, and Gene Evans. The main area displays activity logs for each user, with entries like "Aug 2022, 3:24 PM", "Aug 2022, 3:16 PM", and "Aug 2022, 9:25 AM". A modal window titled "Add out-of-office" is centered over the list. It contains three input fields: "Add assignee" (with a dropdown arrow), "Start date" (with a calendar icon), and "End date" (with a calendar icon). At the bottom of the modal are two buttons: "Cancel" (in a white box) and "Save" (in a blue box).

3. Click **Save**.

Editing out-of-office records

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

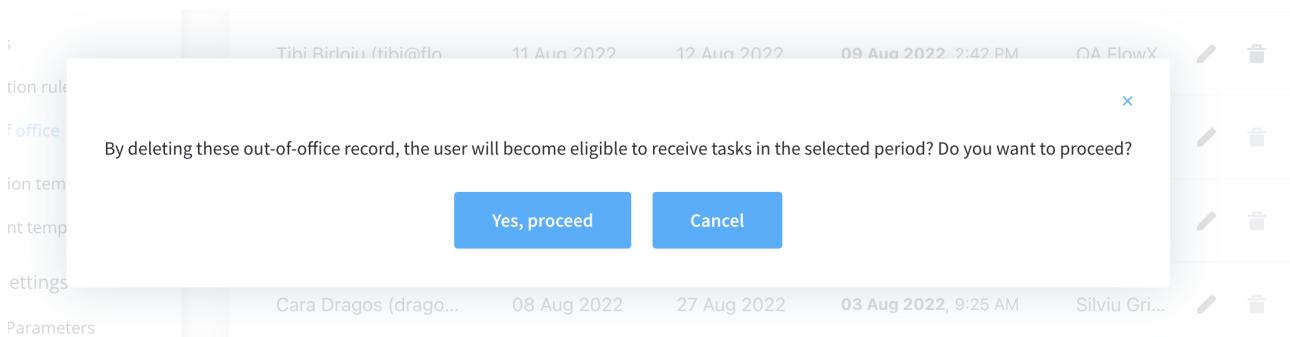
1. Click **Edit** button.
2. Modify the dates (! cannot be earlier than tomorrow).
3. Click **Save**.



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

Out of office					Search by user	Add out-of-office
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 modal dialog box titled "Add new stage" with a close button (X) in the top right corner. Inside the dialog, there is a text input field labeled "Name". Below the input field, a tooltip message reads: "i Name shouldn't contain any special characters or white spaces." At the bottom of the dialog is a blue "Add" button. In the background, a vertical sidebar on the left lists several stages: "Name", "PF", "Offboa", "Onboa", "Identifi", "testirin", and "Silviu".

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)

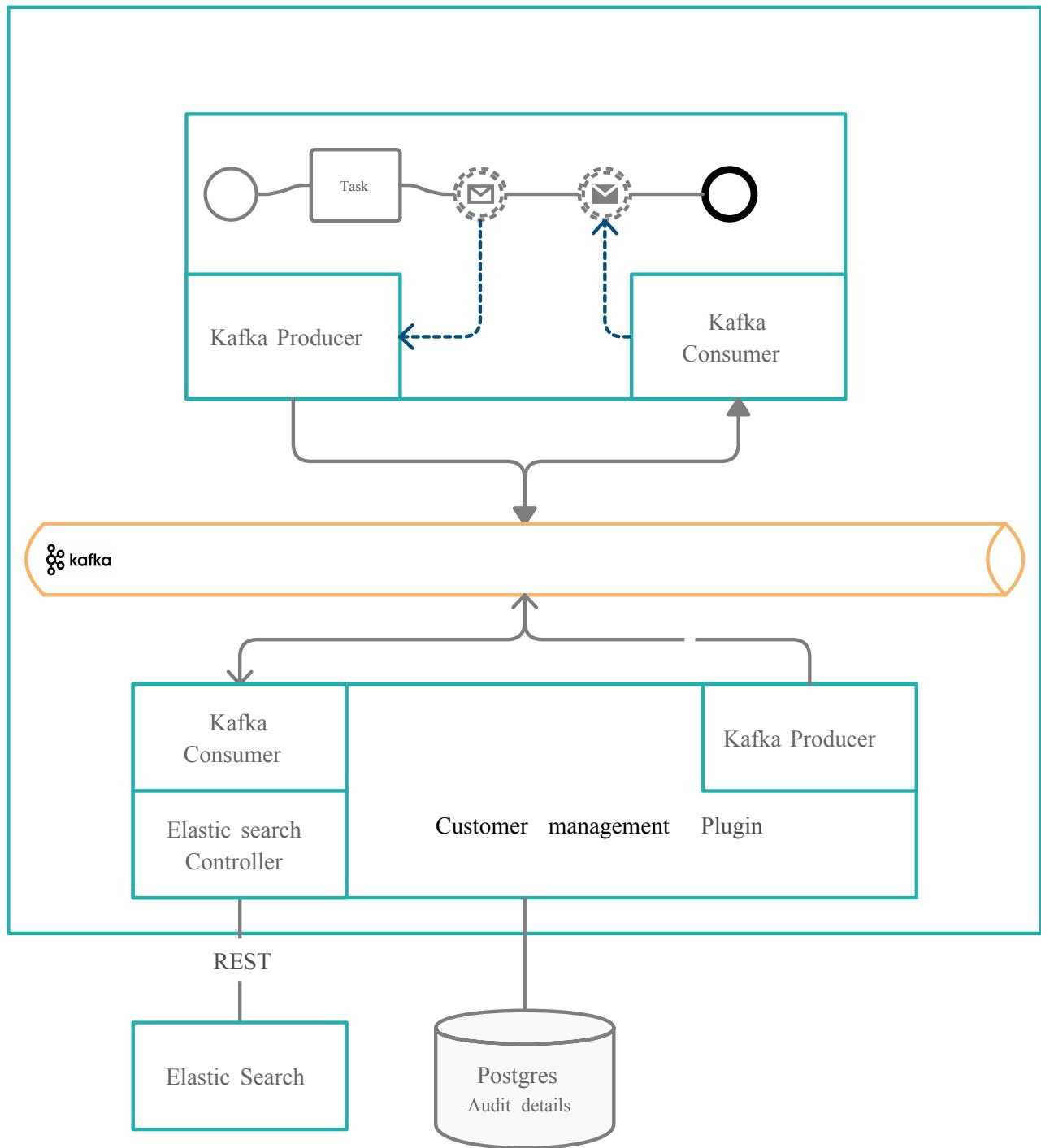
Data stream topics

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 navigation links: 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), and a user profile section (DC John Doe, three dots).

The main area is titled "Process Definitions". It has a search bar and a more button. Below it, 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	▶ ⚒ ⋮

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' interface for a Kafka Send Action. The left sidebar lists actions under 'Actions'. The main area shows the configuration for action ID 584701. It includes fields for Name ('send_main_applicant_consent'), Order (set to 1), Timer Expression (empty), and a dropdown for Action Type ('Kafka Send Action'). Below these are checkboxes for 'Automatic' (checked), 'Manual', 'Mandatory' (checked), 'Optional', and 'Repeatable'. A toggle switch for 'Autorun Children?' is also present. The 'Parameters' section has tabs for 'Custom' (selected) and 'From integration'. Under 'Topics', the value 'ai.flowx.in.sandbox.document.html.generate.v1' is listed.

(!) 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

! INFO

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/elastic-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

```
cramdb:
  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_BYT`ES - 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

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

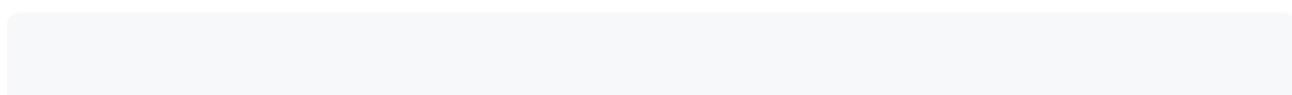
Environment Variable	Definition	Default Value	
STORAGE_S3_SECRET_KEY	The secret key for connecting to the S3 storage service	-	

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's document template editor. On the left, there is a sidebar with the following navigation:

- 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
- Platform status

The main area is titled "Documents Templates - ExampleTemplate". It has tabs for "Body" and "Data model". The "Body" tab is selected, showing a rich text editor toolbar and a preview area. The preview area contains placeholder text "#firstInput" and "#secondInput", followed by a large block of Lorem ipsum text. At the bottom of the preview area, there is a code editor snippet showing the corresponding HTML and CSS. Below the preview area are "Test", "Save", and "Publish" buttons. The bottom left corner of the sidebar has a "Logout" link.

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, there is a sidebar with the following navigation items:

- 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
- Platform status

On the right, 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 and two input fields labeled "#firstInput" and "#secondInput". The preview area also contains some sample text: "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sollicitudin leo in sem tempus, ut aliquet eros porttitor. Nullam condimentum dapibus massa in maximus. Vivamus nulla est, posuere eget metus quis, iaculis euismod magna. Integer non nisi non ipsum porttitor pulvinar. Fusce commodo varius felis at finibus. Vestibulum pulvinar mauris sit amet nisl vestibulum, vitae facilisis sem convallis. In hac habitasse platea dictumst. Nulla sagittis nibh mi, venenatis body p". At the bottom of the preview area are "Test", "Save", and "Publish" buttons.

» 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

Notifications Template - ExampleTemplate

Body Data model

Type: MAIL

Forward on Kafka

Language: Romanian (Romania)-ro-RO

Romanian (Romania)-ro-RO subject: Contract [\${firstInput}]

[Source](#)

WYSIWYG Editor:

Content:

Salut #firstInput #secondInput, ne bucurăm sa te avem alături!

Găsești toate detaliele în brosura atașata acestui email.

Buttons: Test, Save, Publish

» 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

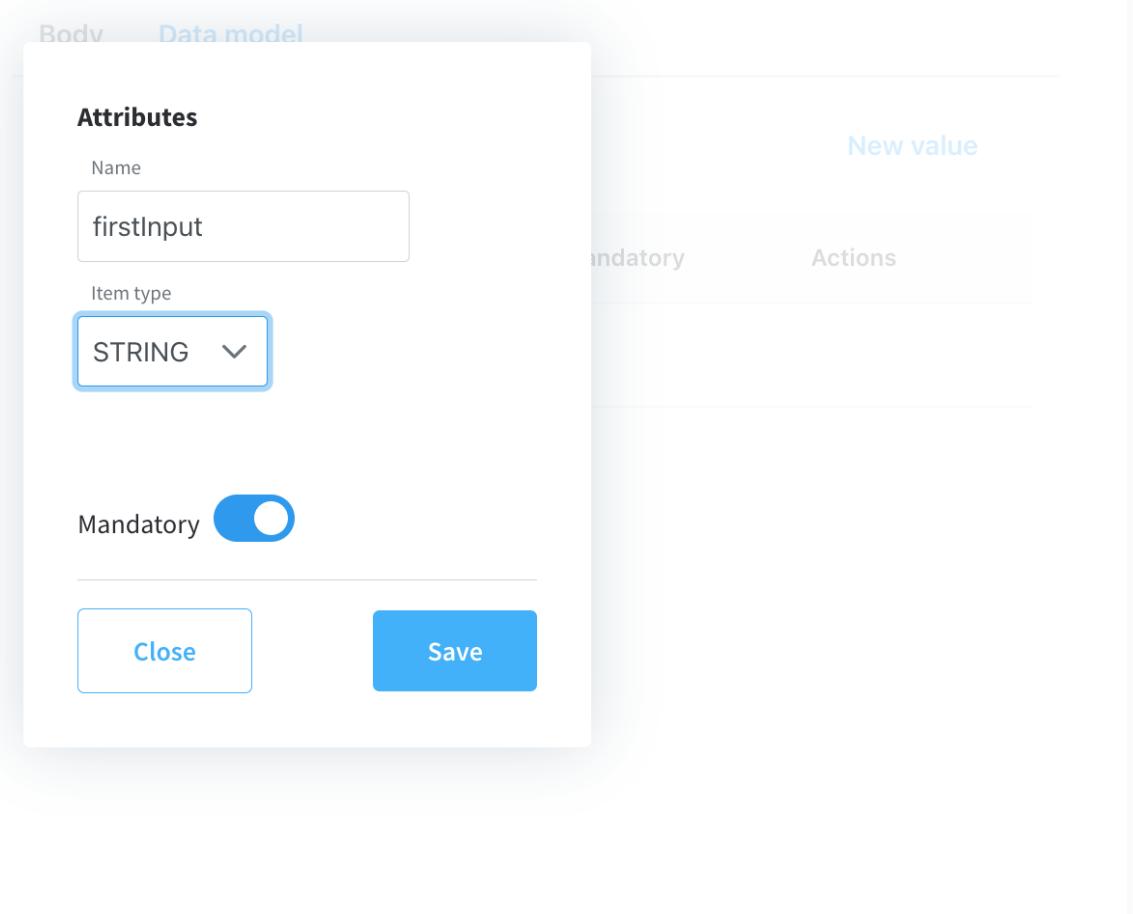
Attributes

Name	firstInput
Item type	STRING
Mandatory	<input checked="" type="checkbox"/>

New value

Actions

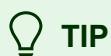
Close **Save**

**!** 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

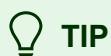
Here's an example of a Kafka listener method:

```
@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](#)

PostgreSQL

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

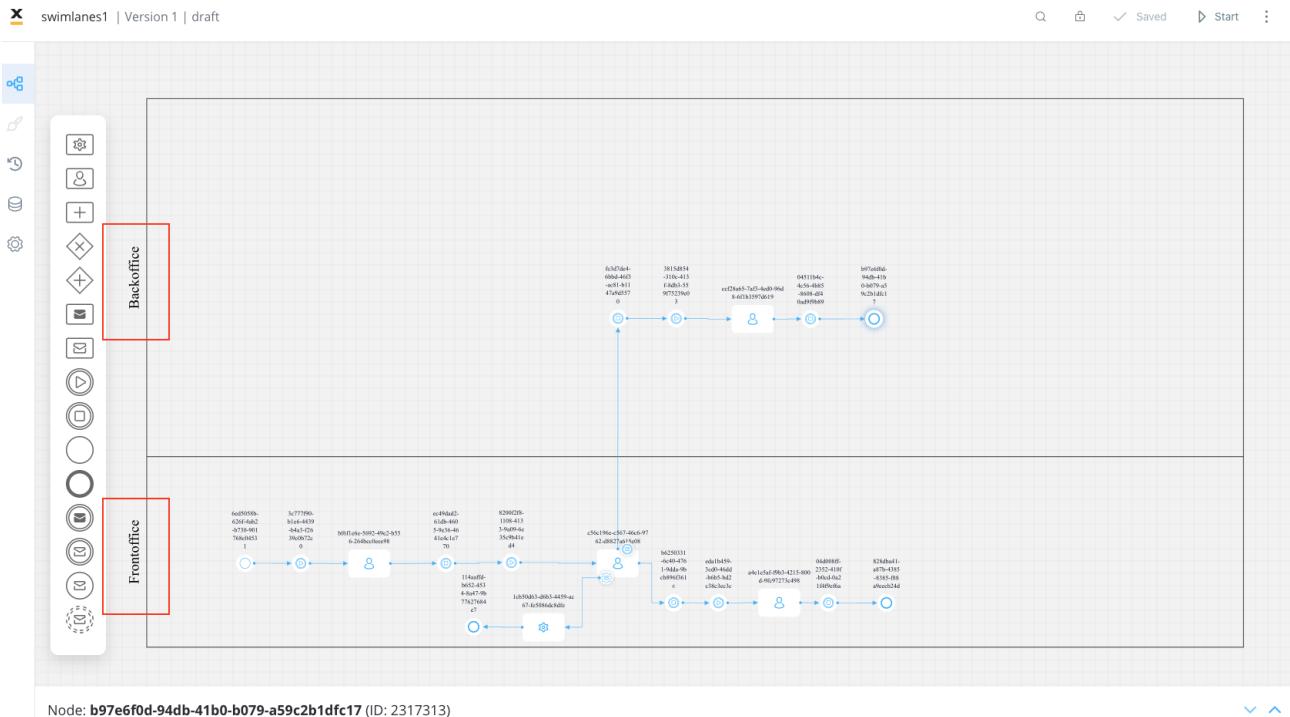
What is it? Swimlanes provide a way of grouping process nodes by process participants.

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

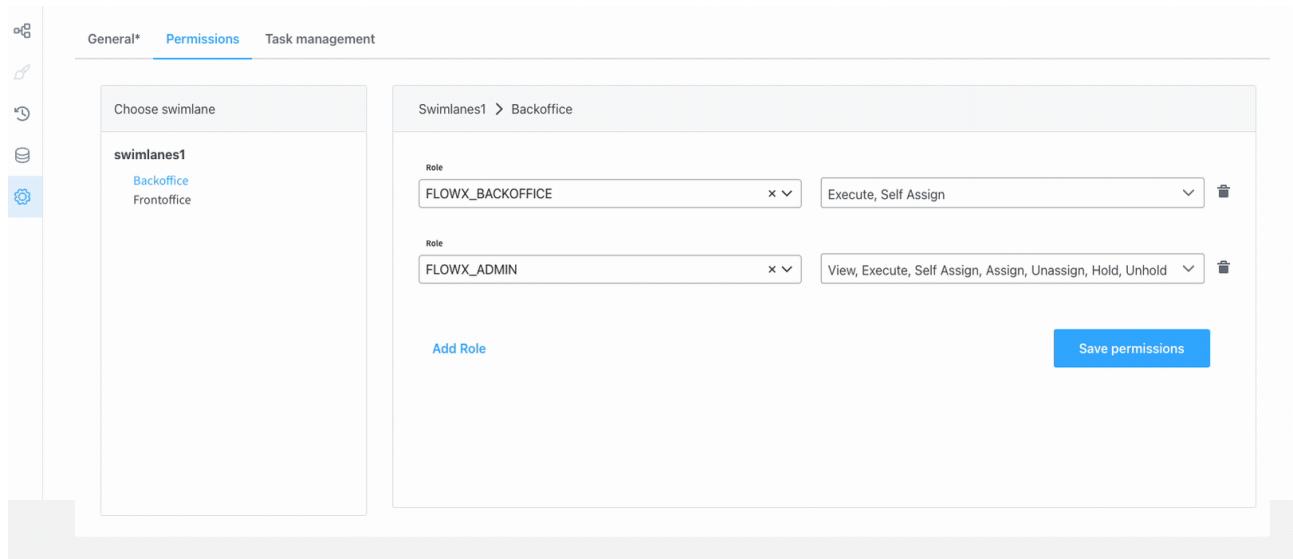
The fallback content to display on prerendering based on user roles. This can be achieved by organizing nodes into different swimlanes.

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?](#)

PLATFORM SETUP GUIDES / Overview

The setup guides in this section will provide information on how to install, configure, and use FLOWX.AI services.

Deploying microservices typically involves breaking down the application into smaller, modular components. Each microservice should be independently deployable, with all the necessary dependencies and configurations included.

Once the microservices have been created, they can be deployed using a container management system such as Docker or Kubernetes. These systems allow for the deployment of multiple microservices in a single environment.

Environment variables

Environment variables are variables that are set in the system environment and can be used by applications and services to store and access configuration information. Environment variables typically include settings such as paths to directories, file locations, settings for the operating system and applications, and more.

Environment variables are used to store and access configuration information in a secure and efficient manner. Below you will find some examples of common/shared environment variables that need to be set for different services and components.

Authorization & access roles

An identity management platform is a software system that helps you manage authorization & access roles, including user accounts, passwords, access control, and authentication. Identity management platforms typically offer features such as user provisioning, identity federation, and single sign-on.

The following variables need to be set in order to connect to the identity management platform:

- `SECURITY_OAUTH2_BASE_SERVER_URL` - the base URL for the OAuth 2.0 Authorization Server, which is responsible for authentication and authorization for clients and users, it is used to authorize clients, as well as to issue and validate access tokens
- `SECURITY_OAUTH2_CLIENT_CLIENT_ID` - a unique identifier for a client application that is registered with the OAuth 2.0 Authorization Server, this is used to authenticate the client application when it attempts to access resources on behalf of a user
- `SECURITY_OAUTH2_CLIENT_CLIENT_SECRET` - secret key that is used to authenticate requests made by an authorization client
- `SECURITY_OAUTH2_REALM` - security configuration env var in the Spring Security OAuth2 framework, it is used to specify the realm name used when authenticating with OAuth2 providers

» [Access Management](#)

Datasource configuration

Datasource configuration is the process of configuring a data source, such as a database, file, or web service, so that an application can connect to it and use the data. This typically involves setting up the connection parameters, such as the host, port, username, and password.

In some cases, additional configuration settings may be required, such as specifying the type of data source (e.g. Oracle, MySQL, etc.) or setting up access control for data access.

Environment variables are more secure than hard-coding credentials in the application code and make it easier to update data source parameters without having to modify the application code.

CAUTION

Some microservices (**Admin** microservice, for example, connects to the same Postgres / Oracle database as the **Engine**).

INFO

Depending on the data source type, various parameters may need to be configured. For example, if connecting to an Oracle database, the driver class name, and the database schema must be provided. For MongoDB, the URI is needed.

The following variables need to be set in order to set the datasource:

- `SPRING_DATASOURCE_URL` - environment variable used to configure a data source URL for a Spring application, it typically contains the JDBC driver name, the server name, port number, and database name
- `SPRING_DATASOURCE_USERNAME` - environment variable used to set the username for the database connection, this can be used to connect to a database instance

- `SPRING_DATASOURCE_PASSWORD` - environment variable used to store the password for the database connection, this can be used to secure access to the database and ensure that only authorized users have access to the data
- `SPRING_DATASOURCE_DRIVERCLASSNAME` (! only for Oracle DBs) - environment variable used to set the class name of the JDBC driver that the Spring DataSource will use to connect to the database
- `SPRING_JPA_PROPERTIES_HIBERNATE_DEFAULTSCHEMA` (! only for Oracle DBs) - environment variable used to overwrite the name of the database schema
- `SPRING_DATA_MONGODB_URI` (! only for MongoDB) - environment variable used to provide the connection string for a MongoDB database that is used with, this connection string provides the host, port, database name, user credentials, and other configuration details for the MongoDB server

⚠ CAUTION

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.

❗ INFO

The datasource is configured automatically via a liquibase script inside the engine. All updates will include migration scripts.

Kafka

The following Kafka-related configurations can be set by using environment variables:

- `SPRING_KAFKA_BOOTSTRAP_SERVERS` - environment variable used to configure the list of brokers to which the kafka client will connect, this is a comma-separated list of host and port pairs that are the addresses of the Apache Kafka brokers in a Kafka cluster
- `SPRING_KAFKA_CONSUMER_GROUP_ID` - environment variable is used to set the consumer group ID for the Kafka consumer, it is used to identify which consumer group the consumer belongs to and allows the Kafka broker to manage which messages are consumed by each consumer in the group

 **INFO**

`SPRING_KAFKA_CONSUMER_GROUP_ID` - might be different for the services that have the group id separated in topics, also thread numbers.

- `KAFKA_CONSUMER_THREADS` - environment variable used to control the number of threads that a Kafka consumer instance can use to consume messages from a cluster, it defines the number of threads that the consumer instance should use to poll for messages from the Kafka cluster
- `KAFKA_AUTH_EXCEPTION_RETRY_INTERVAL` - environment variable used to set the interval at which Kafka clients should retry authentication exceptions (the interval between retries after AuthorizationException is thrown by KafkaConsumer)
- `KAFKA_MESSAGE_MAX_BYT`ES - 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

FLOWX.AI Engine is listening for messages on topics with names of a certain pattern, make sure to use correct outgoing topic names when configuring the services.

Redis configuration

Redis configuration involves setting up the connection parameters, such as the host, port, username, and password. In some cases, additional configuration settings may be required, such as specifying the type of data store or setting up access control for data access.

- `SPRING_REDIS_HOST` - environment variable used to configure the hostname or IP address of a Redis server when using Spring Data Redis
- `SPRING_REDIS_PASSWORD` - environment variable is used to store the password used to authenticate with a Redis server, it is used to secure access to the Redis server and should be kept confidential
- `REDIS_TTL` - environment variable is used to specify the maximum time-to-live (TTL) for a key in Redis, it is used to set a limit on how long a key can exist before it is automatically expired (Redis will delete the key after the specified TTL has expired)

Debugging

Advanced debugging features can be enabled. When this happens, snapshots of the process status will be taken after each action and can be later used for debugging purposes. This feature comes with an exponential increase in database usage, so we suggest having the flag set to true on debugging media and false production ones.

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` - controls the verbosity of the application's logs and how much information is recorded (app level logs)

Tracing via Jaeger

Tracing via Jaeger involves collecting timing data from the components in a distributed application. This allows you to better identify bottlenecks and latency issues.

The following FLOWX.AI services use Jaeger tracing:

1. **scheduler-core**
2. **customer-management-plugin**
3. **document-plugin**
4. **notification-plugin**
5. **process-engine**

Environment variables to be set for tracing:

- `APPLICATION_JAEGER_ENABLED` - environment variable used to enable or disable Jaeger tracing
- `APPLICATION_JAEGER_PREFIX` - environment variable used to change the name in the Jaeger dashboard

License model

A license model is a set of rules and regulations governing how software can be used, distributed, and modified. It also outlines the rights and responsibilities of the software user and the software developer. Common license models include open source, freeware, shareware, and commercial software.

Most of the **third-party components used by FLOWX.AI** are under **Apache License 2.0** source code.

Third-party components

Third-party components are software components or libraries that are not part of FLOWX.AI but are instead created by another company or individual and used in a development project.

These components can range from databases and operating systems to user interface components and libraries that provide support for a specific feature or task.

Third party components are components such as libraries, frameworks, APIs, etc.

» [Third-party components](#)

[Was this page helpful?](#)

PLATFORM SETUP GUIDES / FLOWX.AI Engine setup guide / Advancing Controller setup guide

This guide provides step-by-step instructions to help you configure and deploy the Advancing Controller effectively.

Infrastructure prerequisites

Advancing controller requires the following components to be set up before it can be started:

- **FLOWX.AI Engine deployment** - the Advancing Controller is dependent on the FLOWX.AI Engine and must be deployed in the same environment, refer to the FLOWX.AI Engine setup guide for more information on how to set up the Engine
- **DB instance** - the Advancing Controller uses a PostgreSQL or OracleDB as database instance

Dependencies

- Database
- Datasource
- FLOWX.AI Engine

Database configuration

Postgres

A basic Postgres configuration for Advancing:

```
postgresql:  
  enabled: true  
  postgresqlUsername: "postgres"  
  postgresqlPassword: ""  
  postgresqlDatabase: "advancing"  
  existingSecret: "postgresql-generic"  
  postgresqlMaxConnections: 200  
  persistence:  
    enabled: true  
    storageClass: standard-rwo  
    size: 20Gi  
  resources:  
    limits:  
      cpu: 1000m  
      memory: 1024Mi  
    requests:  
      memory: 256Mi  
      cpu: 100m  
  metrics:  
    enabled: true  
    serviceMonitor:  
      enabled: false  
    prometheusRule:
```

```
  enabled: false
  primary:
    nodeSelector:
      preemptible: "false"
```

⚠ CAUTION

If the parallel advancing configuration already exists, resetting the 'advancing' database must be done by executing the SQL command `DROP DATABASE advancing;`. Once the database has been dropped, the Liquibase script will automatically re-enable it.

Configuration

The following configuration details need to be added using environment variables:

Advancing controller uses a PostgreSQL or an Oracle database as a dependency.

- the user, password, connection link, and database name need to be configured correctly, if these details are not configured correctly, errors will occur at startup
- the datasource is configured automatically via a Liquibase script inside the engine. All updates will include migration scripts.

Configuring datasource

The following configuration details need to be added using environment variables:

- `SPRING_DATASOURCE_URL` - environment variable used to configure a data source URL for a Spring application, it typically contains the JDBC driver name, the server name, port number, and database name
- `SPRING_DATASOURCE_USERNAME` - environment variable used to set the username for the database connection, this can be used to connect to a database instance
- `SPRING_DATASOURCE_PASSWORD` - environment variable used to store the password for the database connection, this can be used to secure access to the database and ensure that only authorized users have access to the data
- `SPRING_JPA_DATABASE` - relevant because it is used to specify the type of database that the Spring application should connect to (accepted values: `oracle` or `postgresql`)
- `SPRING_JPA_PROPERTIES_HIBERNATE_DEFAULTSCHEMA` (! only for Oracle DBs) - specifies the default schema to use for the database (default value: `public`)

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.

CAUTION

It's important to keep in mind that the Advancing Controller is tightly integrated with the FLOWX.AI Engine. Therefore, it is important to ensure that both the Engine and the Advancing Controller are configured correctly and are in sync.

Was this page helpful?

PLATFORM SETUP GUIDES / FLOWX.AI Engine setup guide / Configuring access rights for Engine

Granular access rights can be configured for restricting access to the Engine component.

Two different access authorizations are provided, each with specified access scopes:

1. **Manage-processes** - for configuring access for running test processes

Available scopes:

- **edit** - users are able to start processes for testing and to test action rules

2. **Manage-instances** - for configuring access for manipulating process instances

Available scopes:

- **read** - users can view the list of process instances
- **admin** - users are able to retry an action on a process instance token

The Engine service is preconfigured with the following default users roles for each of the access scopes mentioned above:

- **manage-processes**

- edit:
 - ROLE_ADMIN_MANAGE_PROCESS_EDIT
 - ROLE_ADMIN_MANAGE_PROCESS_ADMIN
- admin:
 - ROLE_ADMIN_MANAGE_PROCESS_ADMIN
- **manage-instances**
 - read:
 - ROLE_ENGINE_MANAGE_INSTANCE_READ
 - ROLE_ENGINE_MANAGE_INSTANCE_ADMIN
 - admin:
 - ROLE_ENGINE_MANAGE_INSTANCE_ADMIN

DANGER

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_SCOPENAM  
E_ROLESALLOWED:NEEDED_ROLE_NAMES
```

Possible values for AUTHORIZATIONNAME: MANAGEPROCESSES, MANAGEINSTANCES.

Possible values for SCOPENAME: read, edit, admin.

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

```
SECURITY_ACCESSAUTHORIZATIONS_MANAGEINSTANCES_SCOPES_READ_ROLES  
ROLE_NAME_TEST
```

[Was this page helpful?](#)

PLATFORM SETUP GUIDES / FLOWX.AI Engine setup guide / Configuring access roles for processes

Access to a process definition

Setting up user role-based access on process definitions is done by configuring swimlanes on the process definition.

» [Swimlanes](#)

By default, all process nodes belong to the same swimlane. If more swimlanes are needed, they can be edited in the process definition settings panel.

Swimlane role settings apply to the whole process, the process nodes or the actions to be performed on the nodes.

First, the desired user roles need to be configured in the identity provider solution and users must be assigned the correct roles.

INFO

You can use the **Access management** tab under **General Settings** to administrate all the roles.

The screenshot shows the Flowx Admin interface. On the left, there is a sidebar with the following navigation:

- Media Library
- Plugins
- Task Manager
- All tasks
- Hooks
- Stages
- Allocation rules
- Out of office
- Notification templates
- Document templates
- General Settings
 - Generic Parameters
 - Integration management
 - Licensing
 - Access management (highlighted with a red box)
 - Users
 - Roles (highlighted with a blue box)
 - Groups
- Audit Log
- Platform status

On the right, a modal window titled "Roles" is open, displaying a list of roles with columns for Name and Description, and edit/delete icons:

Name	Description	Action
FLOWX_ADMIN	-	edit delete
FLOWX_BACKOFFICE	FLOWX_BACKOFFICE	edit delete
FLOWX_FRONTOFFICE	FLOWX_FRONTOFFICE	edit delete
FLOWX_ROLE	Permission to start processes	edit delete
FLOWX_SUPERVISOR	FLOWX_SUPERVISOR	edit delete
ROLE_ADMIN_MANAGE_C	Admin access - delete generic parameters menu item	edit delete
ROLE_ADMIN_MANAGE_C	Edit access - create/edit generic parameters menu item	edit delete
ROLE_ADMIN_MANAGE_C	Import access - import generic parameters menu item	edit delete
ROLE_ADMIN_MANAGE_C	Read access - view generic parameters menu item/ export generic parameters	edit delete
ROLE_ADMIN_MANAGE_IN	-	edit delete

🔥 DANGER

To be able to access the roles defined in the identity provider solution, a **service account** with appropriate permissions needs to be added in the identity provider. And the details of that service account **need to be set up in the platform configuration.**

The defined roles will then be available to be used in the process definition settings (**Permissions** tab) panel for configuring swimlane access.

A **Default** swimlane comes with two default permissions assigned based on a specific role.

The screenshot shows the FLOWX.AI process editor interface. At the top, there's a navigation bar with icons for Home, Process, Data, and Settings. Below the navigation bar, the title is "Amazing process | Version 1 | draft". On the left, there's a sidebar with icons for General, Sensitive data, Swimlanes, Permissions (which is highlighted in blue), and Task management. The main content area has tabs for "Choose swimlane" and "Amazing process > Default". Under the "Default" tab, there's a "Role" dropdown containing "SPECIFIC_ROLE" and a permission dropdown showing "Execute, Self Assign". At the bottom right of this section is a "Save permissions" button. To the left of the main content area, there's a sidebar with icons for General, Sensitive data, and Task management.

- **execute** - the user will be able to start process instances and run actions on them
- **self-assign** - the user can assign a process instance to them and start working on it



DANGER

This is valid for > 2.11.0 FLOWX.AI platform release.

Other **Permissions** can be added manually, depending on the needs of the user. Some permissions are needed to be configured so you can use features inside **Task Management** plugin. Specific roles need to be assigned separately on a few available process operations. These are:

- **view** - the user will be able to view process instance data
- **assign** - user can assign tasks to other users (this operation is only accessible through the **Task management** plugin)
- **unassign** - user can unassign tasks from other users (this operation is only accessible through the **Task management** plugin)
- **hold** - user can mark the process instance as on hold (this operation is only accessible through the **Task management** plugin)
- **unhold** - user can mark the process instance as not on hold (this operation is only accessible through the **Task management** plugin)

The screenshot shows the FLOWX.AI platform's configuration interface for a process. The top navigation bar includes tabs for General*, Sensitive data, Swimlanes, Permissions (which is currently selected), and Task management. On the left, there's a sidebar titled 'Choose swimlane' with a single item: 'Amazing process' under 'Default'. The main content area is titled 'Amazing Process > Default'. It shows a 'Role' section where 'SPECIFIC_ROLE' is listed with the permission 'Execute, Self Assign'. Below this, there's a search bar labeled 'Search by operations name' with a placeholder 'I'. To the right of the search bar is a list of permissions: View, Execute, Self Assign, Assign, and Unassign.

🔥 DANGER

< 2.11.0 platform release - if no role is configured on an operation, no restrictions will be applied.

Configuration examples

⚠ CAUTION

Valid for < 2.11.0 release version.

Regular user

Below you can find an example of configuration of roles for a regular user:

The screenshot shows a user interface for managing roles. At the top, it displays "Amazing Process > Default". Below this, there are two input fields: "Role" containing "ROLE_USER" and "Permissions" containing "Execute". There are "Add Role" and "Save permissions" buttons at the bottom.

Role	Permissions
ROLE_USER	Execute

Add Role Save permissions

Admin

Below you can find an example of configuration of roles for an admin user:

The screenshot shows a user interface for managing roles. At the top, it says "Amazing Process > Default". Below that, there's a section for "Role" with a dropdown menu containing "ROLE_ADMIN" and a permissions dropdown set to "Execute, Self Assign, View". There are "Add Role" and "Save permissions" buttons. A trash can icon is also visible.

⚠ CAUTION

! Starting with **2.11.0** release, specific roles are needed, otherwise, restrictions will be applied.

After setting up your preferred identity provider solution, you will need to add the desired access roles in the application configuration for the FLOWX Engine (using environment variables):

» [Authorization & access roles](#)

Restricting process instance access based on business filters

» 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.

Viewing processes instances

Active process instances and their related data can be viewed from the FLOWX Designer. A user needs to be assigned to a specific role in the identity provider solution to be able to view this information.

By default, this role is named `FL0WX_R0LE`, but its name can be changed from the application configuration of the Engine by setting the following environment variable:

`FL0WX_PROCESS_DEFAULTROLES`

When viewing process instance-related data, it can be configured whether to hide specific sensitive user data. This can be configured using the `FL0WX_DATA_ANONYMIZATION` environment variable.

Access to REST API

To restrict API calls by user role, you will need to add the user roles in the application config:

```
security:  
  pathAuthorizations:  
    -  
      path: "/api/**"  
      rolesAllowed: "ANY_AUTHENTICATED_USER" or  
      "USER_ROLE_FROM_IDENTITY_PROVIDER"
```

Was this page helpful?

PLATFORM SETUP GUIDES / FLOWX.AI Engine setup guide / Process instance indexing / Configuration guidelines

The configuration of Elasticsearch for process instances indexing depends on various factors related to the application load, the number of process instances, parallel requests, and indexed keys per process. Although the best approach to sizing and configuring Elasticsearch is through testing and monitoring under load, here are some guidelines to help you get started:

Indexing strategy

When deleting data in Elasticsearch, it's recommended to delete entire indices instead of individual documents. Creating multiple smaller indices provides the flexibility to delete entire indices of old data that are no longer needed.

- Advantages of Multiple Small Indices:
 - Fast indexing process.
 - Flexibility in cleaning up old data.
- Potential Drawbacks:
 - Hitting the maximum number of shards per node, resulting in exceptions when creating new indices.
 - Increased search response time and memory footprint.

Alternatively, you can create fewer indices that span longer periods of time, such as one index per year. This approach offers small search response times but may result in longer indexing times and difficulty in cleaning up and recovering data in case of failure.

» [What is indexing?](#)

Shard and replica configuration

The solution includes an index template that gets created with the settings from the process-engine app (name, shards, replicas) when running the app for the first time. This template controls the settings and mapping of all newly created indices.

» [What is sharding?](#)

» Index template

Once an index is created, you cannot update its number of shards and replicas. However, you can update the settings from the index template at runtime in Elasticsearch, and new indices will be created with the updated settings. Note that the mapping should not be altered as it is required by the application.

Recommendations for resource management

To manage functional indexing operations and resources efficiently, consider the following recommendations:

- Sizing indexes upon creation
- Balancing
- Delete unneeded indices
- Reindex large indices
- Force merge indices
- Shrink indices
- Combine indices

Sizing indexes upon creation

Recommendations:

- Start with monthly indexes that have 2 shards and 1 replica. This setup is typically sufficient for handling up to 200k process instances per day; ensures a parallel indexing in two main shards and has also 1 replica per each main

shard (4 shards in total). This would create 48 shards per year in the elastic search nodes; A lot less than the default 1000 shards, so you will have enough space for other indexes as well.

- If you observe that the indexing gets really, really slow, then you should look at the physical resources / shard size and start adapting the config.
- If you observe that indexing one monthly index gets massive and affects the performance, then think about switching to weekly indices.
- If you have huge spikes of parallel indexing load (even though that depends on the Kafka connect cluster configuration), then think about adding more main shards.
- Consider having at least one replica for high availability. However, keep in mind that the number of replicas is applied to each shard, so creating many replicas may lead to increased resource usage.
- Monitor the number of shards created and estimate when you might reach the maximum shards per node, taking into account the number of nodes in your cluster.

Balancing

When configuring index settings, consider the number of nodes in your cluster. The total number of shards (calculated by the formula: `primary_shards_number * (replicas_number + 1)`) for an index should be directly proportional to the number of nodes. This helps Elasticsearch distribute the load evenly across nodes and avoid overloading a single node. Avoid adding shards and replicas unnecessarily.

Delete unneeded indices

Deleting unnecessary indices reduces memory footprint, the number of used shards, and search time.

Reindex large indices

If you have large indices, consider reindexing them. Process instance indexing involves multiple updates on an initially indexed process instance, resulting in multiple versions of the same document in the index. Reindexing creates a new index with only the latest version, reducing storage size, memory footprint, and search response time.

Force merge indices

If there are indices with no write operations performed anymore, perform force merge to reduce the number of segments in the index. This operation reduces memory footprint and response time. Only perform force merge during off-peak hours when the index is no longer used for writing.

Shrink indices

If you have indices with many shards, consider shrinking them using the shrink operation. This reindexes the data into an index with fewer shards. Perform this operation during off-peak hours.

Combine indices

If there are indices with no write operations performed anymore (e.g., process_instance indices older than 6 months), combine these indices into a larger one and delete the smaller ones. Use the reindexing operation during off-peak hours. Ensure that write operations are no longer needed from the FLOWX platform for these indices.

Was this page helpful?

PLATFORM SETUP GUIDES / FLOWX.AI Engine setup guide / Old access roles

⚠ CAUTION

Deprecated since platform version 1.16.0

Old access roles

Access to a process definition

You can restrict access to process definitions by user roles. This can be done by setting the desired operation permissions on a process definition.

Start by adding the needed roles in the database. These need to match the roles configured in the identity provider solution. Each role can have one or more permissions defined on it. Permissions can be applied to all users or only to the owner of the specific resource (for example the person that started the process instance).

After saving a new process definition, you can also save specific user roles for it to restrict user access. Access rights can be defined on the following operations that can be performed on a process definition:

- starting a new instance of the process definition
- viewing the instance of that process definition

Here's an example of setting operation permissions for a process definition:

```
{  
    "START": ["PROCESS_START"],  
    "VIEW": ["PROCESS_VIEW", "PROCESS_VIEW_ALL"]  
}
```

where `START` and `VIEW` are the possible operations to be performed on the definitions and `PROCESS_START`, `PROCESS_VIEW`, `PROCESS_VIEW_ALL` are permissions stored in the database.

Access to actions from process definitions

Operation permissions can also be set on specific nodes in order to restrict the access to the actions defined on that node. This can be done similarly to setting operation permissions on process definitions. The operation name to be used for nodes is `NODE_RUN`.

As nodes also hold the definitions for the user interface, deciding which user role can see a certain UI template can also be done by using node permissions. The templates linked to a node can only be viewed by a user that has the `NODE_RUN` permission on that node, if the access on that node is restricted.

Access to a process definition

You can restrict access to process definitions by user roles. This can be done by setting the desired operation permissions on a process definition.

Start by adding the needed roles in the database. These need to match the roles configured in the identity provider solution. Each role can have one or more

permissions defined on it. Permissions can be applied to all users or only to the owner of the specific resource (for example the person that started the process instance).

After saving a new process definition, you can also save specific user roles for it to restrict user access.

Access rights can be defined on the following operations that can be performed on a process definition:

- starting a new instance of the process definition
- viewing the instance of that process definition

Here's an example of setting operation permissions for a process definition:

```
{  
    "START": ["PROCESS_START"],  
    "VIEW": ["PROCESS_VIEW", "PROCESS_VIEW_ALL"]  
}
```

where `START` and `VIEW` are the possible operations to be performed on the definitions and `PROCESS_START`, `PROCESS_VIEW`, `PROCESS_VIEW_ALL` are permissions stored in the database.

Access to actions from process definitions

Operation permissions can also be set on specific nodes in order to restrict the access to the actions defined on that node. This can be done similarly to setting operation permissions on process definitions. The operation name to be used for nodes is `NODE_RUN`.

As nodes also hold the definitions for the user interface, deciding which user role can see a certain UI template can also be done by using node permissions. The templates linked to a node can only be viewed by a user that has the `NODE_RUN` permission on that node, if the access on that node is restricted.

[Was this page helpful?](#)

PLATFORM SETUP GUIDES / Access management / Configuring an IAM solution

Recommended Keycloak setup

To configure a minimal required Keycloak setup, follow these steps:

- [Create a new realm](#)
 - Define available roles and realm-level roles assigned to new users.
- [Create/import user roles and groups](#)
- [Create new users](#)
- [Add clients](#)
 - Configure access type, valid redirect URIs, and enable necessary flows.
- [Add role mappers](#)
- [Add service accounts](#)
 - Set up **admin**, **task management**, and **process engine** service accounts.

! INFO

Recommended keycloak version: **18.0.x**

For more detailed information, refer to the official Keycloak documentation:

» [Keycloak documentation](#)

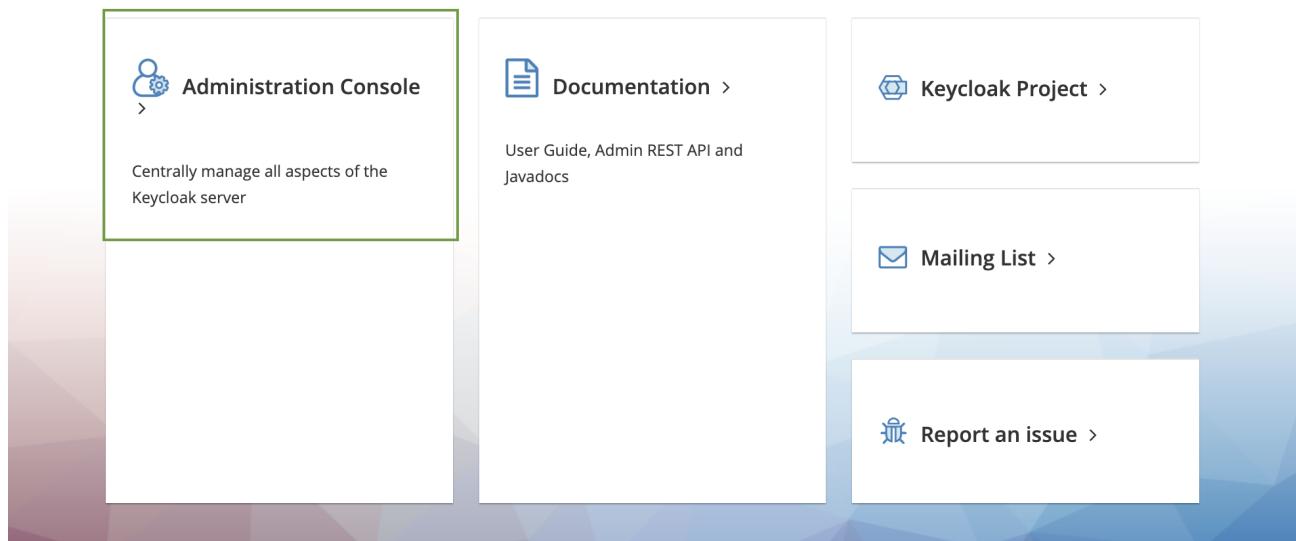
Creating a new realm

A realm is a space where you manage objects, including users, applications, roles, and groups. To create a new realm:

1. Log in to the **Keycloak Admin Console** using the appropriate URL for your environment (e.g., QA, development, production).



Welcome to **Keycloak**

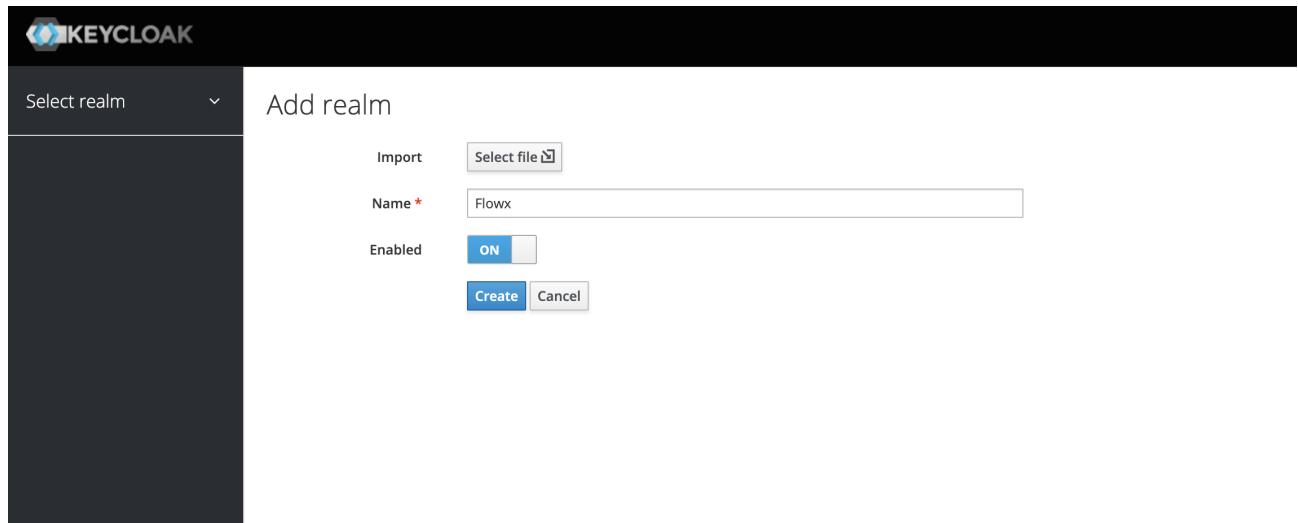


2. In the top left corner dropdown menu, click **Add Realm**.

! INFO

If you are logged in to the master realm this dropdown menu lists all the realms created. The **Add Realm** page opens.

3. Enter a realm name and click Create.



4. Configure the realm settings (**Realm Settings → Tokens**), such as SSO session idle and access token lifespan, according to your organization's needs:

- **SSO Session idle** - suggested: **30 Minutes**
- **Access Token Lifespan** - suggested: **30 Minutes**

The screenshot shows the Flowx configuration interface with the 'Tokens' tab selected. The interface is divided into several sections:

- General:** Includes fields for 'Default Signature Algorithm' (with a dropdown menu) and 'Revoke Refresh Token' (with an 'OFF' switch).
- SSO Session Idle:** Set to 30 Minutes.
- SSO Session Max:** Set to 30 Minutes.
- Offline Session Idle:** Set to 30 Minutes.
- Offline Session Max:** Set to OFF.
- Access Token Lifespan:** Set to 30 Minutes.
- Access Token Lifespan For Implicit Flow:** Set to 15 Minutes.

Creating/importing user groups and roles

You can either create or import a user group into a realm. We prepared a [script](#) that helps you to import a **super admin group** provided with the necessary **default user roles**.

You can create or import user groups into a realm. If you choose to import, follow the provided [script](#) to import a **super admin group**(`SUPER_ADMIN_USERS`) with **default user roles**. After importing, add an admin user to the group and assign the necessary roles.

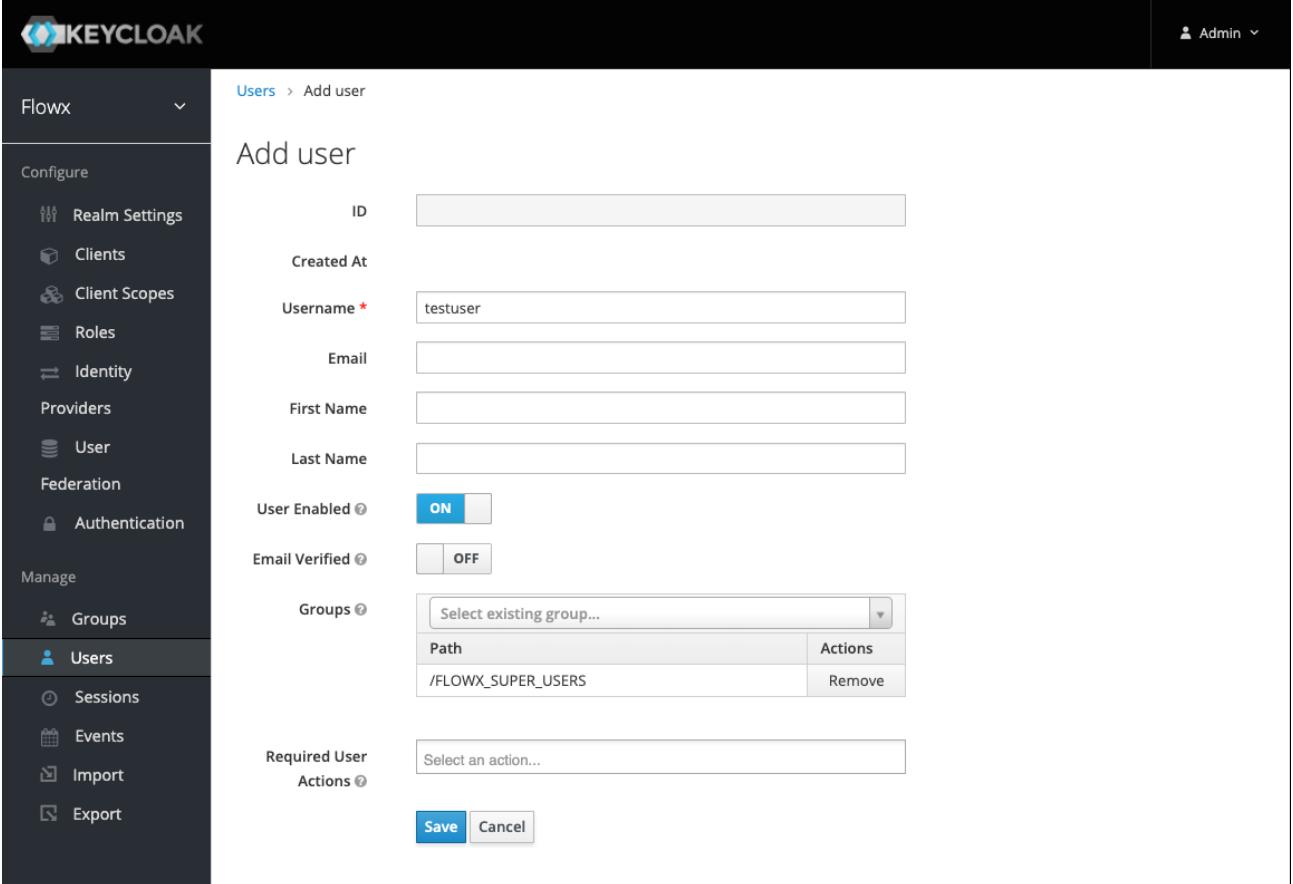
Make sure to validate the imported roles by checking the following section:

» [Default roles](#)

Creating new users

To create a new user in a realm and generate a temporary password:

1. In the left menu bar, click **Users** to open the user list page.
2. On the right side of the empty user list, click **Add User**.
3. Fill in the required fields, including the **username**, and ensure **Email Verified** is set to **ON**.
4. In the **Groups** field, choose a group from the dropdown menu, in our case:
/FLOWX_SUPER_USERS.



The screenshot shows the 'Add user' interface in Keycloak. The left sidebar is titled 'Flowx' and includes sections for Configure (Realm Settings, Clients, Client Scopes, Roles, Identity, Providers, User, Federation, Authentication) and Manage (Groups, Users, Sessions, Events, Import, Export). The 'Users' option is currently selected. The main 'Add user' page has fields for ID (empty), Created At (empty), Username * (set to 'testuser'), Email (empty), First Name (empty), Last Name (empty), User Enabled (set to ON), Email Verified (set to OFF), Groups (dropdown menu showing 'Select existing group...' with '/FLOWX_SUPER_USERS' listed), and Required User Actions (empty). At the bottom are 'Save' and 'Cancel' buttons.

5. Save the user, go to the **Credentials** tab, and set a temporary password.

The screenshot shows the Keycloak administration interface. On the left, there's a sidebar with a 'Flowx' dropdown and several configuration sections: Realm Settings, Clients, Client Scopes, Roles, Identity, Providers, User, Federation, and Authentication. Under 'Manage', there are links for Groups and Users, with 'Users' being the active one. The main content area shows a navigation path 'Users > testuser' and the title 'Testuser'. Below the title is a trash can icon. A horizontal menu bar includes 'Details', 'Attributes', 'Credentials' (which is highlighted in blue), 'Role Mappings', 'Groups', 'Consents', and 'Sessions'. The 'Credentials' tab leads to a 'Manage Credentials' section with a table header: 'Position', 'Type', 'User Label', 'Data', and 'Actions'. Below this is a 'Set Password' section. It contains two input fields for 'Password' and 'Confirmation', each with an eye icon to show/hide the text. There's also a 'Temporary' toggle switch, which is currently set to 'ON'. At the bottom of this section is a 'Set Password' button.

Adding clients

Clients represent trusted browser apps and web services in a realm. To add clients:

1. Click **Clients** in the top left menu, then click **Create**.
2. Set a client ID as `{example}-authenticate`, which will be used for login, logout, and refresh token operations.
3. Set the **Client Protocol** type as `openid-connect`.

The screenshot shows the Keycloak administration interface. On the left, a sidebar menu titled 'Flowx' contains options like 'Configure', 'Clients', 'Client Scopes', 'Roles', 'Identity', 'Providers', 'User', 'Federation', and 'Authentication'. The 'Clients' option is currently selected. The main content area is titled 'Add Client' and includes fields for 'Import' (with a 'Select file' button), 'Client ID' (set to 'test-authenticate'), 'Client Protocol' (set to 'openid-connect'), and 'Root URL' (empty). At the bottom are 'Save' and 'Cancel' buttons.

3. Open the newly created **client** and edit the following properties:

- Set **Access type** to **public** (this will not require a secret)
- Set **Valid redirect URLs**, specifying a valid URI pattern that a browser can redirect to after a successful login or logout, simple wildcards are allowed
- Enable **Direct Access Grants** and **Implicit Flow** by setting them to **ON**.
- Switch **Backchannel Logout Session Required** to **OFF**

The screenshot shows the Keycloak admin interface with the following details:

- Client ID:** test-authenticate
- Name:** (empty)
- Description:** (empty)
- Enabled:** ON
- Always Display in Console:** OFF
- Consent Required:** OFF
- Login Theme:** (dropdown menu)
- Client Protocol:** openid-connect
- Access Type:** public
- Standard Flow Enabled:** ON
- Implicit Flow Enabled:** ON
- Direct Access Grants Enabled:** ON
- OAuth 2.0 Device Authorization Grant Enabled:** OFF
- Front Channel Logout:** OFF
- Root URL:** (empty)
- * Valid Redirect URIs:** http://localhost:4200/*
- Base URL:** (empty)

4. Add **mappers** to {example}-authenticate client.

!(INFO)

Refer to the next section on how to add mappers and which mappers to clients.

Adding protocol mappers

Protocol mappers in Keycloak allow for the transformation of tokens and documents, enabling actions such as mapping user data into protocol claims or modifying requests between clients and the authentication server.

To enhance your clients, consider adding the following mappers:

- **Group Membership mapper** - `realm-groups`: This mapper can be utilized to map user groups to the authorization token.
- **User Attribute mapper** - `business filter mapper`: Use this mapper to map custom attributes, for example, mapping the **businessFilters** list, to the token claim.
- **User Realm role** - `realm-roles`: This mapper enables mapping a user's realm role to a token claim.

By incorporating these mappers, you can further customize and enrich the information contained within your tokens.

Group Membership mapper

To add a group membership mapper:

1. Navigate to **Clients** and select your desired client, in our case, `{example}-authenticate`

2. Go to the **Mappers** tab and click **Create** to create a new mapper.
3. Provide a descriptive **Name** for the mapper to easily identify its purpose.
4. Select **Group Membership** as the mapper type.
5. Set the token claim name for including groups in the token. In this case, set it as **groups**.

Clients > test-authenticate > Mappers > Create Protocol Mappers

Create Protocol Mapper

Protocol	openid-connect
Name	realm-groups
Mapper Type	Group Membership
Token Claim Name	groups
Full group path	ON
Add to ID token	ON
Add to access token	ON
Add to userinfo	ON
Save Cancel	

By configuring the group membership mapper, you will be able to include the user's group information in the token for authorization purposes.

User Attribute mapper

To include custom attributes such as **business filters** in the token claim, you can add a user attribute mapper with the following settings:

1. Go to the desired client, **{example}-authenticate**, and navigate to the **Mappers** section.

2. Click on **Create** to create a new mapper.
3. Configure the following settings for the user attribute mapper:

- **Mapper Type:** User Attribute
- **User Attribute:** businessFilters
- **Token Claim Name:** attributes.businessFilters
- **Add to ID token:** OFF
- **Multivalued:** ON

The screenshot shows the Keycloak interface for managing a 'Business Filter Mapper'. The top navigation bar includes 'Clients' (selected), 'flowx-platform-authenticate', 'Mappers', and 'business filter mapper'. The main title is 'Business Filter Mapper' with a trash icon. The configuration form contains the following fields:

Protocol	openid-connect
ID	b1fd6322-bec6-4793-b598-eb3edf4968a8
Name	business filter mapper
Mapper Type	User Attribute
User Attribute	businessFilters
Token Claim Name	attributes.businessFilters
Claim JSON Type	Select One... ▾
Add to ID token	<input type="checkbox"/> OFF
Add to access token	<input checked="" type="checkbox"/> ON
Add to userinfo	<input checked="" type="checkbox"/> ON
Multivalued	<input checked="" type="checkbox"/> ON
Aggregate attribute values	<input type="checkbox"/> OFF

At the bottom are 'Save' and 'Cancel' buttons.

By adding this user attribute mapper, the custom attribute "businessFilters" will be included in the token claim under the name "attributes.businessFilters". This will

allow you to access and utilize the business filters information within your application.

You can find more information about business filters in the following section:

» **Business filters**

User realm role

Add **roles mapper** to `{example}-authenticate` client - so roles will be available on the OAuth user info response.

To add a roles mapper, follow these steps:

1. Go to the desired client, `{example}-authenticate`, and navigate to the Mappers section.
2. Click on **Create** to create a new mapper.
3. Configure the following settings for the user attribute mapper:
 - **Mapper Type:** User Realm Role
 - **Token Claim Name:** role
 - **Add to userinfo:** ON

By adding this roles mapper, the assigned realm roles of the user will be available in the OAuth user info response under the claim name "roles". This allows you to access and utilize the user's realm roles within your application.

Please note that you can repeat these steps to add multiple roles mappers if you need to include multiple realm roles in the token claim.

Realm-roles

Protocol: openid-connect
ID: cb34d468-2d94-4f40-9153-fc0c325564ed
Name: realm-roles
Mapper Type: User Realm Role
Realm Role prefix:
Multivalued: ON
Token Claim Name: roles
Claim JSON Type: Select One...
Add to ID token: OFF
Add to access token: OFF
Add to userinfo: ON
Save Cancel

Examples

Login

```
curl --location --request POST
'http://localhost:8080/realms/flowx/protocol/openid-
connect/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=password' \
--data-urlencode 'username=admin@flowx.ai' \
--data-urlencode 'password=password' \
--data-urlencode 'client_id= example-authenticate'
```

Refresh token

```
curl --location --request POST
'http://localhost:8080/realms/flowx/protocol/openid-
connect/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=refresh_token' \
```

```
--data-urlencode 'client_id= example-authenticate' \
--data-urlencode 'refresh_token=ACCESS_TOKEN'
```

User info

```
curl --location --request GET
'localhost:8080/realmms/flowx/protocol/openid-
connect/userinfo' \
--header 'Authorization: Bearer ACCESS_TOKEN' \
```

Authorizing client

Add `{example}-platform-authorize` client - it will be used to authorize rest requests to microservices and Kafka

- set **Client Protocol** to **openid-connect**
- set **Access type** as **confidential**
- disable **Direct Access Grants Enabled** - OFF
- **Valid Redirect URIs** - mandatory
- disable **Backchannel Logout Session Required** - OFF

Once you have configured these settings, the `{example}-platform-authorize` client will be created and can be used to authorize REST requests to microservices and Kafka within your application.

The screenshot shows the Flowx Platform Authorization configuration interface. The left sidebar has a dark theme with white icons and text. It includes sections for Configure (Realm Settings, Clients), Providers (User, Federation, Authentication), Manage (Groups, Users, Sessions, Events, Import, Export), and a general section (Client Scopes, Roles, Identity). The main area shows the 'Clients' section with the path 'Clients > flowx-platform-authorize'. The 'Settings' tab is selected, showing various configuration options for the client 'flowx-platform-authorize'. The configuration fields include:

- Client ID:** flowx-platform-authorize
- Name:** (empty)
- Description:** (empty)
- Enabled:** ON
- Always Display in Console:** OFF
- Consent Required:** OFF
- Login Theme:** (dropdown menu)
- Client Protocol:** openid-connect
- Access Type:** confidential
- Standard Flow Enabled:** ON
- Implicit Flow Enabled:** OFF
- Direct Access Grants Enabled:** OFF
- Service Accounts Enabled:** OFF
- OAuth 2.0 Device Authorization Grant Enabled:** OFF
- OIDC CIBA Grant Enabled:** OFF
- Authorization Enabled:** OFF
- Front Channel Logout:** OFF
- Root URL:** (empty)
- * Valid Redirect URIs:** http://localhost:8086/*

Minimal auth config for microservices

```
security:
  type: oauth2
  basic:
    enabled: false
  oauth2:
    base-server-url: http://localhost:8080
    realm: flowx
    client:
      access-token-uri: ${security.oauth2.base-server-
url}/realms/${security.oauth2.realm}/protocol/openid-connect/token
      client-id: example-authorize
      client-secret: CLIENT_SECRET
    resource:
      user-info-uri: ${security.oauth2.base-server-
url}/realms/${security.oauth2.realm}/protocol/openid-connect/userinfo
```

Adding service accounts

ⓘ INFO

What is a service account?

A service account is an account that grants direct access to the Keycloak API for a specific component.

Admin service account

The admin microservice requires an admin service account to make direct calls to the Keycloak API.

Follow these steps to add an **admin service account**:

1. Add a new client by selecting **Clients** then click **Create**.

The screenshot shows the Keycloak administration interface. The left sidebar has a 'Flowx' dropdown and sections for 'Configure' (Realm Settings, Clients, Client Scopes, Roles, Identity Providers, User Federation, Authentication) and 'Manage' (Groups, Users, Sessions, Events). The 'Clients' section is currently selected. The main area is titled 'Add Client' and contains fields for 'Import' (with a 'Select file' button), 'Client ID' (marked with a red asterisk), 'Client Protocol' (set to 'openid-connect'), and 'Root URL'. At the bottom are 'Save' and 'Cancel' buttons.

2. Next, set **Access type** as **confidential** and enable **Service Accounts**.

Admin-service-account 

Settings Roles Client Scopes  Mappers  Scope  Revocation Sessions  Offline Access  Installation 

Client ID 	admin-service-account
Name 	
Description 	
Enabled 	<input checked="" type="button"/> ON <input type="button"/> OFF
Consent Required 	<input type="button"/> OFF
Login Theme 	
Client Protocol 	openid-connect
Access Type 	confidential
Standard Flow Enabled 	<input checked="" type="button"/> ON <input type="button"/> OFF
Implicit Flow Enabled 	<input type="button"/> OFF
Direct Access Grants Enabled 	<input checked="" type="button"/> ON <input type="button"/> OFF
Service Accounts Enabled 	<input checked="" type="button"/> ON <input type="button"/> OFF

3. Go to **Clients** → **realm-management** → **Roles** and add the following **service account client roles** under **realm-management**:

- **view-users**
- **query-groups**
- **query-users**

4. Assign the necessary **service account roles**:

Clients > flowx-admin-sa

Flowx-admin-sa trash

Settings Credentials Keys Roles Client Scopes ? Mappers ? Scope ? Revocation Sessions ? Offline Access ? Clustering Installation ?

Service Account Roles ?

Service Account

Service Account User ?

service-account-flowx-admin-service-account

Service Account Roles

Realm Roles	Available Roles ?	Assigned Roles ?	Effective Roles ?
	FLOWX_ADMIN FLOWX_BACKOFFICE FLOWX_FRONTOFFICE FLOWX_ROLE FLOWX_SUPERVISOR	default-roles-flowx	default-roles-flowx offline_access uma_authorization
Client Roles	realm-management	query-groups query-users view-users	query-groups query-users view-users

In the provided example, the **admin service account** can have the following assigned roles, depending on the required access scopes:

- **manage-users**
- **query-users**
- **manage-realm**

! INFO

The admin service account does not require mappers as it doesn't utilize roles. Service account roles include client roles from the **realm-management**.

For detailed information, refer to the following section:

» **Configuring access rights for admin**



Task management service account

The task management microservice requires a service account to make direct calls to the Keycloak API. Follow these steps to add a task management service account:

1. Add a new client by selecting **Clients** then click **Create**.

The screenshot shows the Keycloak administration interface. The top navigation bar has the Keycloak logo and the text 'Admin'. The left sidebar is titled 'Flowx' and contains sections for 'Configure' (Realm Settings, Clients, Client Scopes, Roles, Identity Providers, User Federation, Authentication) and 'Manage' (Groups, Users, Sessions, Events). The 'Clients' section is currently selected. The main content area is titled 'Add Client' and contains fields for 'Import' (with a 'Select file' button), 'Client ID' (set to 'flowx'), 'Client Protocol' (set to 'openid-connect'), and 'Root URL' (empty). At the bottom are 'Save' and 'Cancel' buttons.

2. Next, set the following properties:

- **Access type** - confidential
- **Service Accounts Enabled** - ON

Task-man-service-account trash

[Settings](#) [Credentials](#) [Roles](#) [Client Scopes ?](#) [Mappers ?](#) [Scope ?](#) [Revocation](#)

[Installation ?](#) [Service Account Roles ?](#)

Client ID ?	task-man-service-account
Name ?	
Description ?	
Enabled ?	ON OFF
Consent Required ?	OFF
Login Theme ?	
Client Protocol ?	openid-connect
Access Type ?	confidential
Standard Flow Enabled	ON OFF
Implicit Flow Enabled ?	ON OFF
Direct Access Grants Enabled ?	ON OFF
Service Accounts Enabled ?	ON OFF

3. Go to **Clients** → **realm-management** → **Roles** and add the following **service account client roles**:

- **view-users**
- **query-groups**
- **query-users**

Clients > realm-management > Roles > view-users

View-users

Details Attributes Users in Role

Role Name	view-users
Description	#{role_view-users}
Composite Roles	<input checked="" type="checkbox"/> ON

Save **Cancel**

Composite Roles

Realm Roles	Available Roles	Associated Roles
	default-roles-flowx FLOWX_ADMIN FLOWX_BACKOFFICE FLOWX_FRONTOFFICE FLOWX_ROLE	
	Add selected »	<< Remove selected
Client Roles	Select a client...	

4. Configure a realm roles mapper:

Realm-roles

Protocol	openid-connect
ID	58cd5211-2b09-4d51-97d2-b2c65de1a159
Name	realm-roles
Mapper Type	User Realm Role
Realm Role prefix	
Multivalued	<input checked="" type="checkbox"/> ON
Token Claim Name	roles
Claim JSON Type	Select One... ▾
Add to ID token	<input type="checkbox"/> OFF
Add to access token	<input checked="" type="checkbox"/> ON
Add to userinfo	<input checked="" type="checkbox"/> ON

Save **Cancel**

5. Assign the necessary service account roles, including **FLOWX_ROLE**.

Clients > flowx-task-management-plugin-sa

Flowx-task-management-plugin-sa

Settings Credentials Keys Roles Client Scopes  Mappers  Scope  Revocation Sessions  Offline Access 
Clustering Installation  Service Account Roles 

Service Account

Service Account User  service-account-flowx-task-man-service-account

Service Account Roles

Realm Roles	Available Roles 	Assigned Roles 	Effective Roles 
	FLOWX_ADMIN FLOWX_BACKOFFICE FLOWX_FRONTOFFICE FLOWX_SUPERVISOR offline_access	default-roles-flowx FLOWX_ROLE	default-roles-flowx FLOWX_ROLE offline_access uma_authorization
Add selected >		« Remove selected	

Client Roles	Available Roles 	Assigned Roles 	Effective Roles 
	realm-management	x	
	create-client impersonation manage-authorization manage-clients manage-events	query-groups query-users view-users	query-groups query-users view-users
Add selected >		« Remove selected	

In the provided example, the **task management service account** can have the following assigned roles, depending on the required access scopes:

- **view-users**
- **query-groups**
- **query-users**

For more information, check the following section:

» [Configuring access rights for Task Management](#)

Process engine service account

The process engine requires a process engine service account to make direct calls to the Keycloak API.

 **INFO**

This service account is needed so the use of Start Catch Event node is possible.

Follow these steps to add a **process engine service account**:

1. Add a new client by selecting **Clients** then click **Create**.

Clients > flowx-process-engine-sa

Flowx-process-engine-sa

Settings Credentials Keys Roles Client Scopes ? Mappers ? Scope ? Revocation Sessions ?

Offline Access ? Clustering Installation ? Service Account Roles ?

Client ID ?	flowx-process-engine-sa
Name ?	
Description ?	
Enabled ?	<input checked="" type="button"/> ON <input type="button"/> OFF
Always Display in Console ?	<input type="button"/> OFF
Consent Required ?	<input type="button"/> OFF
Login Theme ?	
Client Protocol ?	openid-connect
Access Type ?	confidential
Standard Flow Enabled ?	<input type="button"/> OFF
Implicit Flow Enabled ?	<input type="button"/> OFF
Direct Access Grants Enabled ?	<input type="button"/> OFF
Service Accounts Enabled ?	<input checked="" type="button"/> ON <input type="button"/> OFF
OAuth 2.0 Device Authorization Grant Enabled ?	<input type="button"/> OFF
OIDC CIBA Grant Enabled ?	<input type="button"/> OFF

2. Next, set **Access type** as **confidential** and enable **Service Accounts**.

Admin-service-account 🗑

Settings Roles Client Scopes ⓘ Mappers ⓘ Scope ⓘ Revocation Sessions ⓘ Offline Access ⓘ Installation ⓘ

Client ID ⓘ	admin-service-account
Name ⓘ	
Description ⓘ	
Enabled ⓘ	<input checked="" type="button"/> ON <input type="button"/> OFF
Consent Required ⓘ	<input type="button"/> OFF
Login Theme ⓘ	
Client Protocol ⓘ	openid-connect
Access Type ⓘ	confidential
Standard Flow Enabled ⓘ	<input checked="" type="button"/> ON <input type="button"/> OFF
Implicit Flow Enabled ⓘ	<input type="button"/> OFF
Direct Access Grants Enabled ⓘ	<input checked="" type="button"/> ON <input type="button"/> OFF
Service Accounts Enabled ⓘ	<input checked="" type="button"/> ON <input type="button"/> OFF

! INFO

This service account does not require client roles.

3. Assign the necessary service account roles, including `FL0WX_ROLE`.

Clients > admin-service-account

Admin-service-account

The screenshot shows the 'Service Accounts' section for the 'admin-service-account'. It displays two main sections: 'Realm Roles' and 'Client Roles'.

- Realm Roles:**
 - Available Roles:** ROLE_CMS_CONTENT_ADMIN, ROLE_CMS_CONTENT_EDIT, ROLE_CMS_CONTENT_IMPORT, ROLE_CMS_CONTENT_READ, ROLE_CMS_TAXONOMIES_ADMIN.
 - Assigned Roles:** ROLE_ADMIN_MANAGE_PROCESS_READ, ROLE_ADMIN_MANAGE_USERS_ADMIN, ROLE_ADMIN_MANAGE_USERS_EDIT, ROLE_ADMIN_MANAGE_USERS_READ, uma_authorization.
 - Effective Roles:** offline_access, ROLE_ADMIN_MANAGE_CONFIG_ADMIN, ROLE_ADMIN_MANAGE_CONFIG_EDIT, ROLE_ADMIN_MANAGE_CONFIG_IMPORT, ROLE_ADMIN_MANAGE_CONFIG_READ.
- Client Roles:**
 - Available Roles:** realm-management.
 - Assigned Roles:** query-realms, query-users, realm-admin, view-authorization, view-realm.
 - Effective Roles:** create-client, impersonation, manage-authorization, manage-clients, manage-events.

Buttons for 'Add selected' and 'Remove selected' are visible between the role lists.

Was this page helpful?

PLATFORM SETUP GUIDES / Access management / Default roles

Below you can find the list of all the default roles that you can add or import into the Identity and Access Management solution to properly manage the access to all the FLOWX.AI microservices.

Default roles

A complete list of all the default roles based on modules (access scope):

Module	Scopes	Role default value

Module	Scopes	Role default value
manage-platform	read	ROLE_ADMIN_MANAGE_PLATFORM_READ
manage-platform	admin	ROLE_ADMIN_MANAGE_PLATFORM_ADMIN
manage-processes	import	ROLE_ADMIN_MANAGE_PROCESS_IMPORT
manage-processes	read	ROLE_ADMIN_MANAGE_PROCESS_READ
manage-processes	edit	ROLE_ADMIN_MANAGE_PROCESS_EDIT
manage-processes	admin	ROLE_ADMIN_MANAGE_PROCESS_ADMIN
manage-integrations	admin	ROLE_ADMIN_MANAGE_INTEGRATIONS_ADMIN
manage-integrations	read	ROLE_ADMIN_MANAGE_INTEGRATIONS_READ
manage-integrations	edit	ROLE_ADMIN_MANAGE_INTEGRATIONS_EDIT

Module	Scopes	Role default value
manage-integrations	import	ROLE_ADMIN_MANAGE_INTEGRATIONS_IMPORT
manage-configurations	import	ROLE_ADMIN_MANAGE_CONFIG_IMPORT
manage-configurations	read	ROLE_ADMIN_MANAGE_CONFIG_READ
manage-configurations	edit	ROLE_ADMIN_MANAGE_CONFIG_EDIT
manage-configurations	admin	ROLE_ADMIN_MANAGE_CONFIG_ADMIN
manage-users	read	ROLE_ADMIN_MANAGE_USERS_READ
manage-users	edit	ROLE_ADMIN_MANAGE_USERS_EDIT
manage-users	admin	ROLE_ADMIN_MANAGE_USERS_ADMIN
manage-processes	edit	ROLE_ENGINE_MANAGE_PROCESS_EDIT

Module	Scopes	Role default value
manage-processes	admin	ROLE_ENGINE_MANAGE_PROCESS_ADMIN
manage-instances	read	ROLE_ENGINE_MANAGE_INSTANCE_READ
manage-instances	admin	ROLE_ENGINE_MANAGE_INSTANCE_ADMIN
manage-licenses	read	ROLE_LICENSE_MANAGE_READ
manage-licenses	edit	ROLE_LICENSE_MANAGE_EDIT
manage-licenses	admin	ROLE_LICENSE_MANAGE_ADMIN
manage-contents	import	ROLE_CMS_CONTENT_IMPORT
manage-contents	read	ROLE_CMS_CONTENT_READ
manage-contents	edit	ROLE_CMS_CONTENT_EDIT

Module	Scopes	Role default value
manage-contents	admin	ROLE_CMS_CONTENT_ADMIN
manage-media-library	import	ROLE_MEDIA_LIBRARY_IMPORT
manage-media-library	read	ROLE_MEDIA_LIBRARY_READ
manage-media-library	edit	ROLE_MEDIA_LIBRARY_EDIT
manage-media-library	admin	ROLE_MEDIA_LIBRARY_ADMIN
manage-taxonomies	import	ROLE_CMS_TAXONOMIES_IMPORT
manage-taxonomies	read	ROLE_CMS_TAXONOMIES_READ
manage-taxonomies	edit	ROLE_CMS_TAXONOMIES_EDIT
manage-taxonomies	admin	ROLE_CMS_TAXONOMIES_ADMIN

Module	Scopes	Role default value
manage-tasks	read	ROLE_TASK_MANAGER_TASKS_READ
manage-hooks	import	ROLE_TASK_MANAGER_HOOKS_IMPORT
manage-hooks	read	ROLE_TASK_MANAGER_HOOKS_READ
manage-hooks	edit	ROLE_TASK_MANAGER_HOOKS_EDIT
manage-hooks	admin	ROLE_TASK_MANAGER_HOOKS_ADMIN
manage-process-allocation-settings	import	ROLE_TASK_MANAGER_PROCESS_ALLOCATION_SETTINGS_IMPORT
manage-process-allocation-settings	read	ROLE_TASK_MANAGER_PROCESS_ALLOCATION_SETTINGS_READ

Module	Scopes	Role default value
manage-process-allocation-settings	edit	ROLE_TASK_MANAGER_PROCESS_ALLOCATION
manage-process-allocation-settings	admin	ROLE_TASK_MANAGER_PROCESS_ALLOCATION
manage-out-of-office-users	import	ROLE_TASK_MANAGER_OOO_IMPORT
manage-out-of-office-users	read	ROLE_TASK_MANAGER_OOO_READ
manage-out-of-office-users	edit	ROLE_TASK_MANAGER_OOO_EDIT
manage-out-of-office-users	admin	ROLE_TASK_MANAGER_OOO_ADMIN

Module	Scopes	Role default value
manage-notification-templates	import	ROLE_NOTIFICATION_TEMPLATES_IMPORT
manage-notification-templates	read	ROLE_NOTIFICATION_TEMPLATES_READ
manage-notification-templates	edit	ROLE_NOTIFICATION_TEMPLATES_EDIT
manage-notification-templates	admin	ROLE_NOTIFICATION_TEMPLATES_ADMIN
manage-notifications	import	ROLE_MANAGE_NOTIFICATIONS_IMPORT
manage-notifications	read	ROLE_MANAGE_NOTIFICATIONS_READ
manage-notifications	edit	ROLE_MANAGE_NOTIFICATIONS_EDIT

Module	Scopes	Role default value
manage-notifications	admin	ROLE_MANAGE_NOTIFICATIONS_ADMIN
manage-document-templates	import	ROLE_DOCUMENT_TEMPLATES_IMPORT
manage-document-templates	read	ROLE_DOCUMENT_TEMPLATES_READ
manage-document-templates	edit	ROLE_DOCUMENT_TEMPLATES_EDIT
manage-document-templates	admin	ROLE_DOCUMENT_TEMPLATES_ADMIN

Importing roles

(!) INFO

You can import a super admin group and its default roles in Keycloak using the following script file.

(!) DOWNLOAD THE SCRIPT + ROLES:**Import Script**

You need to edit the following script parameters:

- `baseAuthUrl`
- `username`
- `password`
- `realm`
- `the name of the group for super admins`

The requests package is needed in order to run the script. It can be installed with the following command:

```
pip3 install requests
```

The script can be run with the following command:

```
python3 importUsers.py
```

[Was this page helpful?](#)

PLATFORM SETUP GUIDES / Audit setup guide

Introduction

This guide will walk you through the process of setting up the Audit service and configuring it to meet your needs.

Infrastructure prerequisites

The Audit service requires the following components to be set up before it can be started:

- **Docker engine** - version 17.06 or higher
- **Kafka** - version 2.8 or higher
- **Elasticsearch** - version 7.11.0 or higher

Dependencies

The Audit service is built as a Docker image and runs on top of Kafka and Elasticsearch. Therefore, these services must be set up and running before starting the Audit service.

- **Kafka configuration**
- **Authorization & access roles**
- **Elastic search**
- **Logging**

Configuration

Configuring Kafka

To configure the Kafka server for the Audit service, set the following environment variables:

- `SPRING_KAFKA_BOOTSTRAP_SERVERS` - address of the Kafka server, it should be in the format "host:port"
- `SPRING_KAFKA_CONSUMER_GROUP_ID` - the consumer group ID to be used for the audit logs
- `KAFKA_CONSUMER_THREADS` - the number of Kafka consumer threads to be used for processing audit logs
- `KAFKA_TOPIC_AUDIT_IN` - the topic key for receiving audit logs

Configuring Elasticsearch

To configure Elasticsearch, set the following environment variables:

- `SPRING_ELASTICSEARCH_REST_URIS` - the URL(s) of one or more Elasticsearch nodes to connect to
- `SPRING_ELASTICSEARCH_REST_DISABLESSL` - a boolean value that determines whether SSL should be disabled for Elasticsearch connections
- `SPRING_ELASTICSEARCH_REST_USERNAME` - the username to use for basic authentication when connecting to Elasticsearch
- `SPRING_ELASTICSEARCH_REST_PASSWORD` - the password to use for basic authentication when connecting to Elasticsearch

- `SPRING_ELASTICSEARCH_INDEX_SETTINGS_DATASTREAM` (used if ES is used across all dev environments) - the index settings for the datastreams that will be created in Elasticsearch

Configuring logging

To control the log levels, set the following environment variables:

- `LOGGING_LEVEL_ROOT` - the log level for the root spring boot microservice logs
- `LOGGING_LEVEL_APP` - the log level for app-level logs

CAUTION

Make sure to overwrite the placeholders (where needed) with the appropriate values before starting the service.

Was this page helpful?

PLATFORM SETUP GUIDES / CMS setup guide / Configuring access rights for CMS

Granular access rights can be configured for restricting access to the CMS component.

Two different access authorizations are provided, each with specified access scopes:

1. Manage-contents - for configuring access for manipulating CMS contents

Available scopes:

- import - users are able to import enumeration/substitution tags/ content models
- read - users are able to show enumeration/substitution tags/ content models, export enumeration/substitution tags/ content models
- edit - users are able to create/edit enumeration/substitution tags/ content models
- admin - users are able to delete enumeration/substitution tags/ content models

2. Manage-taxonomies - for configuring access for manipulating taxonomies

Available scopes

- read - users are able to show languages/source systems
- edit - users are able to edit languages/source systems
- admin - users are able to delete languages/source systems

3. Manage-media-library - for configuring access rights to use Media Library

Available scopes

- import - users are able to import assets
- read - users are able to view assets

- edit - users are able to edit assets
- admin - users are able to delete assets

The CMS service is preconfigured with the following default users roles for each of the access scopes mentioned above:

- **manage-contents**
 - import:
 - ROLE_CMS_CONTENT_IMPORT
 - ROLE_CMS_CONTENT_EDIT
 - ROLE_CMS_CONTENT_ADMIN
 - read:
 - ROLE_CMS_CONTENT_EDIT
 - ROLE_CMS_CONTENT_ADMIN
 - ROLE_CMS_CONTENT_READ
 - ROLE_CMS_CONTENT_IMPORT
 - edit:
 - ROLE_CMS_CONTENT_EDIT
 - ROLE_CMS_CONTENT_ADMIN
 - admin:
 - ROLE_CMS_CONTENT_ADMIN
- **manage-taxonomies**
 - import:
 - ROLE_CMS_TAXONOMIES_IMPORT
 - ROLE_CMS_TAXONOMIES_EDIT
 - ROLE_CMS_TAXONOMIES_ADMIN
 - read:

- ROLE_CMS_TAXONOMIES_READ
- ROLE_CMS_TAXONOMIES_IMPORT
- ROLE_CMS_TAXONOMIES_EDIT
- ROLE_CMS_TAXONOMIES_ADMIN
- edit:
 - ROLE_CMS_TAXONOMIES_EDIT
 - ROLE_CMS_TAXONOMIES_ADMIN
- admin:
 - ROLE_CMS_TAXONOMIES_ADMIN
- **manage-media-library**
 - import:
 - ROLE_MEDIA_LIBRARY_IMPORT
 - ROLE_MEDIA_LIBRARY_EDIT
 - ROLE_MEDIA_LIBRARY_EDIT
 - read:
 - ROLE_MEDIA_LIBRARY_READ
 - ROLE_MEDIA_LIBRARY_EDIT
 - ROLE_MEDIA_LIBRARY_ADMIN
 - ROLE_MEDIA_LIBRARY_IMPORT
 - edit:
 - ROLE_MEDIA_LIBRARY_EDIT
 - ROLE_MEDIA_LIBRARY_ADMIN
 - admin:
 - ROLE_MEDIA_LIBRARY_ADMIN

 **CAUTION**

The needed roles should 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: MANAGECONTENTS, MANAGETAXONOMIES.

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

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

```
SECURITY_ACCESSAUTHORIZATIONS_MANAGECONTENTS_SCOPES_IMPORT_ROLE_CMS_CONTENT_IMPORT
```

Was this page helpful?

PLATFORM SETUP GUIDES / Events gateway setup guide

Introduction

This guide will walk you through the process of setting up the events-gateway service.

Infrastructure prerequisites

Before proceeding with the setup, ensure that the following components have been set up:

- **Redis** - version 6.0 or higher
- **Kafka** - version 2.8 or higher

Dependencies

- **Kafka** - used for event communication
- **Redis** - used for caching

Configuration

Configuring Kafka

Set the following Kafka-related configurations using environment variables:

- `SPRING_KAFKA_BOOTSTRAP_SERVERS` - the address of the Kafka server, it should be in the format "host:port"

Groupd IDs

The configuration parameters "KAFKA_CONSUMER_GROUP_ID*" are used to set the consumer group name for Kafka consumers that consume messages from topics. Consumer groups in Kafka allow for parallel message processing by distributing the workload among multiple consumer instances. By configuring the consumer group ID, you can specify the logical grouping of consumers that work together to process messages from the same topic, enabling scalable and fault-tolerant message consumption in your Kafka application.

Configuration Parameter

KAFKA_CONSUMER_GROUP_ID_PROCESS_ENGINE_COMMANDS_MESSAGE

KAFKA_CONSUMER_GROUP_ID_PROCESS_ENGINE_COMMANDS_DISCONNECT

Configuration Parameter

KAFKA_CONSUMER_GROUP_ID_PROCESS_ENGINE_COMMANDS_CONNECT

KAFKA_CONSUMER_GROUP_ID_PROCESS_TASK_COMMANDS

Threads

The configuration parameters "KAFKA_CONSUMER_THREADS*" are utilized to specify the number of threads assigned to Kafka consumers for processing messages from topics. These parameters allow you to fine-tune the concurrency and parallelism of your Kafka consumer application, enabling efficient and scalable message consumption from Kafka topics.

Configuration Parameter

De
Va

Configuration Parameter	Description
KAFKA_CONSUMER_THREADS_PROCESS_ENGINE_COMMANDS_MESSAGE	10
KAFKA_CONSUMER_THREADS_PROCESS_ENGINE_COMMANDS_DISCONNECT	5
KAFKA_CONSUMER_THREADS_PROCESS_ENGINE_COMMANDS_CONNECT	5
KAFKA_CONSUMER_THREADS_TASK_COMMANDS	10
KAFKA_AUTH_EXCEPTION_RETRY_INTERVAL	10

Kafka topics related to process instances

Configuration Parameter	
KAFKA_TOPIC_EVENTS_GATEWAY_PROCESS_INSTANCE_IN_MESSAGE	ai
KAFKA_TOPIC_EVENTS_GATEWAY_PROCESS_INSTANCE_IN_DISCONNECT	ai
KAFKA_TOPIC_EVENTS_GATEWAY_PROCESS_INSTANCE_IN_CONNECT	ai

Kafka topics related to tasks

Configuration Parameter	
KAFKA_TOPIC_EVENTS_GATEWAY_TASK_IN_MESSAGE	ai.flowx.eventsga

Configuring authorization & access roles

Set the following environment variables to connect to the identity management platform:

Configuration Parameter	Description
SECURITY_OAUTH2_BASE_SERVER_URL	Base URL of the OAuth2 server
SECURITY_OAUTH2_CLIENT_CLIENT_ID	Client ID for OAuth2 authentication

Configuration Parameter	Description
SECURITY_OAUTH2_CLIENT_CLIENT_SECRET	Client secret for OAuth2 authentication
SECURITY_OAUTH2_REALM	Realm for OAuth2 authentication

Redis

The process engine sends the messages to the events-gateway, which is responsible for sending them to Redis.

Configuration Parameter	Description
SPRING_REDIS_HOST	Hostname of the Redis server
SPRING_REDIS_PASSWORD	Password for Redis server
SPRING_REDIS_TTL	Time-to-live for Redis keys (default value:5000000 # milliseconds)

Master replica

The events-gateway can be configured to communicate with Redis using the MASTER_REPLICA replication mode by configuring the following property:

spring.redis.sentinel.nodes: replica1, replica2, replica3, etc...

Example

```
spring.redis.sentinel.nodes=host1:26379,host2:26379,host3:26379
```

In the above example, the Spring Boot application will connect to three Redis Sentinel nodes: host1:26379, host2:26379, and host3:26379.

The property value should be a comma-separated list of host:port pairs, where each pair represents the hostname or IP address and the port number of a Redis Sentinel node.

ⓘ INFO

By default, Redis is standalone, so the configuration with `redis-replicas` is optional for high load use cases.

In the context of Spring Boot and Redis Sentinel integration, the `spring.redis.sentinel.nodes` property is used to specify the list of Redis Sentinel nodes that the Spring application should connect to. These nodes are responsible for monitoring and managing Redis instances.

Configuring logging

The following environment variables could be set in order to control log levels:

Configuration Parameter	Description

Configuration Parameter	Description
LOGGING_LEVEL_ROOT	Logging level for the root Spring Boot microservice logs
LOGGING_LEVEL_APP	Logging level for the application-level logs

Was this page helpful?

PLATFORM SETUP GUIDES / License engine setup guide / Configuring access rights for License

Granular access rights can be configured for restricting access to the License component.

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

1. **Manage-licenses** - for configuring access for managing license related details

Available scopes:

- read - users are able to view the license report

- edit - users are able to update the license model and sync license data
- admin - users are able to download the license data

The License component is preconfigured with the following default users roles for each of the access scopes mentioned above:

- manage-licenses
 - read:
 - ROLE_LICENSE_MANAGE_READ
 - ROLE_LICENSE_MANAGE_EDIT
 - ROLE_LICENSE_MANAGE_ADMIN
 - edit:
 - ROLE_LICENSE_MANAGE_EDIT
 - ROLE_LICENSE_MANAGE_ADMIN
 - admin:
 - ROLE_LICENSE_MANAGE_ADMIN

DANGER

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_SCOPENAM  
E_ROLESALLOWED: NEEDED_ROLE_NAMES
```

Possible values for `AUTHORIZATIONNAME: MANAGELICENSES`.

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

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

```
SECURITY_ACCESSAUTHORIZATIONS_MANAGELICENSES_SCOPES_READ_ROLES  
ROLE_NAME_TEST
```

Was this page helpful?

PLATFORM SETUP GUIDES / License engine setup guide / Configuring access roles

CAUTION

Deprecated since platform version 1.16.0

The License engine is able to offer different levels of accessing license related information.

In order to restrict API calls by user role you will need to add the user roles in the application config. You can configure separate roles for the provided API base routes:

```
- path: "/api/report"
  rolesAllowed: ${LICENSE_VIEW}
- path: "/api/license-model"
  rolesAllowed: ${LICENSE_MANAGER}
- path: "/api/sync/**"
  rolesAllowed: ${LICENSE_SUPER_MANAGER}
- path: "/api/data/**"
  rolesAllowed: ${LICENSE_SUPER_USER}
```

- **LICENSE_VIEW** - users with this role will be able to view the status of the license (just the usage info, no extra details)
- **LICENSE_MANAGER** - users with this role will be able to configure the license
- **LICENSE_SUPER_MANAGER** - users with this role will be able to trigger sync for the existing license
- **LICENSE_SUPER_USER** - users with this role will be able to request a detailed report with details of custom identifiers and dates when they appear (this can contain personal data)

Was this page helpful?

PLATFORM SETUP GUIDES / Scheduler setup guide

Introduction

This guide will walk you through the process of setting up the Scheduler service using a Docker image.

Infrastructure prerequisites

- **MongoDB** - version 4.4 or higher for storing taxonomies and contents
- **Kafka** - version 2.8 or higher

Dependencies

- **MongoDB** database
- ability to connect to a Kafka instance used by the engine

The service comes with most of the needed configuration properties filled in, but there are a few that need to be set up using some custom environment variables.

Dependencies

MongoDB helm example

Basic MongoDB configuration - helm values.yaml

```
scheduler-mdb:  
  existingSecret: {{secretName}}  
  mongodbDatabase: {{SchedulerDatabaseName}}  
  mongodbUsername: {{SchedulerDatabaseUser}}  
  persistence:  
    enabled: true
```

```
mountPath: /bitnami/mongodb
size: 4Gi
replicaSet:
  enabled: true
  name: rs0
  pdb:
    enabled: true
    minAvailable:
      arbiter: 1
      secondary: 1
  replicas:
    arbiter: 1
    secondary: 1
  useHostnames: true
serviceAccount:
  create: false
usePassword: true
```

🔥 DANGER

This service needs to connect to a Mongo database that has replicas, in order to work correctly.

Configuration

Configuring MongoDB

The MongoDB database is used to persist scheduled messages until they are sent back. The following configurations need to be set using environment variables:

- `SPRING_DATA_MONGODB_URI` - the URI for the MongoDB database

Configuring Kafka

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`

Each action available in the service corresponds to a Kafka event. A separate Kafka topic must be configured for each use-case.

⚠ CAUTION

Make sure the topics configured for this service don't follow the engine pattern.

Configuring 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 SETUP GUIDES / Data search service setup guide

Introduction

This guide will walk you through the process of setting up the Search Data service using a Docker image.

Infrastructure prerequisites

Before proceeding with the setup, ensure that the following components have been set up:

- **Redis** - version 6.0 or higher
- **Kafka** - version 2.8 or higher
- **Elasticsearch** - version 7.11.0 or higher

Dependencies

- **Kafka** - used for communication with the engine
- **Elasticsearch** - used for indexing and searching data
- **Redis** - used for caching

Configuration

Configuring Kafka

Set the following Kafka-related configurations using environment variables:

- `SPRING_KAFKA_BOOTSTRAP_SERVERS` - address of the Kafka server
- `KAFKA_TOPIC_DATA_SEARCH_IN`
- `KAFKA_TOPIC_DATA_SEARCH_OUT`
- `KAFKA_CONSUMER_THREADS` - the number of Kafka consumer threads

Configuring Elasticsearch

Set the following Elasticsearch-related configurations using environment variables:

- `SPRING_ELASTICSEARCH_REST_URIS`
- `SPRING_ELASTICSEARCH_REST_DISABLESSL`
- `SPRING_ELASTICSEARCH_REST_USERNAME`
- `SPRING_ELASTICSEARCH_REST_PASSWORD`
- `SPRING_ELASTICSEARCH_INDEX_SETTINGS_NAME` - the index can be customized for data-search and it should be similar to what is configured on the process-engine

Configuring authorization & access roles

Set the following environment variables to connect to the identity management platform:

- SECURITY_OAUTH2_BASE_SERVER_URL
- SECURITY_OAUTH2_CLIENT_CLIENT_ID
- SECURITY_OAUTH2_REALM

Configuring logging

The following environment variables could be set in order to control log levels:

- LOGGING_LEVEL_ROOT - for root spring boot microservice logs
- LOGGING_LEVEL_APP - for app level logs

Elasticsearch

Data search in Elasticsearch runs against an index pattern representing multiple indices. The index pattern is derived from the configuration property:

spring.elasticsearch.index-settings.name

Below is an example of a filter to be used in Kibana (as generated by data search):

```
{  
  "query": {  
    "bool": {  
      "adjust_pure_negative": true,  
      "boost": 1,  
      "must": [  
        {  
          "nested": {  
            "boost": 1,  
            "path": "category"  
          }  
        }  
      ]  
    }  
  }  
}
```

```
"ignore_unmapped": false,
"path": "keyIdentifiers",
"query": {
    "bool": {
        "adjust_pure_negative": true,
        "boost": 1,
        "must": [
            {
                "match": {
                    "keyIdentifiers.key.keyword": {
                        "auto_generate_synonyms_phrase_query": true,
                        "boost": 1,
                        "fuzzy_transpositions": true,
                        "lenient": false,
                        "max_expansions": 50,
                        "operator": "OR",
                        "prefix_length": 0,
                        "query": "astonishingAttribute",
                        "zero_terms_query": "NONE"
                    }
                }
            },
            {
                "match": {
                    "keyIdentifiers.originalValue.keyword": {
                        "auto_generate_synonyms_phrase_query": true,
                        "boost": 1,
                        "fuzzy_transpositions": true,
                        "lenient": false,
                        "max_expansions": 50,
                        "operator": "OR",
                        "prefix_length": 0,

```

```
        "query": "OriginalGangsta",
        "zero_terms_query": "NONE"
    }
}
]
}
},
"score_mode": "none"
}
},
{
"terms": {
"boost": 1,
"processDefinitionName.keyword": [
"TEST_PORCESS_NAME_0",
"TEST_PORCESS_NAME_1"
]
}
}
]
}
}
}
```

Was this page helpful?