

---

**FLOW<sup>X</sup>.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 /  Custom Plugins / WYSIWYG editor
- 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

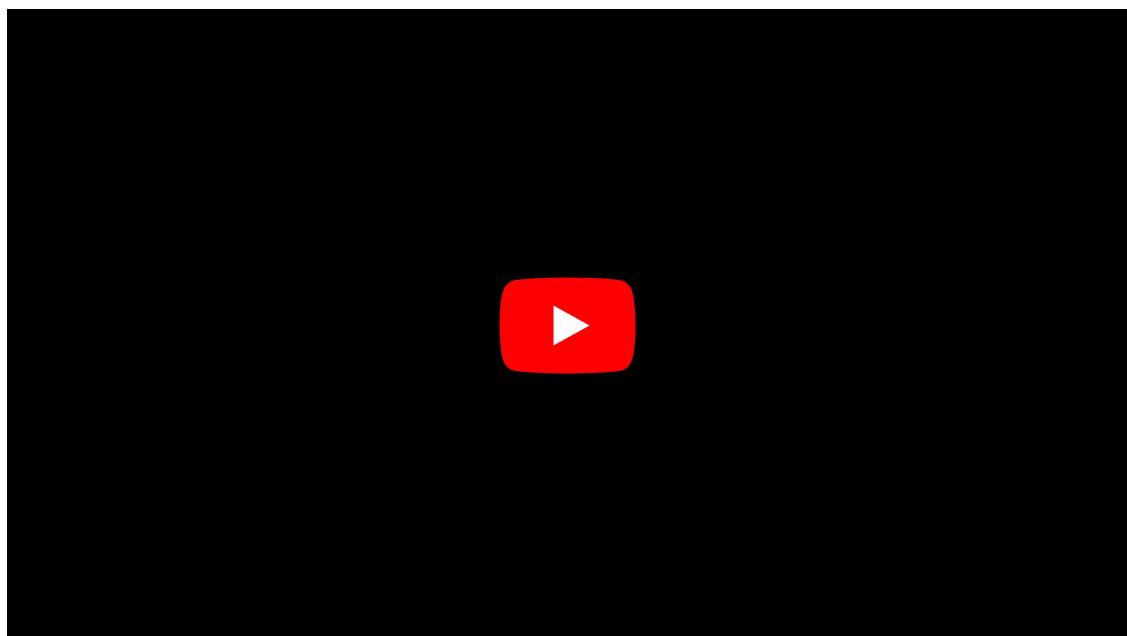
The fallback content to display on prerendering  
built on a

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

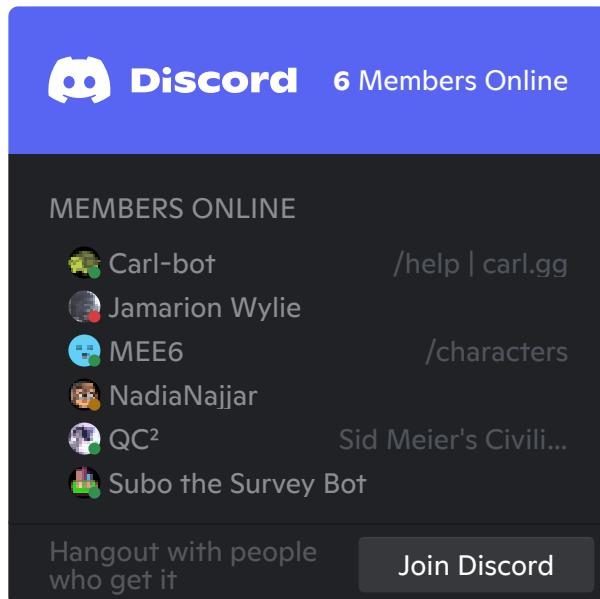
Find about the core platform components:

» 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

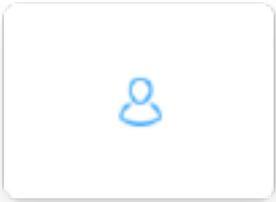
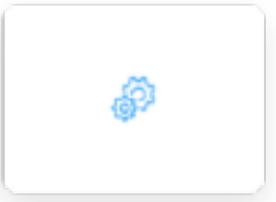
- represents incoming or outgoing messages from external parties - information, email, bank transfer
- Receive Message Event - incoming message occurring during the process flow, somewhere between start and end
- Send Message Event - outgoing message

| 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 **process 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 **business rule** 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**.

### 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"><li>defines a decision point</li></ul>            | <ul style="list-style-type: none"><li>no decision making;</li><li>all outgoing branches are activated</li></ul> |

### Closing gateway

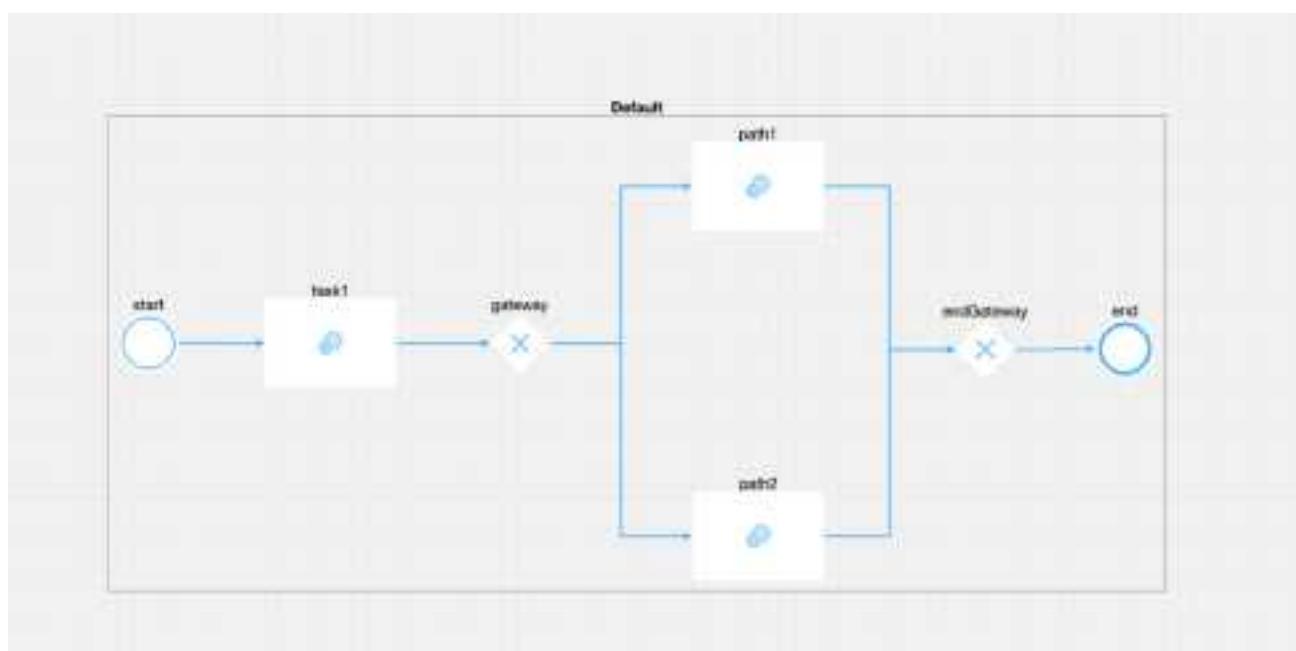
- closes gateways by connecting branches with no logic involved
- symbol used depends on the initial gateway
- parallel gateways - waits for all input tokens and merges all into one single token
- inclusive gateways
  - waits for all active inputs

- 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.

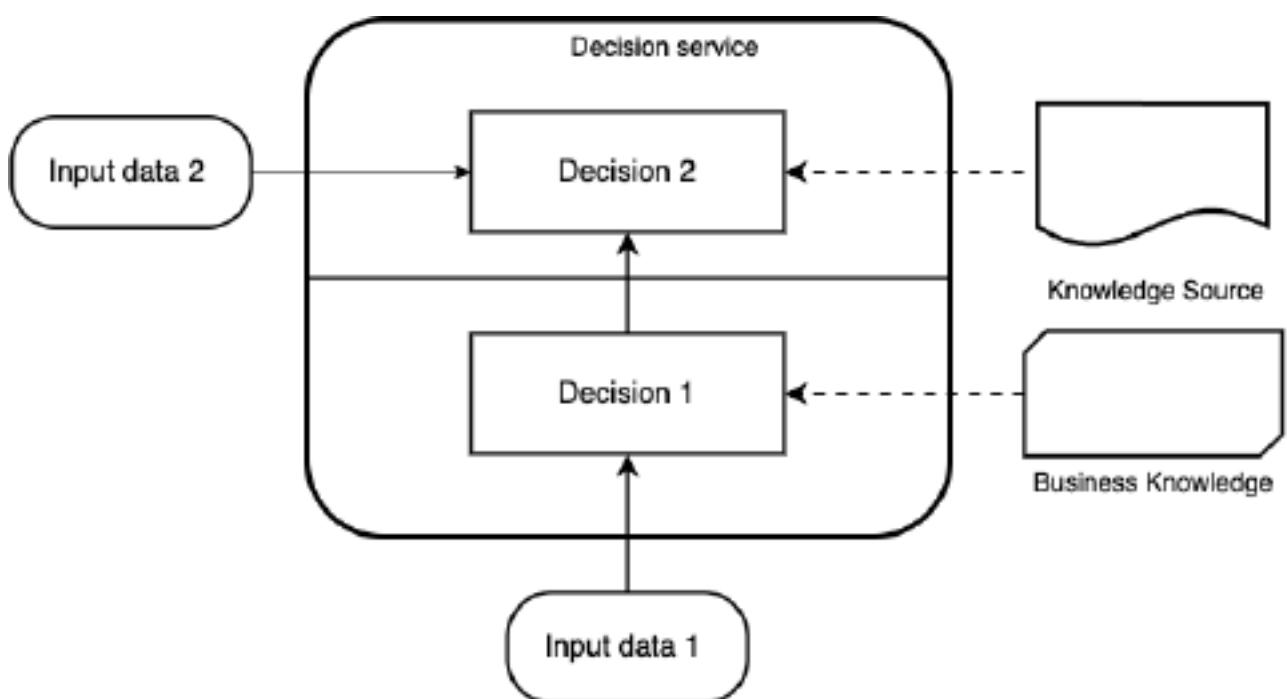
### 🔥 DANGER

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.   |
| 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 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 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)

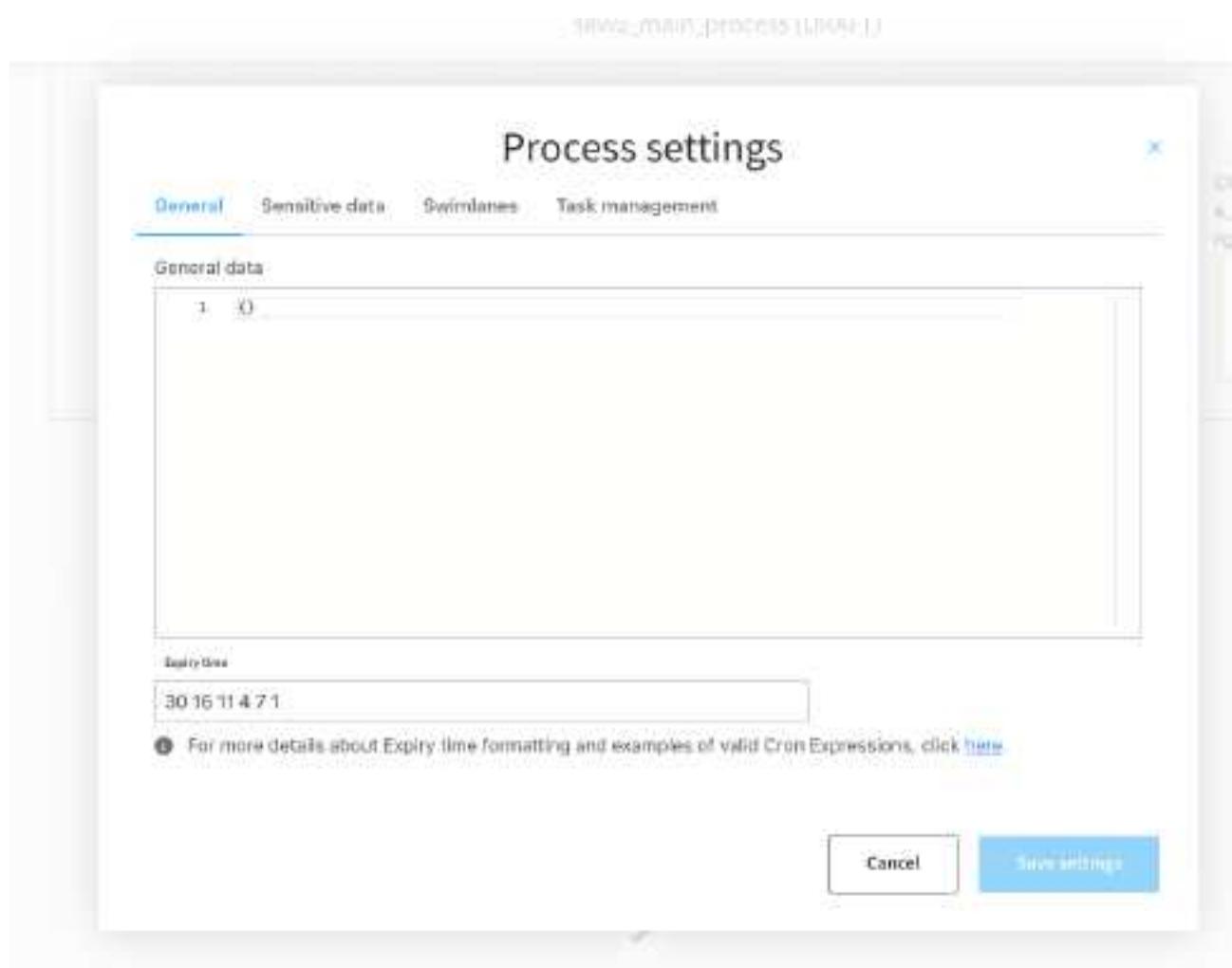
An example of a complete cron-expression is the string `0 0 12 ? * FRI` - which means **every Friday at 12:00:00 PM**.

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:



## 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                 |

| Format accepted      | Value ranges                                 |
|----------------------|--|
| 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

✖

Add stream

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



Was this page helpful?

## PLATFORM OVERVIEW / FLOWX.AI architecture

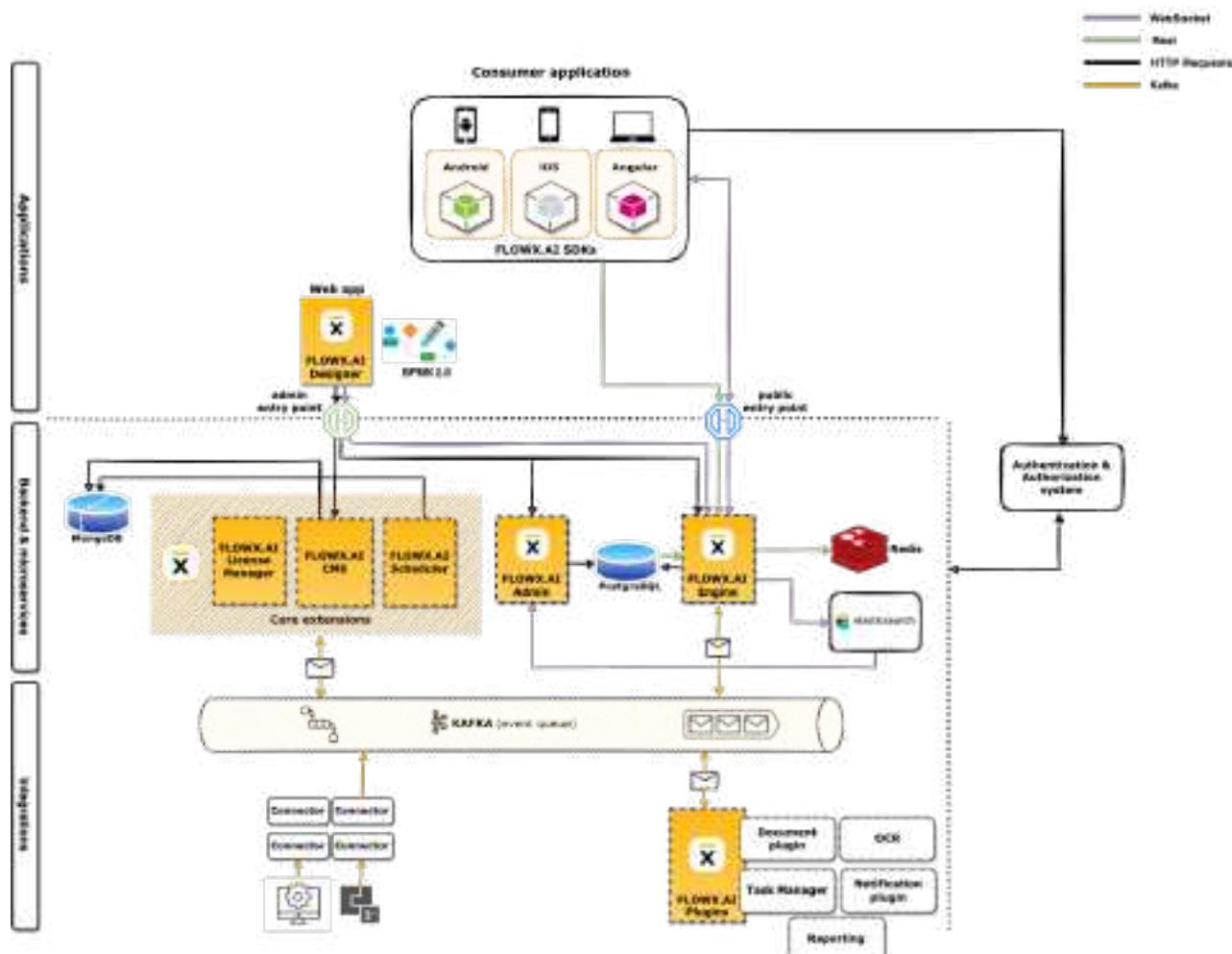
Let's go through the main components of the FLOWX.AI platform:

- **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](#)

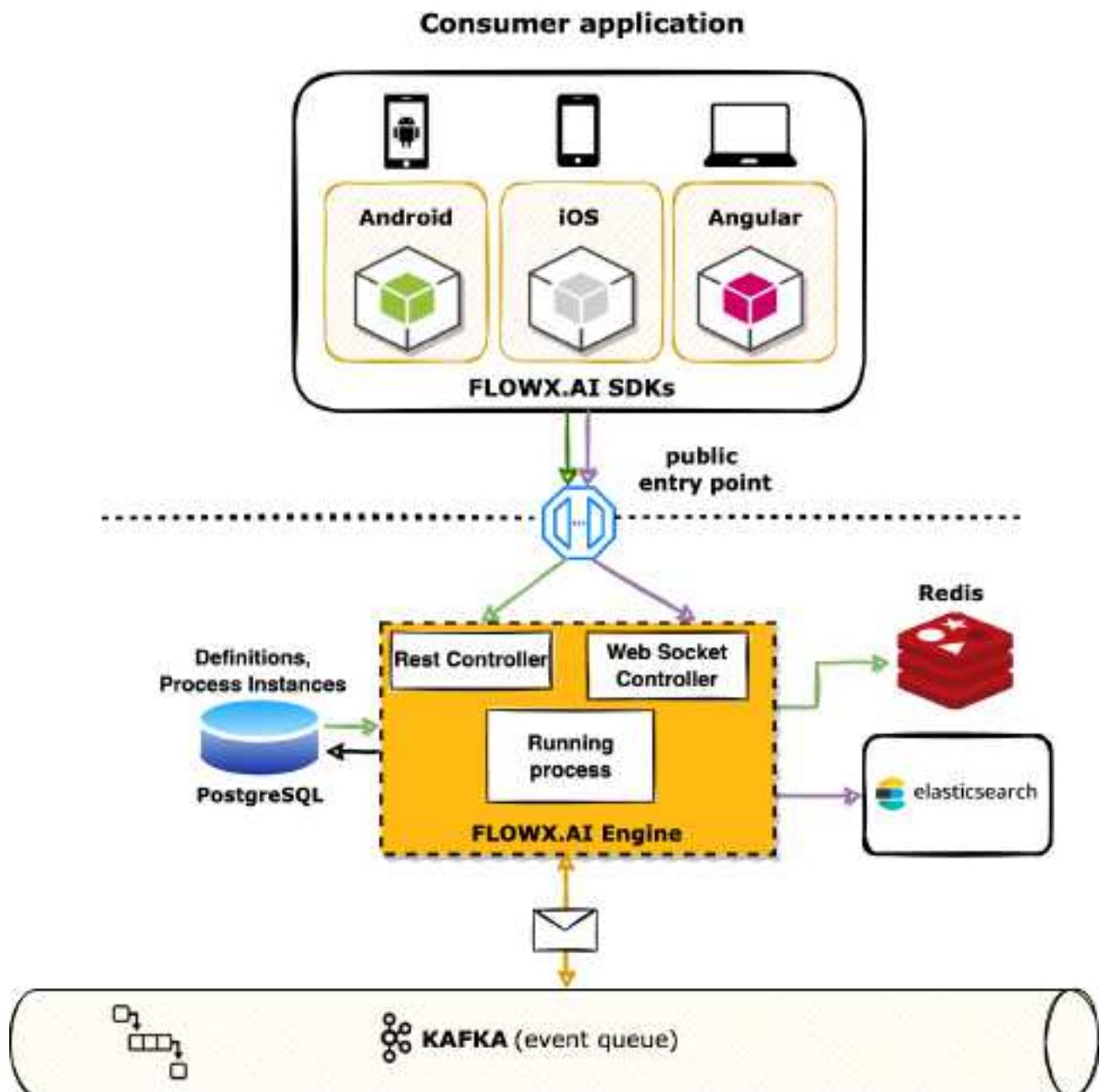
## Microservices

- **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.

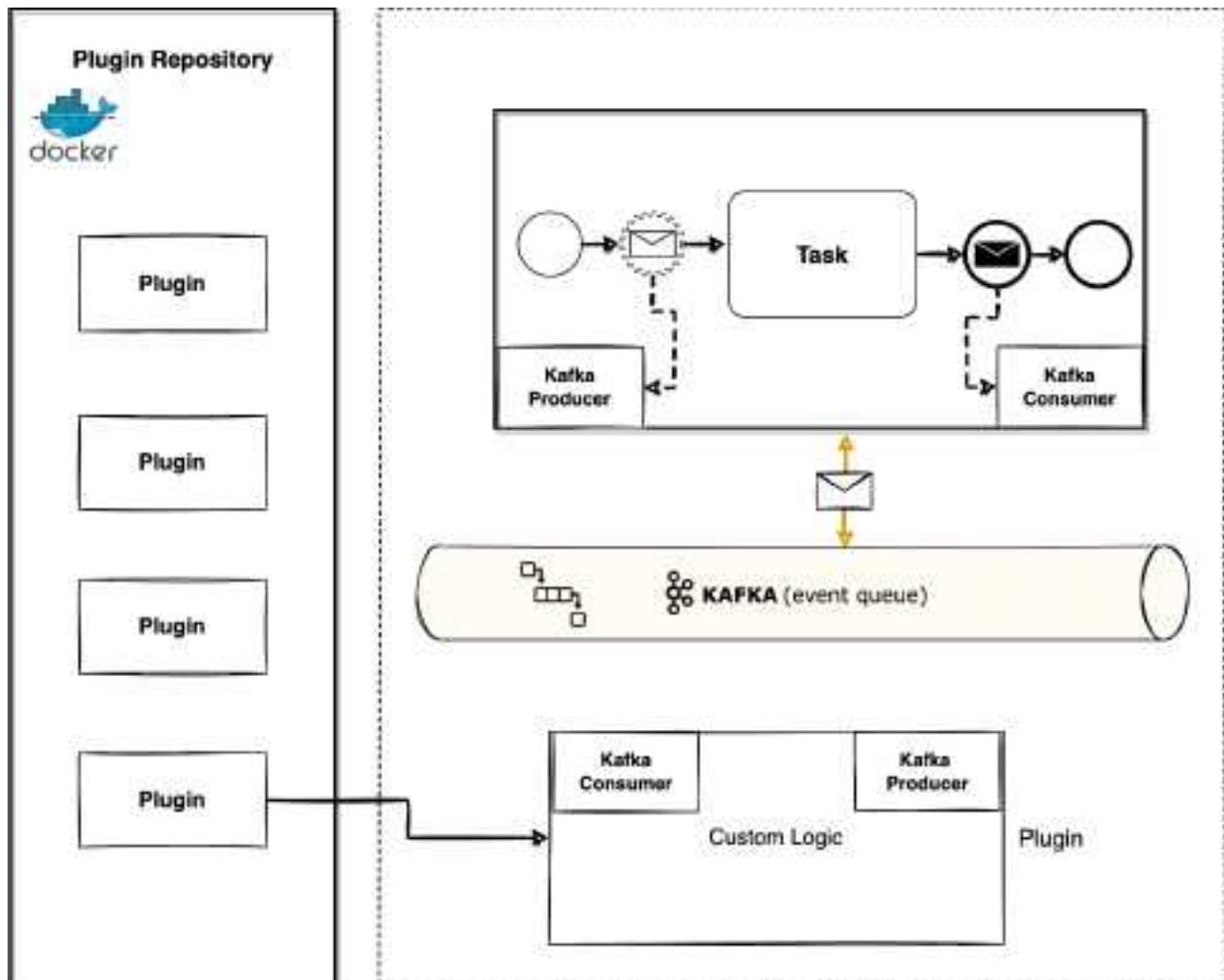
Every communication that comes from a consumer application, goes through a public entry point (API Gateway). To communicate with this component, the consumer application tries to start a process and the public entry point will check for authentication (Keycloak will send you a token) and the entry point 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 | Created at           | Edited by | Actions  |
|-------------------|---------|----------------------|-----------|--|
| Amazing Process   | 1.0     | 05 Oct 2022, 4:71 PM | John Doe  | <span>View</span> <span>Edit</span> <span>⋮</span> |
| Awesome Process   | 1.0     | 05 Oct 2022, 4:97 PM | John Doe  | <span>View</span> <span>Edit</span> <span>⋮</span> |
| Exquisite Process | 0.9     | 05 Oct 2022, 2:49 PM | Jane Doe  | <span>View</span> <span>Edit</span> <span>⋮</span> |

Published

| Name                 | Version | Published at          | Published by   | Actions  |
|----------------------|---------|-----------------------|----------------|--|
| Incredible Process   | 0.1     | 05 Oct 2022, 2:09 PM  | John Doe       | <span>View</span> <span>Edit</span> <span>⋮</span> |
| Breathtaking Process | 0.1     | 05 Oct 2022, 8:11 AM  | Bess Twishes   | <span>View</span> <span>Edit</span> <span>⋮</span> |
| Stunning Process     | 0.1     | 04 Oct 2022, 12:30 PM | Silvia Giroire | <span>View</span> <span>Edit</span> <span>⋮</span> |

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.

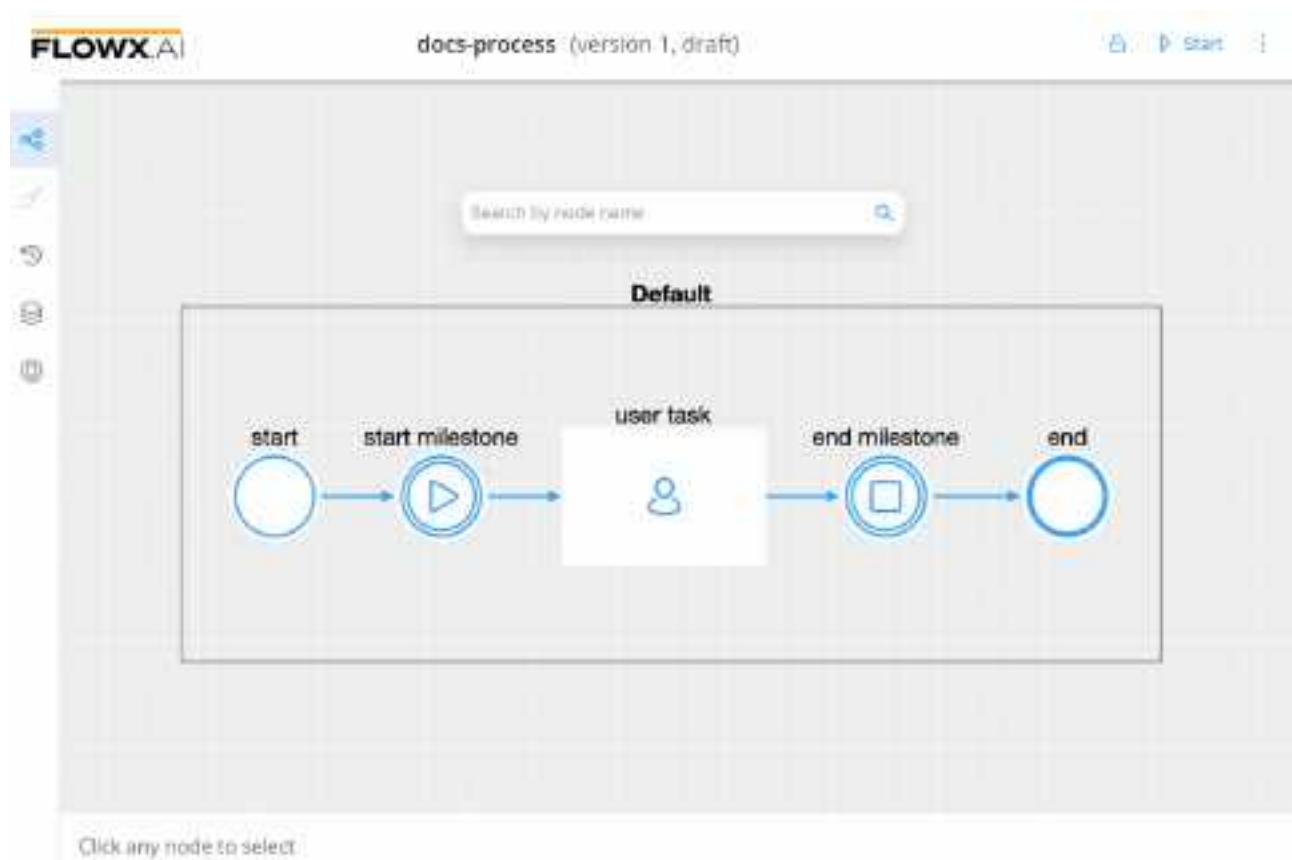
» [Process instance](#)

» [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.

| Name               | Type   | Used in reporting | Sensitive data | Description |
|--------------------|--------|-------------------|----------------|-------------|
| application        | OBJECT | +                 | ⚠️             | + ↗ 📁       |
| client             | OBJECT | +                 | ⚠️             | + ↗ 📁       |
| nameDeclarations   | OBJECT | +                 | ⚠️             | + ↗ 📁       |
| identificationData | OBJECT | +                 | ⚠️             | + ↗ 📁       |
| legalAddress       | OBJECT | +                 | ✖️             | + ↗ 📁       |
| residenceAddress   | OBJECT | +                 | ⚠️             | + ↗ 📁       |
| mailingAddress     | OBJECT | +                 | ⚠️             | + ↗ 📁       |
| lastName           | STRING | ✖️                | ⚠️             | ↗ 📁         |
| firstName          | STRING | ✖️                | ⚠️             | ↗ 📁         |

## Attributes type

Add attribute

Attribute name  Type  ▲

! Name is required

Description

Example value

Use in reporting

Sensitive Data

! 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 a data model configuration screen. At the top, it says "data\_model | Version 1 | draft". Below that is a table titled "data model" with columns: "name", "Type", "Used in reporting", "Sensitive data", and "Description". There are seven rows in the table, each with a small icon on the left. The "Sensitive data" column contains yellow circles with red dots, indicating sensitive data status. The "Description" column shows a sequence of three icons: a person, a lock, and a checkmark.

|   | name         | Type    | Used in reporting | Sensitive data | Description |
|---|--------------|---------|-------------------|----------------|-------------|
| 1 | username     | STRING  | ✓                 | ✓              | ✓, ⚡, ✅     |
| 2 | password     | STRING  | ✓                 | ✓              | ✓, ⚡, ✅     |
| 3 | dateOfBirth  | STRING  | ✓                 | ✓              | ✓, ⚡, ✅     |
| 4 | agreementKey | STRING  | ✓                 | ✓              | ✓, ⚡, ✅     |
| 5 | match        | BOOLEAN | ✓                 | ✓              | ✓, ⚡, ✅     |
| 6 | userObject   | OBJECT  | ✓                 | ✓              | ✓, ⚡, ✅     |

## 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.

×

## Add Attribute

Attribute name ⓘ

Type

Enumeration (Opt)

Example Value

Description (Opt)



Use in reporting



Sensitive data

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

## 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 a software interface for managing data models. At the top, a header bar displays the title "generate\_data\_model | version 1 (draft)" and various navigation icons. Below the header is a table titled "Data Model" with five columns: "Name", "Type", "Used in reporting", "Sensitive data", and "Description". A message "No attributes found" is centered in the table area. To the left of the main content is a vertical sidebar containing several icons: a back arrow, a forward arrow, a magnifying glass, a document, a gear, and a refresh symbol.

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.

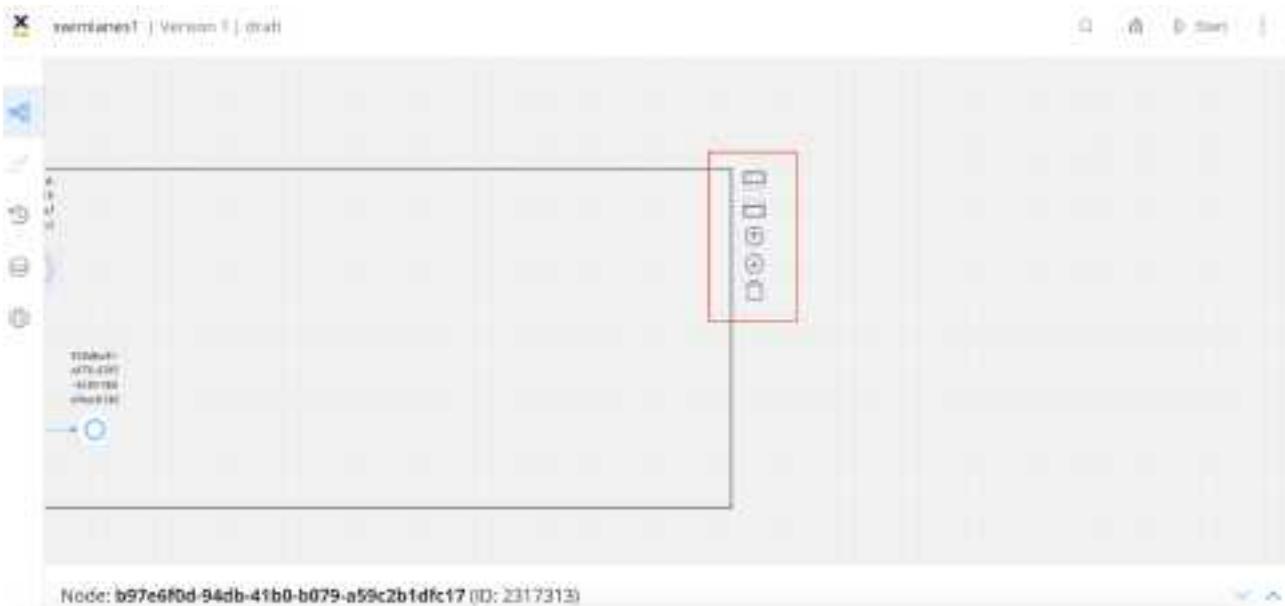
## Swimlanes

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:

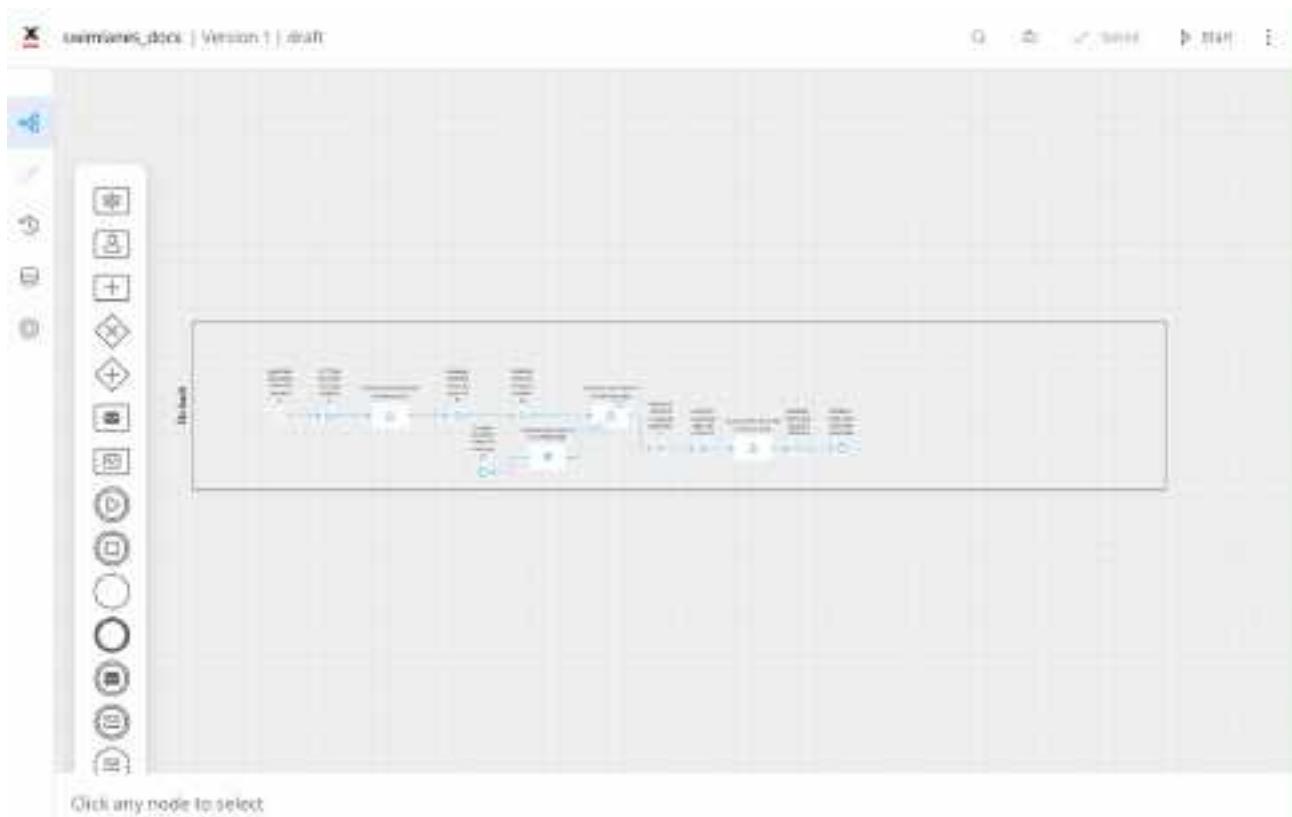
1. Access the **FLOWX.AI Designer**.
2. Open an existing process definition or create a new one.
3. Identify the default swimlane and select it to display the contextual menu.



### 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 ⋮

**General\*** Sensitive data Swimlanes Permissions Task management

Process definition name  
test-process

Use process in reporting

General data:

1

Expiry time  
ex: PT3M22S or 0 0 9-17 \* \* MON-FRI

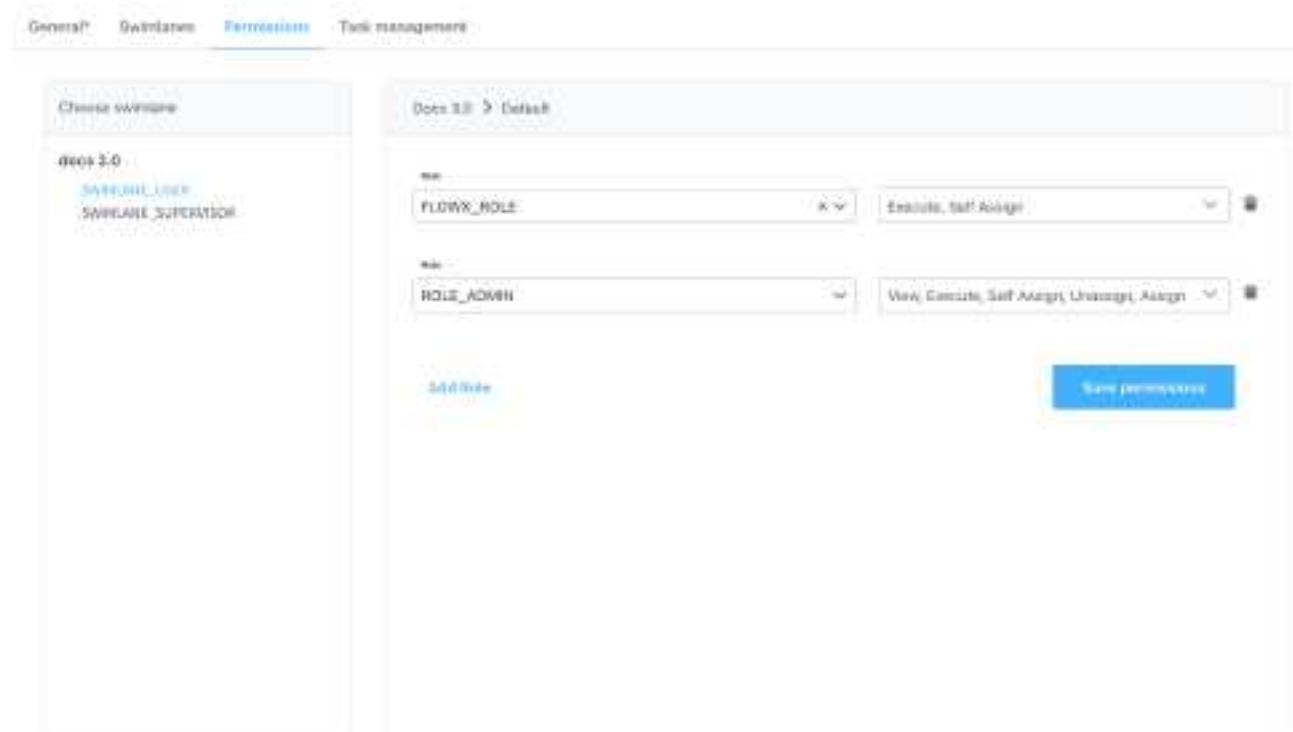
For more details about Expiry time formatting and examples of valid Cron Expressions, click [here](#).

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 of the FLOWX.AI interface. On the left, there's a sidebar with 'Create swimlane' and 'demos 3.0' sections. Under 'demos 3.0', there are two entries: 'SAFETY\_USER' and 'SWIMLANE\_SUPERVISION'. The main panel has a 'Role' section with two dropdown menus: 'ROLE\_FLOWX\_ROLE' and 'ROLE\_ADMIN'. Below these, there's a 'Permissions' section with a list of options: View, Execute, Self Assign, Unassign, Assign. At the bottom right of the main panel is a blue 'Save permissions' button.

» [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.

docs 3.0 | Version 1 | draft Start

General\*    Swimlanes    Permissions    **Task management**

**Application url**

ⓘ Generic parameter pointing to the application url where the flowx process is loaded.

**Search indexing**

ⓘ Process keys will be used to search data stored in the process.

[Add Key](#)

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

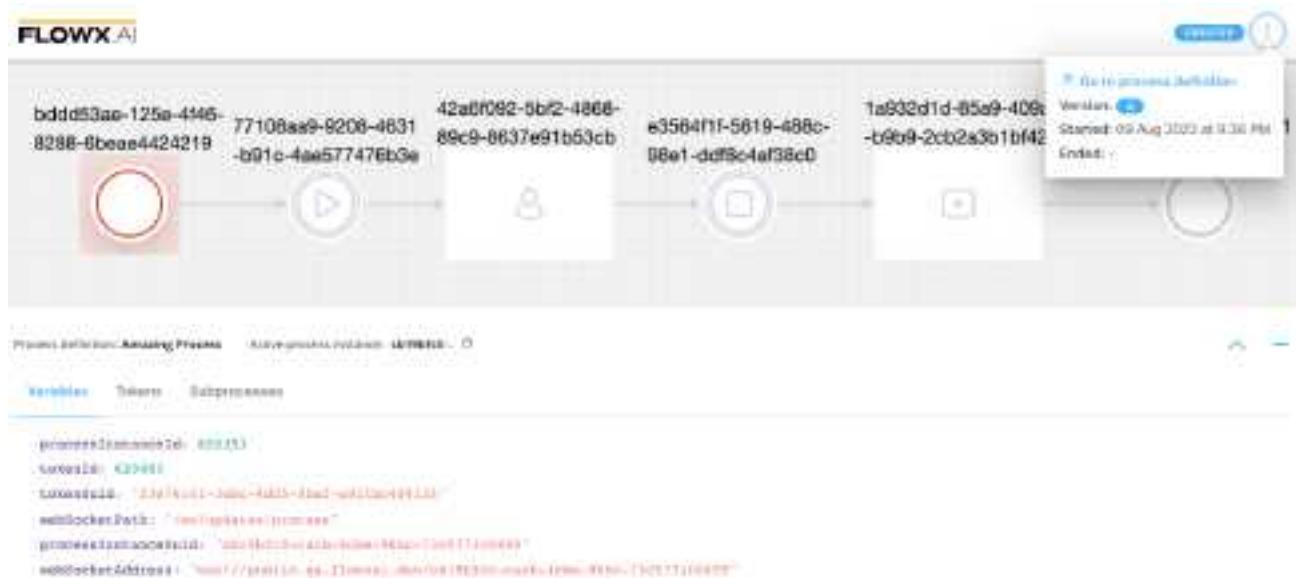
To check the status of a process or troubleshoot a failed process, follow these steps:

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

| Active processes                     |                    |             |                      |                                       |                      |
|--------------------------------------|--------------------|-------------|----------------------|---------------------------------------|----------------------|
| Workflow ID                          | Definition Name    | Description | Status               | Current State Name                    | Timestamp            |
| c118bfcb-cwcb-4cbe-9ab4-71c5771ca639 | Awesome Process    | -           | <span>CREATED</span> | boldd53ee-125e-4446-8288-6ce6e4424219 | 09 Aug 2022, 9:36 PM |
| a1c17f2b-f3fb-4fe1-wf24-13369ade550  | Super Process      | -           | <span>PENDING</span> | node_name                             | 09 Aug 2022, 8:35 PM |
| 3ed049fb-85ac-4833-94d2-6f14637e8496 | Awesome Process    | -           | <span>PENDING</span> | user_task                             | 09 Aug 2022, 8:32 PM |
| 8389430b-370a-4dc5-9a88-6605bbe87720 | Incredible Process | -           | <span>PENDING</span> | user_task                             | 09 Aug 2022, 8:30 PM |
| aa0ff73c-8ad8-46c6-9292-877f0bd7cf63 | Nice Process       | -           | <span>EXPIRED</span> | page_1                                | 09 Aug 2022, 7:47 PM |

# Understanding the Process Status Data

The process status data includes the following:



- **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](#) 

```
processInstanceId: 619702
tokenId: 619752
tokenUuid: "1d7aa0c4-d961-4154-8af6-3422e1d34fb6"
webSocketPath: "/ws/updates/process"
processInstanceUuid: "12156103-b40d-4cee-b0e0-e296371ccedbf"
webSocketAddress: "ws://public.qa.flowxai.dev/12156183-b40d-4cee-b0e0-e296371ccedbf"
```

- **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](#) 

| Token uuid                           | Token status  | Status Current Node | Date updated         |   |
|--------------------------------------|---|---------------------|----------------------|---|
| 1d7aa0c4-d961-4154-8af6-3422e1d34fb6 |  | 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: [Analyze Process](#)    ACTIVE PROCESS INSTANCE: 12156185 [...](#)

| Source            | Message                                      | Type            | Timestamp             |
|-------------------|--|-----------------|-----------------------|
| NODE (smart Node) | Outgoing node for process instance not fo... | Node Definition | 06 Aug 2022 at 4:06 F |
| NODE (smart Node) | Outgoing node for process instance not fo... | Node Definition | 06 Aug 2022 at 4:06 F |
| NODE (smart Node) | Outgoing node for process instance not fo... | Node Definition | 06 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

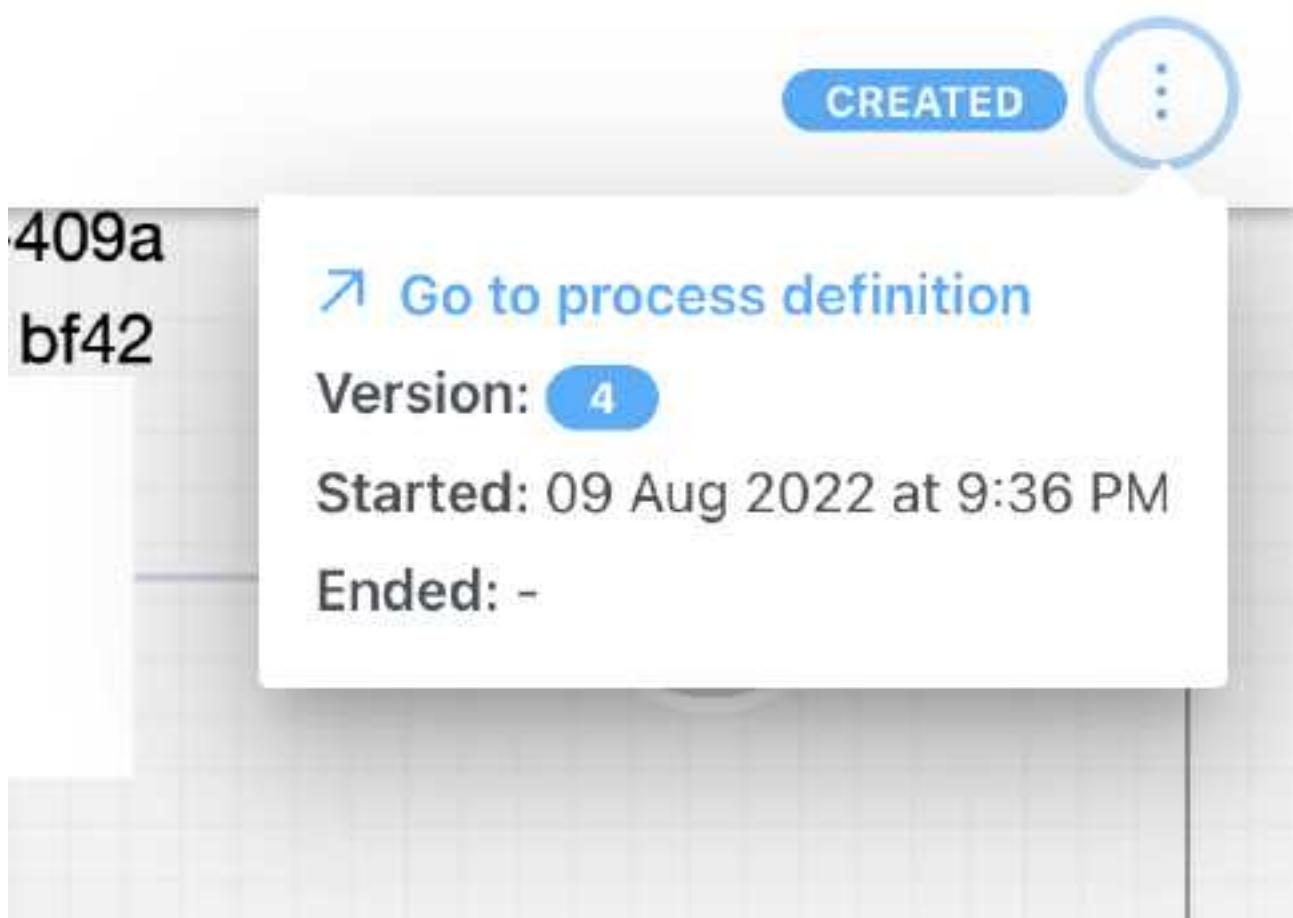
| Timestamp            | User             | Reason           | Event    | Subject Identifier               | Status  |
|----------------------|------------------|------------------|----------|----------------------------------|---------|
| 06 Oct 2022, 1:54 PM | xfrdzko@mail.com | Exception        | View     |                                  |         |
| 06 Oct 2022, 1:53 PM | system           | Exception        | ClearAll | P18001                           |         |
| 06 Oct 2022, 1:53 PM | xfrdzko@mail.com | Process instance | Start    | 60101795-0001-4061-80ad-00add174 | 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:

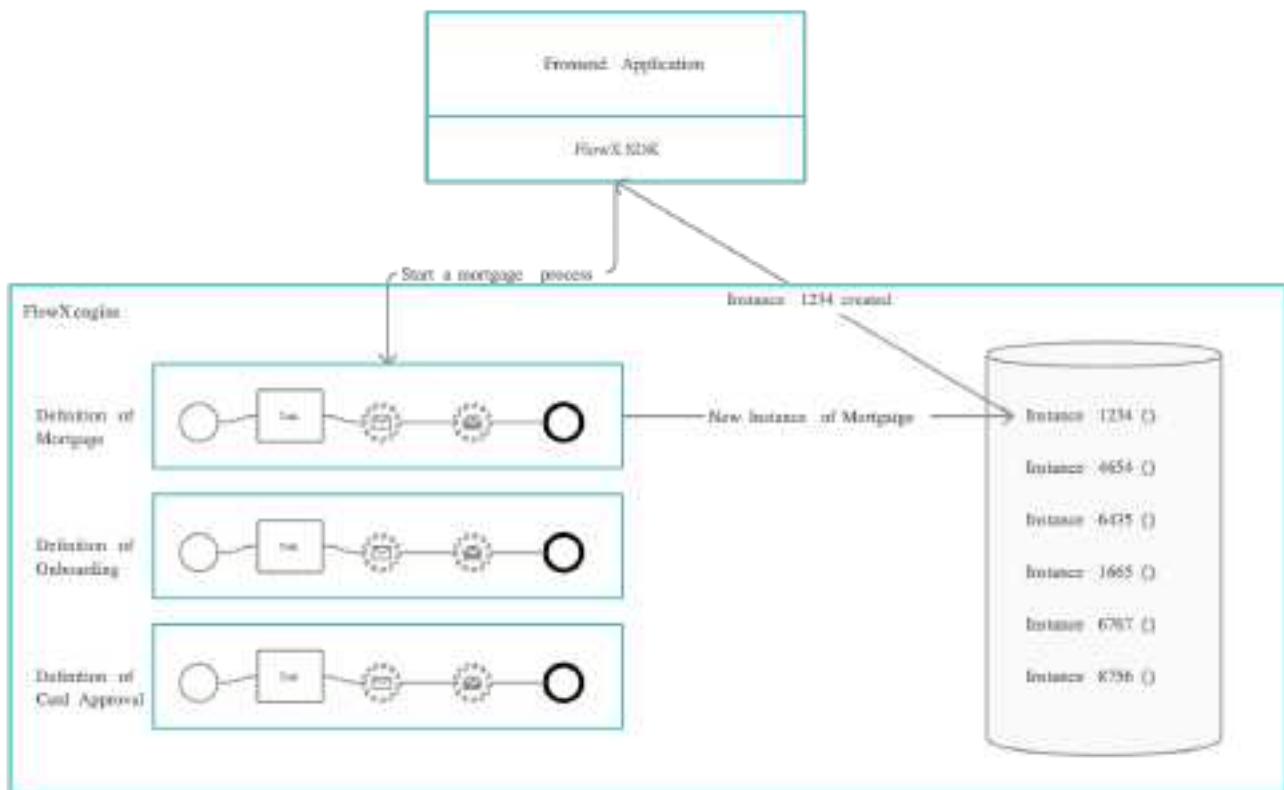
- **Green** - nodes highlighted with green mark the nodes passed by the token
- **Red** - the node highlighted with red marks the node where the token is stuck (process failure)

The screenshot shows the FLOWX.AI web application's process status interface. On the left, there is a sidebar with navigation links for Processes, Definitions, Active processes, Content Management, Plugins, Task Manager, and Drag & Drop Canvas. The main area is titled "Active processes" and contains a table with the following data:

| Process ID                           | Description name            | Status | Current Node Name | Timestamp            | Action |
|--------------------------------------|-----------------------------|--------|-------------------|----------------------|--------|
| e111f3ef-ac3b-4c66-8e50-#0268f8a676  | test_renderer_proc_instance | ENDED  | Step              | 22 Jul 2023, 4:11 PM | ...    |
| e148cf93-418e-4558-8e64-a7668685fed  | test_renderer_proc_instance | FAILED | Step              | 22 Jul 2023, 4:17 PM | ...    |
| 937a8fe-4945-#12b-ac18-a54e8e83c125  | test_renderer_proc_instance | ENDED  | End               | 22 Jul 2023, 4:14 PM | ...    |
| 9019d399-0199-#72a-#87e-a55c133e0a0f | test_renderer_proc_instance | ENDED  | Step              | 22 Jul 2023, 4:10 PM | ...    |
| 04482787-1008-#64b-#71a-1d1663f0fbc  | test_renderer_proc_instance | ENDED  | Step              | 22 Jul 2023, 4:09 PM | ...    |
| 4ffba4a2-#0a4-#70b-#0268f8a676       | test_renderer_proc_instance | ENDED  | Step              | 22 Jul 2023, 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.                         |
| <b>HTTP code 403 –<br/>Forbidden</b>                                    | The current user does not have the process access role for starting that process.                |
| <b>HTTP code 401 –<br/>Unauthorized</b>                                 | 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.

| Exceptions          |  |            |                         |
|---------------------|--|------------|-------------------------|
| Definition name     | Message                                    | Cause Type | Timestamp               |
| Amazing Process     | Start node for process definition not f... | -          | 10 Aug 2022 at 3:36 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.              | -          | 08 Aug 2022 at 2:59 PM  |
| Shocking Process    | Process definition not found.              | -          | 05 Aug 2022 at 3:48 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

| Source            | Message                                      | Type            | Timestamp             | View | Details |
|-------------------|--|-----------------|-----------------------|------|---------|
| NODE (smart Node) | Outgoing node for process instance not fo... | Node Definition | 09 Aug 2022 at 4:06 F |      |         |
| NODE (smart Node) | Outgoing node for process instance not fo... | Node Definition | 09 Aug 2022 at 4:06 F |      |         |
| NODE (smart 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.

## &lt; Exceptions: 619820

|                     |   |                        |                                      |
|---------------------|---|------------------------|--------------------------------------|
| Process Definition: | <a href="#">Awesome Process</a>               | 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:349), ai.flowx.engine.instance.service.
impl.NodeProcessorServiceimpl.process(NodeProcessorServiceimpl.java:135), ai.flowx.engine.instance.service.
impl.NodeProcessorServiceimpl$$FastClassBySpringCGLIB$$badf4a43.invoke($generated$1), org.springframework.
cglib.proxy.MethodProxy.invoke(MethodProxy.java:230), org.springframework.aop.framework.
CglibAopProxy$CglibMethodInvocation.invokeJoinpoint(CglibAopProxy.java:379), 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:80), ai.flowx.
commons.trace.aop.JaegerTraceAspect.around(JaegerTraceAspect.java:32), 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:586), 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<br>Rule<br>Evaluation | when executing action rules fails for any reason |
|                                |  |
|                                |  |
| Condition<br>Evaluation        | when executing action conditions                 |

| Type                  | Cause   |
|-----------------------|---|
| Engine                | <p>when the connection with the database fails</p> <p>when the connection with <b>Redis</b> 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<br>Evaluation | <p>when the token can't pass a gateway for any reason, possible causes:</p> <ul style="list-style-type: none"> <li>• missing sequence/node</li> <li>• failed node rule</li> </ul> |
| 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 node configuration interface. At the top, there's a toolbar with icons for creating, deleting, and managing nodes. Below the toolbar, a search bar is labeled "Search by node name". A list of nodes is displayed, with the first item being "Start Node Example".

Below the list, a detailed configuration panel is shown for the "Start Node Example" node. The panel has tabs for "Node Config" and "General Config". Under "General Config", the "Node name" field is set to "Start Node Example". There is also a "Cancel task?" switch, which is turned on (blue). Under "Variables", the "Default" value is selected. At the bottom right of the panel is a blue "Save" button.

## 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

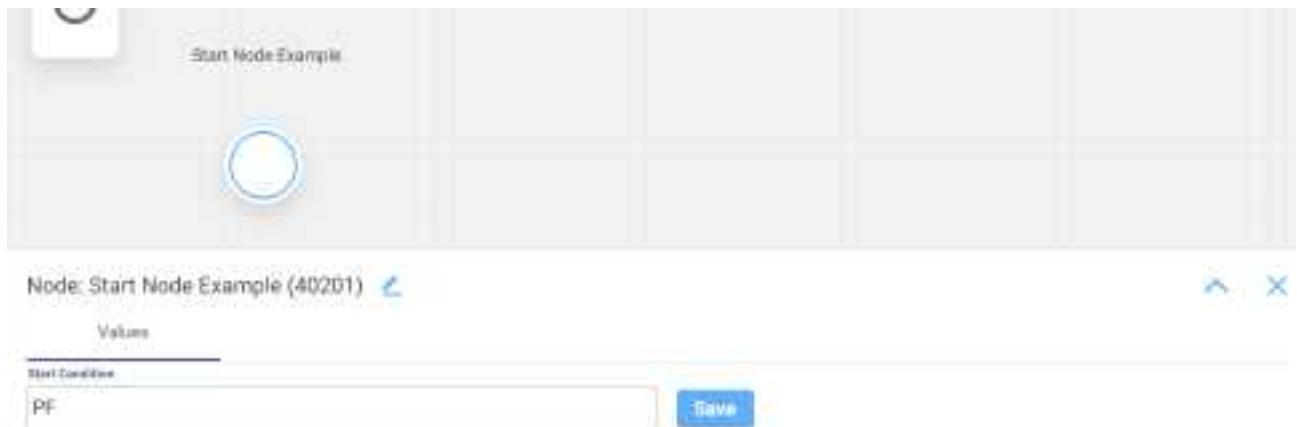
#### (!) INFO

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",  
}
```

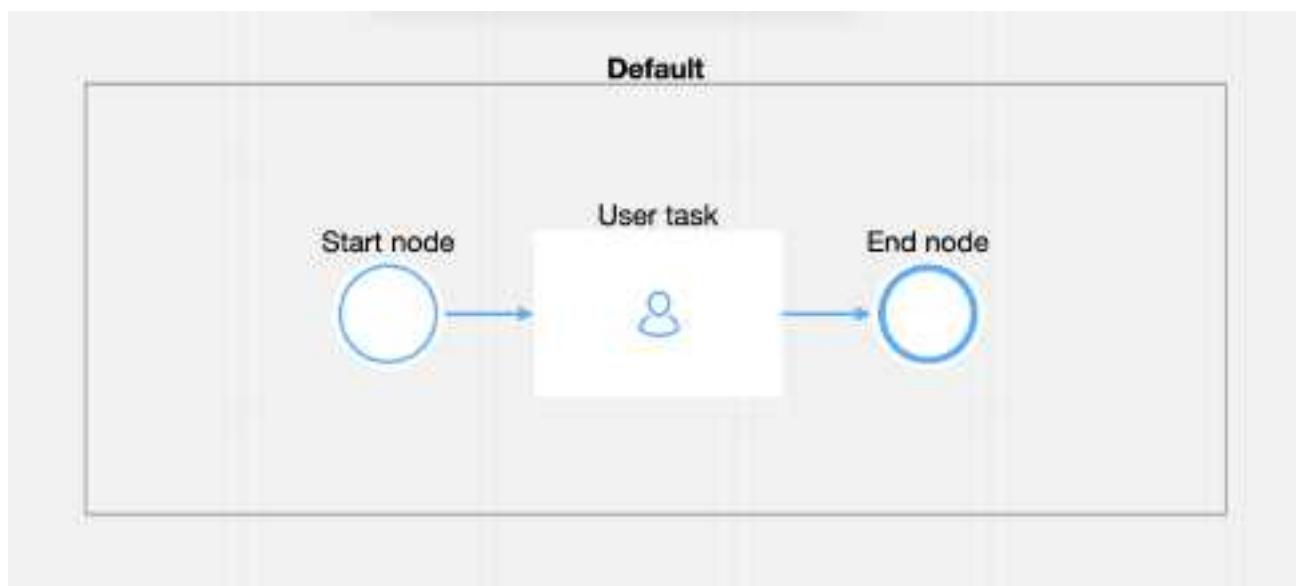
## End node



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)**

The screenshot shows the configuration interface for a 'Message send task' node. At the top, there are tabs for 'Node Config' (which is selected) and 'Actions'. Below this, under 'General Config', the 'Node name' is set to 'Message send task'. A toggle switch labeled 'Can go back?' is turned on. Under 'Swimlane', the value is 'Default'. Under 'Stage', the value is also 'Default'. The entire configuration area is enclosed in a light gray border.

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: 540653

Name:

f29bdc4-4e42-4e0d-9b66-953d67f272c4

Value:

1

Timer Expression:

Kafka Send Action

Automatic    Manual

Mandatory    Optional

Reputable

Automate 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**.



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

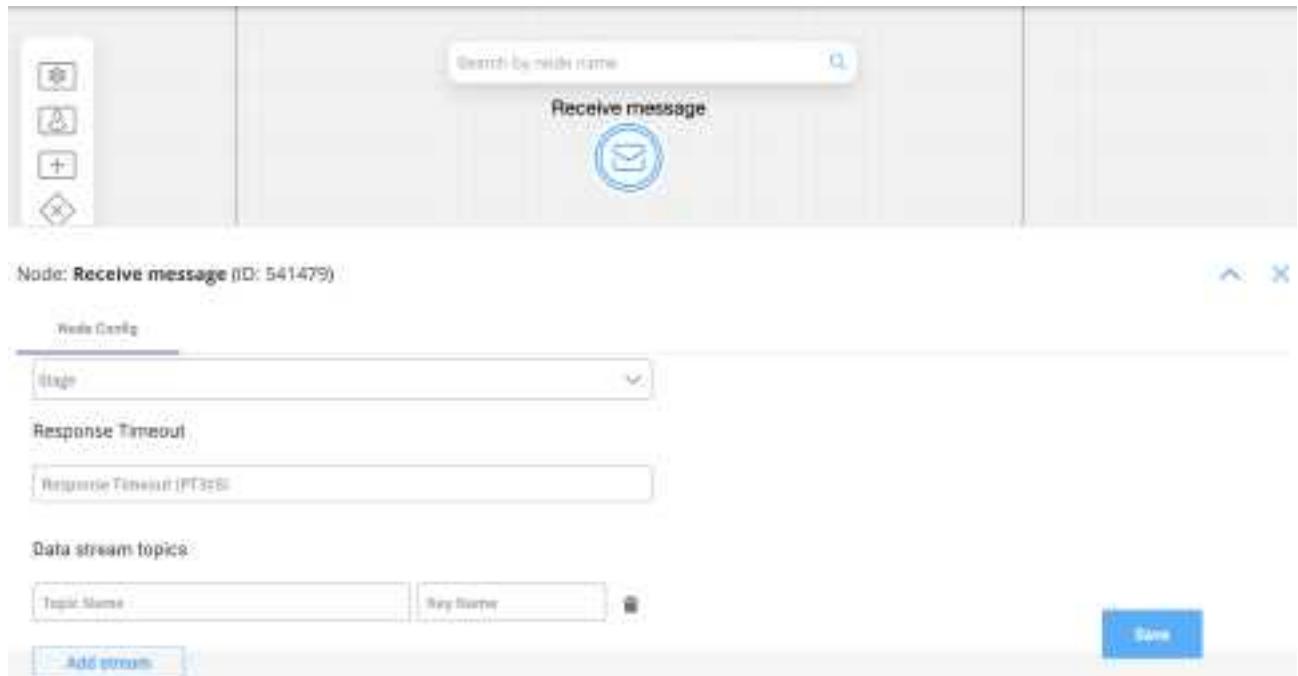
## Message receive task

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](#).

The screenshot shows the configuration of a 'Receive message' node. At the top, there is a toolbar with icons for 'Receive message', 'Edit', and 'Delete'. Below the toolbar, the node name 'Receive message' is displayed. The configuration section is titled 'Values' and contains two rows of fields. The first row includes 'Response Timeout (PT30S)', 'Topic name' (set to '@flowx.out.com.search.v1'), and 'Key name' (set to 'crmResponse'). The second row includes 'Topic name' (empty) and 'Key name' (empty). There are 'Save', 'Delete', and 'Add' buttons at the bottom right of the configuration area.

## 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

↑

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

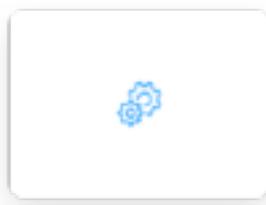
A task

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><b>General Config</b></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)

The screenshot shows a user interface for configuring a node. At the top, there are tabs for "Node Config" and "Actions". Below these, there are two dropdown menus: "Swimlane" (set to "Default") and "Stage" (set to an empty value). Under the "Response Timeout" section, there is a text input field containing "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?

ⓘ 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

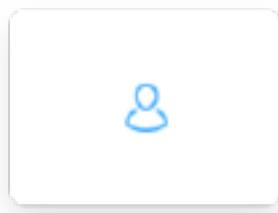
Was this page helpful?

# 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 |
|--|---------|
| <b>General Config</b><br>Node name<br><input type="text" value="User task"/><br>Can go back? <input checked="" type="checkbox"/> |         |
| <b>Flow Names</b><br><input type="text" value="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)

The screenshot shows two sections of a configuration interface. The top section, 'Node Config', has tabs for 'Node Config' (selected) and 'Actions'. Below this is a 'Data stream topics' section with fields for 'Topic Name' and 'Key Name', each with a trash icon. A blue button labeled 'Add stream' is also present. The bottom section, 'Task Management', contains a toggle switch for 'Update task management?' and a descriptive text: '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 platform's process management interface. On the left, there is a sidebar with a tree view of system components:

- Processes:
  - Definitions (selected)
  - Active processes
- Content Management
  - Item templates
  - Submission tags
  - Content models
  - Languages
  - Source systems
- Plugins
  - Task Manager
  - All tasks
  - Helpers
  - Stages
  - Notification templates

The main area is titled "Process Definitions" and contains two tables:

**Drafts / In progress**

| Name           | Version | Edited at            | Edited by | Actions |
|----------------|---------|----------------------|-----------|---------|
| Test Process   | 1.0     | 08-Jun-2022, 4:53 PM | John Doe  |         |
| Test Process 2 | 1.0     | 09-Jun-2022, 9:03 AM | Jane Doe  |         |

**Published**

| Name            | Version | Published at          | Published by | Actions |
|-----------------|---------|-----------------------|--------------|---------|
| Happy Process   | 1.0     | 26-May-2022, 12:09 PM | John Doe     |         |
| Stopper Process | 1.0     | 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?](#)

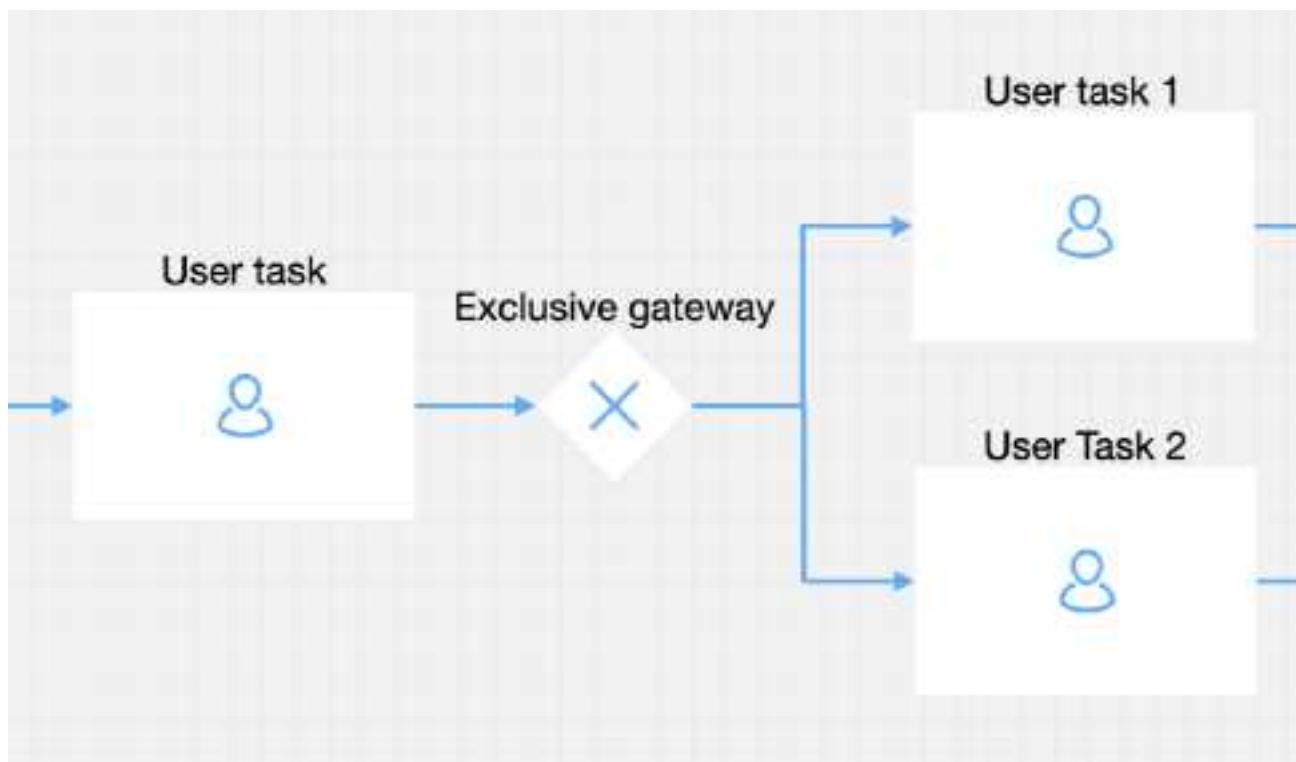
# BUILDING BLOCKS / Node / Exclusive gateway

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



## 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)

Main Config    Gateway Details

Language: English

Add

if: Input[get("application.client.page") == 10] then go to: user task 1

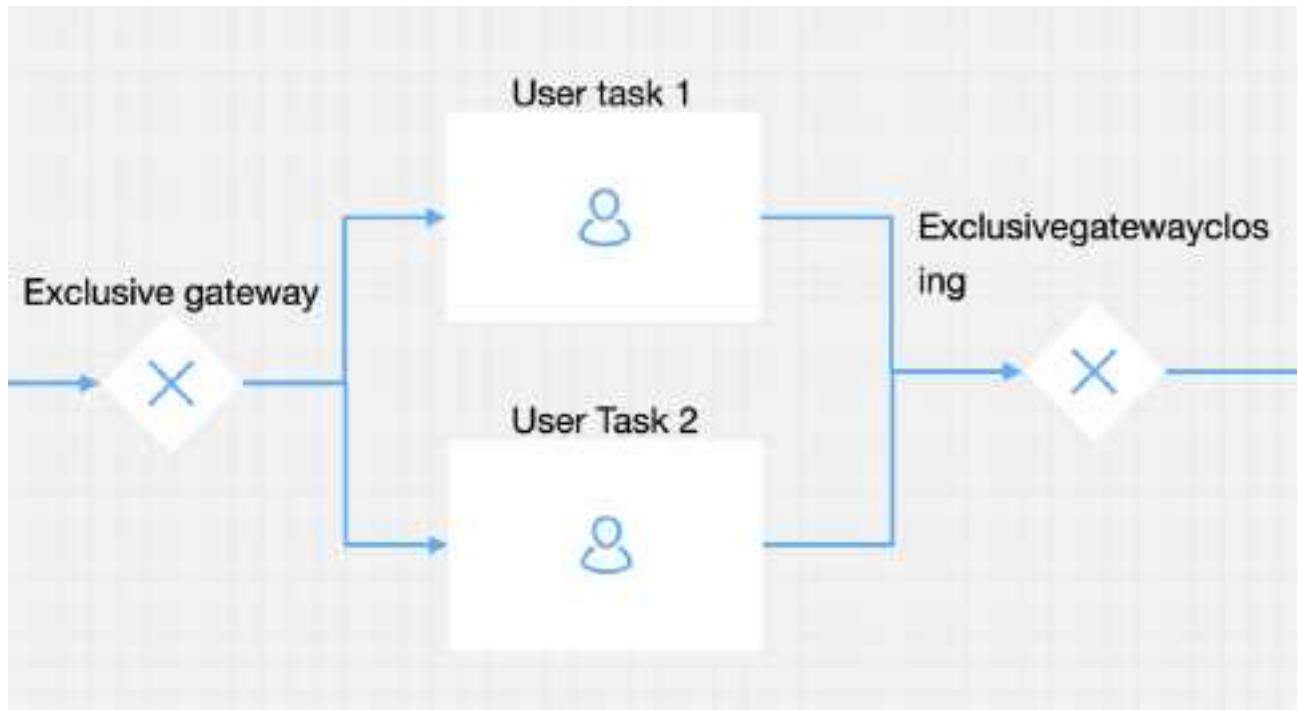
else if: Input[get("application.client.page") == 10] then go to: user task 1

else if: true then go to:

And

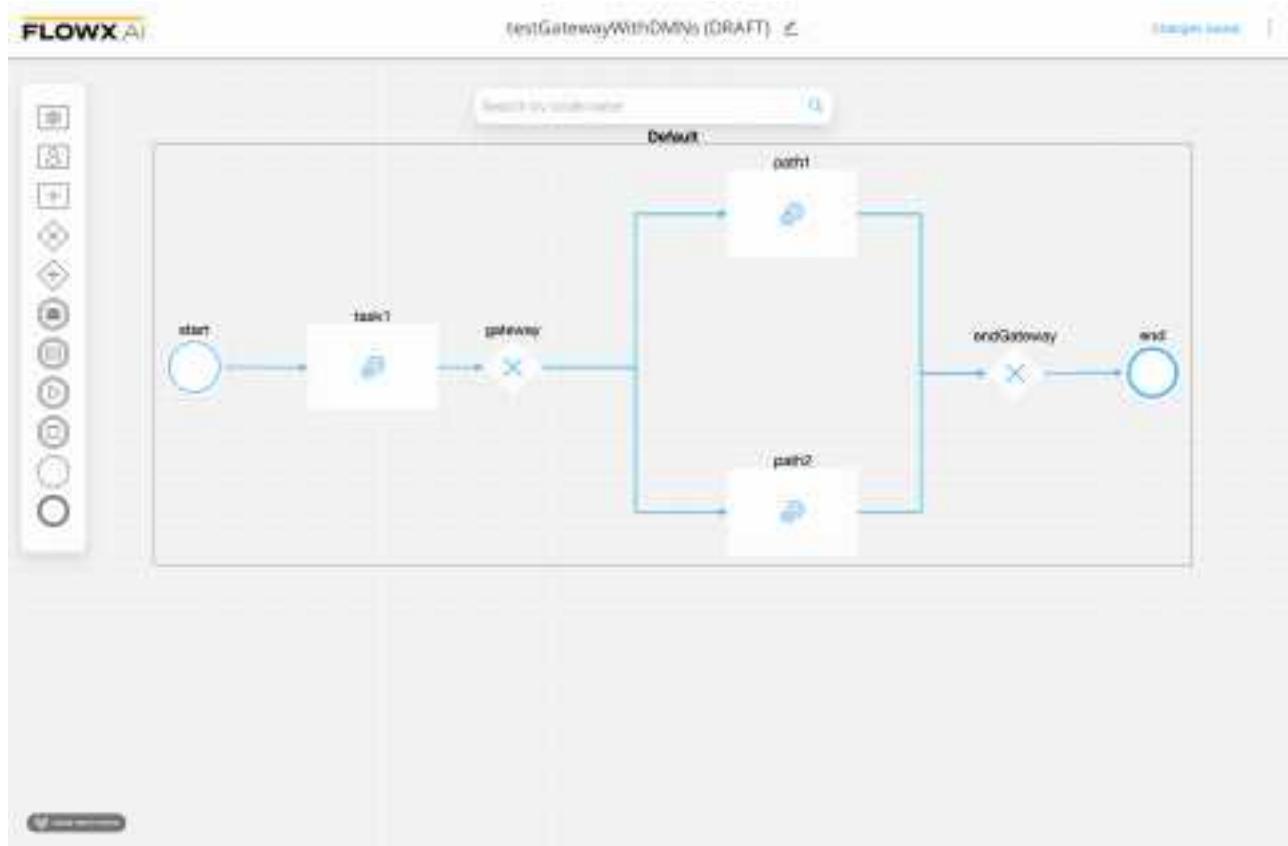
Save

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)

|   | When    | Then           | Annotations |
|---|---------|----------------|-------------|
|   | Input   | Next Node Name |             |
|   | 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.

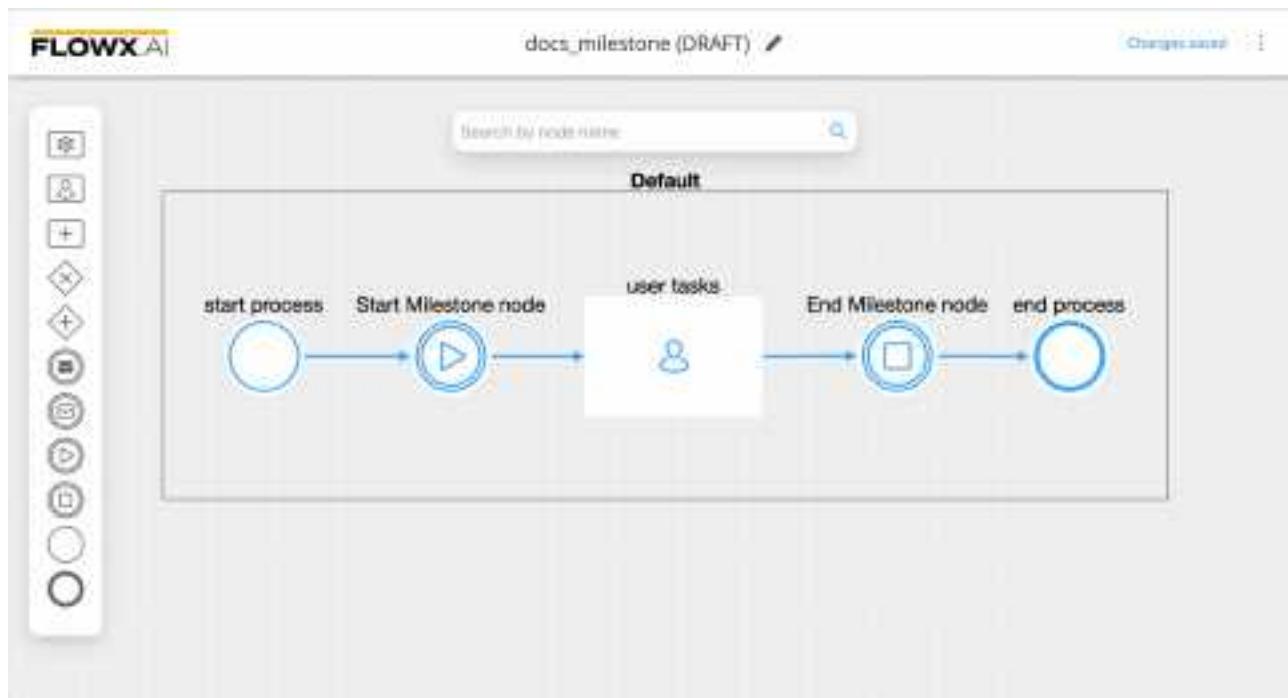
To configure a user task to be displayed in a Modal:

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.

7. 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

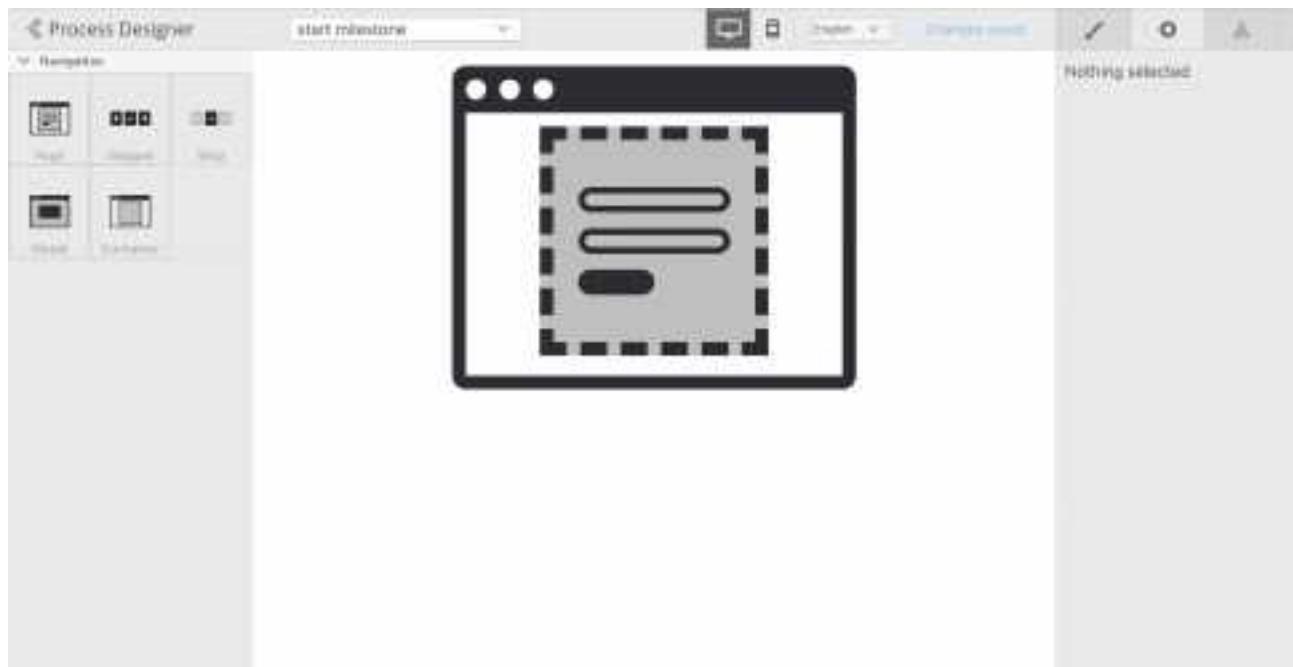
### Modal

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).



## 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**.

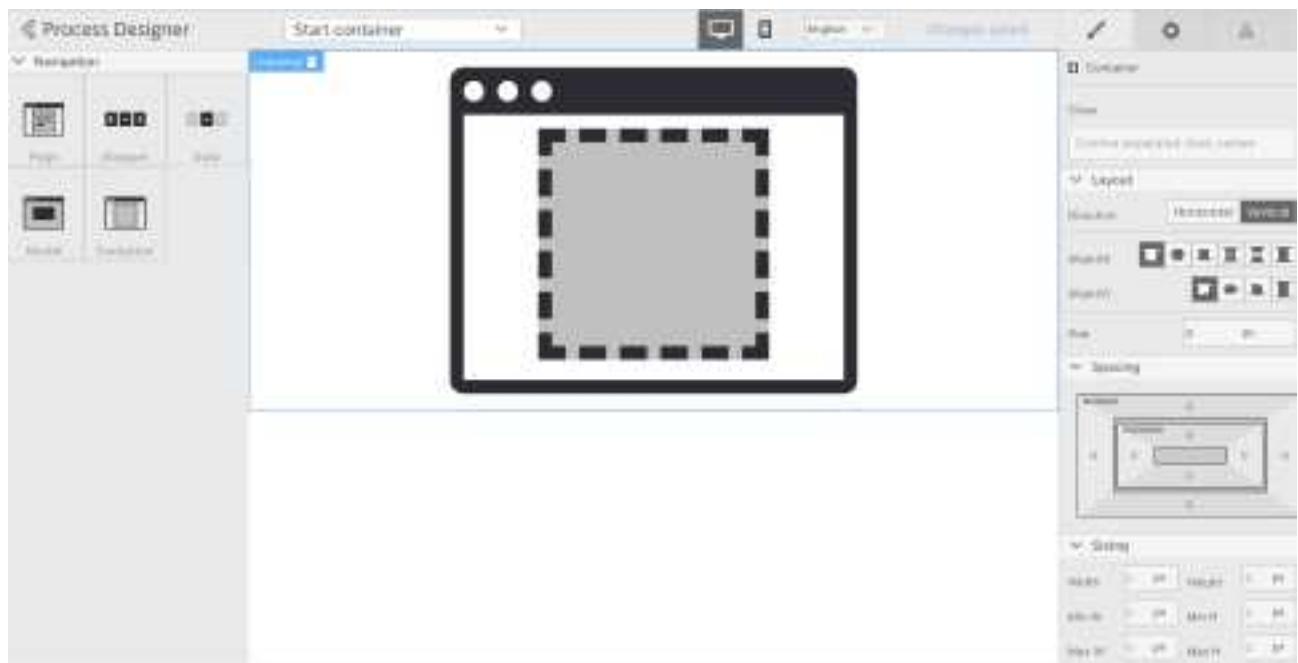


For more information about how to create a process with a Stepper + Steps structure, and how to configure the UI, check the following section:

» 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 ▼

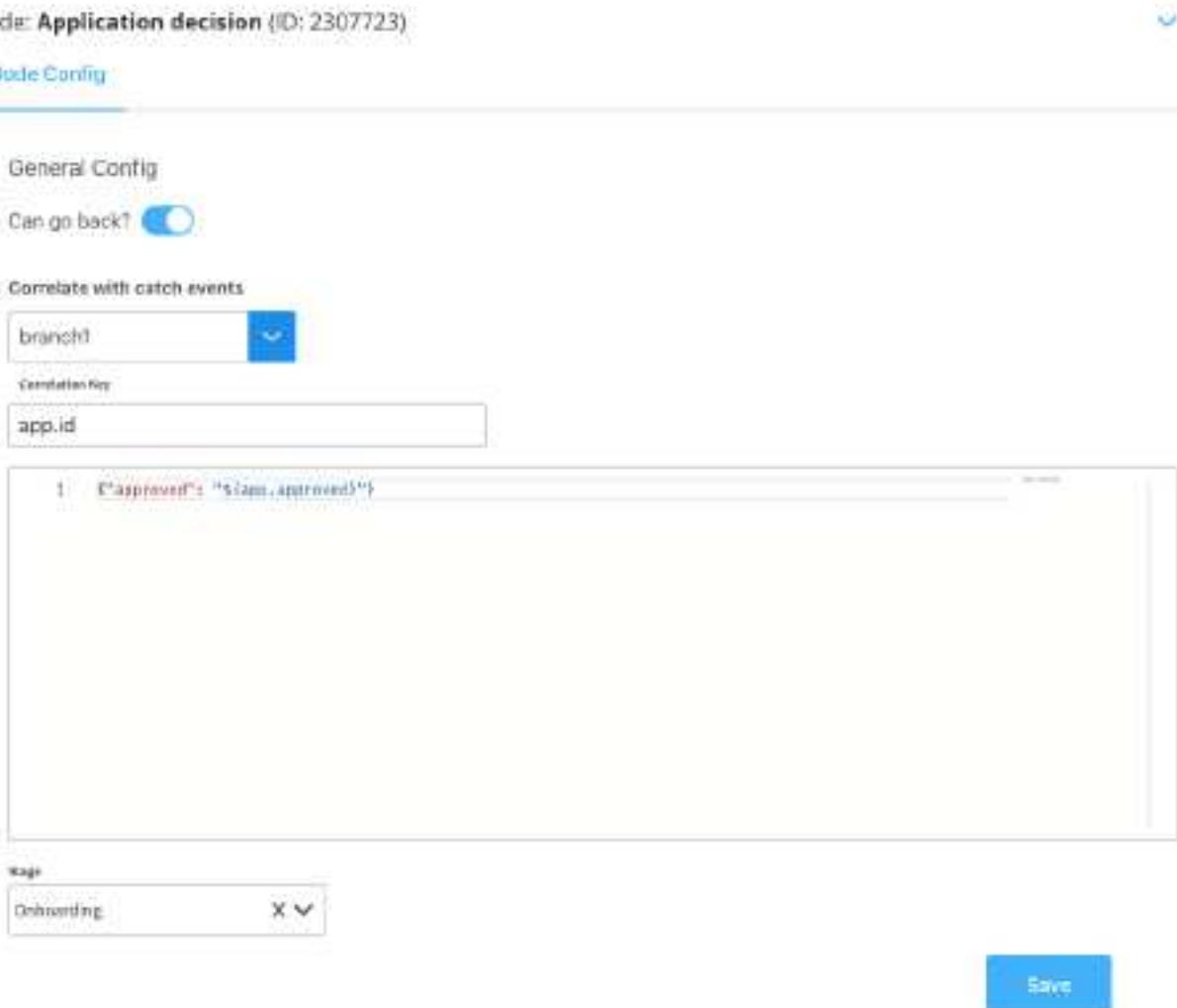
Correlation Key

app.id

1 E{approved": "item approved"}

Tags X ▾

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

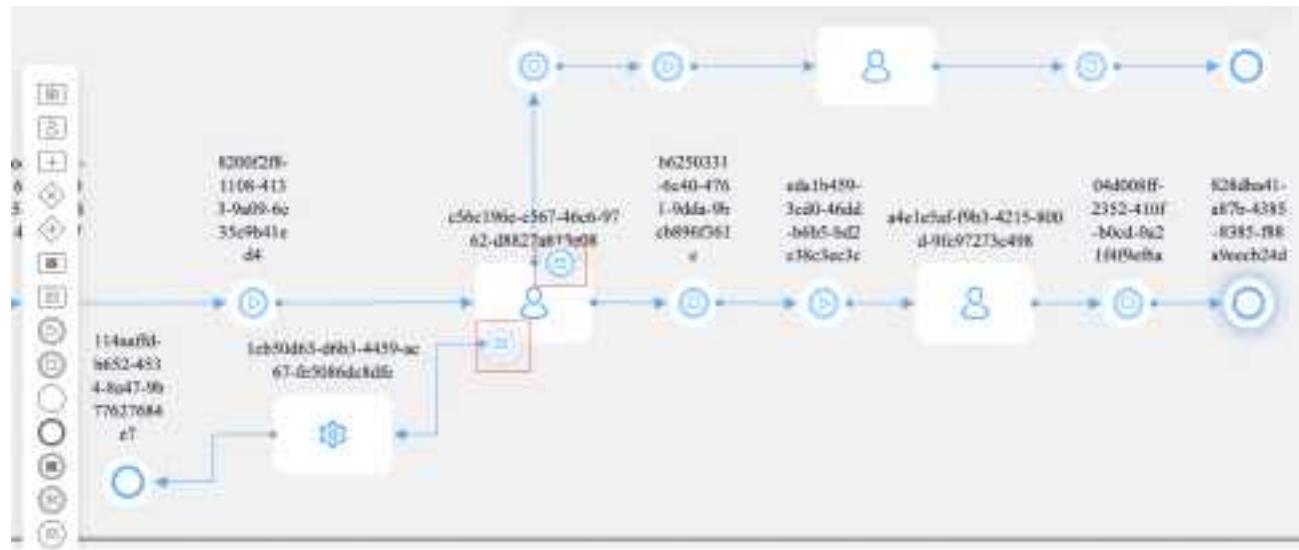
**Why it is important?** It allows the process to listen for and capture specific messages during the execution of the associated user task.

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

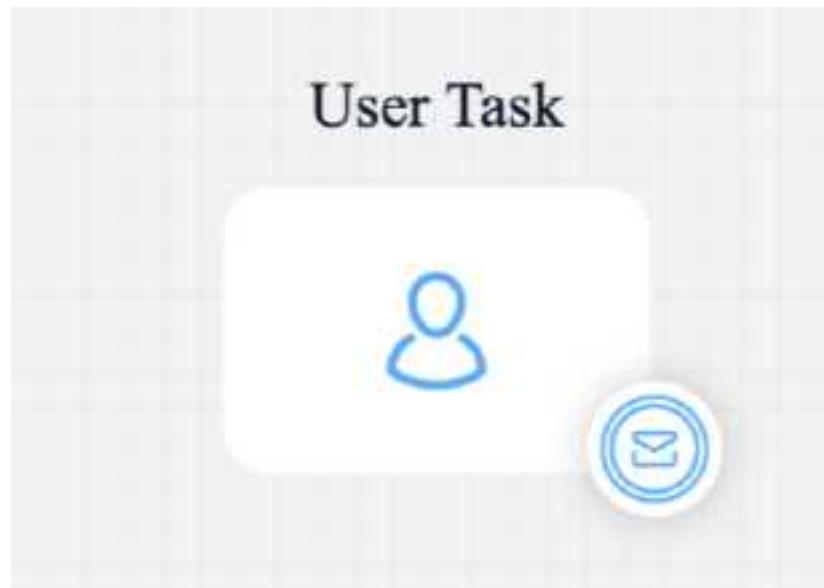
. You can have multiple intermediate boundary events on the same user task, but only one can be activated at a time.



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.

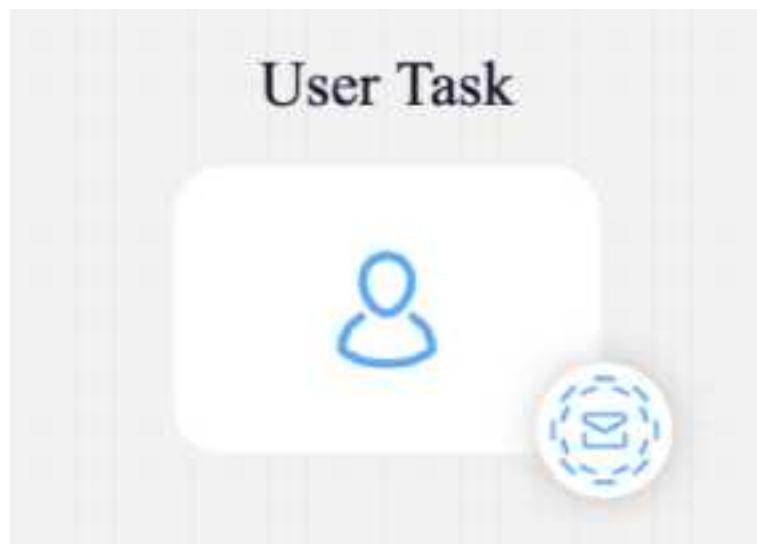
It can also be used as a standalone node, see more information on the following section:

» [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  | Alert Title                       | Alert Description  | Annotations |
| 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

Autonrun 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

Action

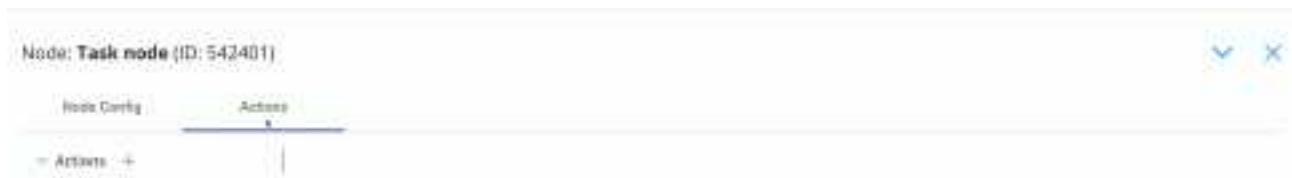
#### Message

```
1  {
2     "processName": "demoProcess",
3     "type": "START_PROCESS_INHERIT",
4     "clientDatakeys": ["webappKey"],
5     "params": {
6         "startCondition": "$listStartCondition",
7         "paramInCopy": []
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**.



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 

```
i {"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

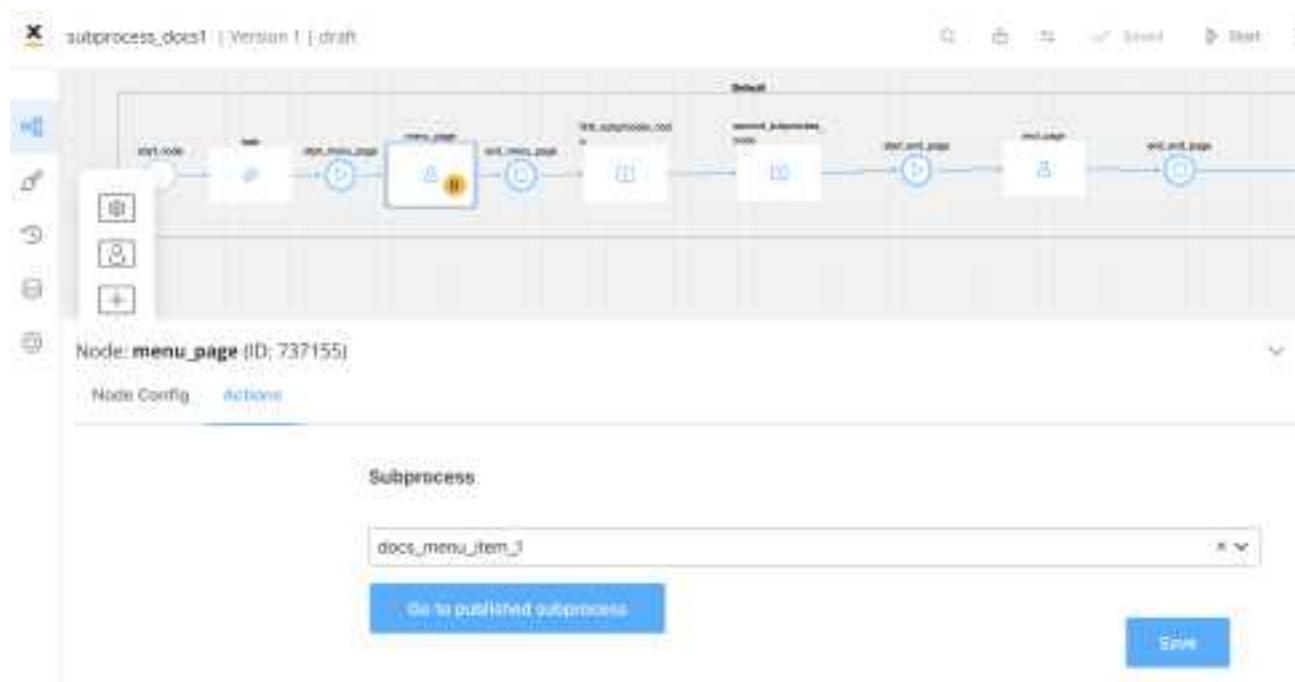
To use a process as a **subprocess**, you must first create it. Once the subprocess is created, you can start it from another (parent) process. To do this, you will need to add a **Start Subprocess** action to a **User task** node in the parent process or by using a **subprocess run node**.

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

selectedSubscription

uri

Add Params

Advanced configuration

Show Target Process

Save

The screenshot shows the configuration interface for the 'menu\_page' node. At the top, it displays the node name and ID. Below this, there are tabs for 'Node Config' and 'Actions', with 'Node Config' being active. The main area is titled 'Copy from current state' and contains three input fields: 'selectedCustomer', 'selectedSubscription', and 'uri'. Each field has a small edit icon to its right. Below these fields is a blue-bordered button labeled 'Add Params'. At the bottom of the configuration area, there is an 'Advanced configuration' section containing a toggle switch for 'Show Target Process' (which is turned on) and a large blue 'Save' button.

## 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 Key

Save

## Example

© FLOWX.AI

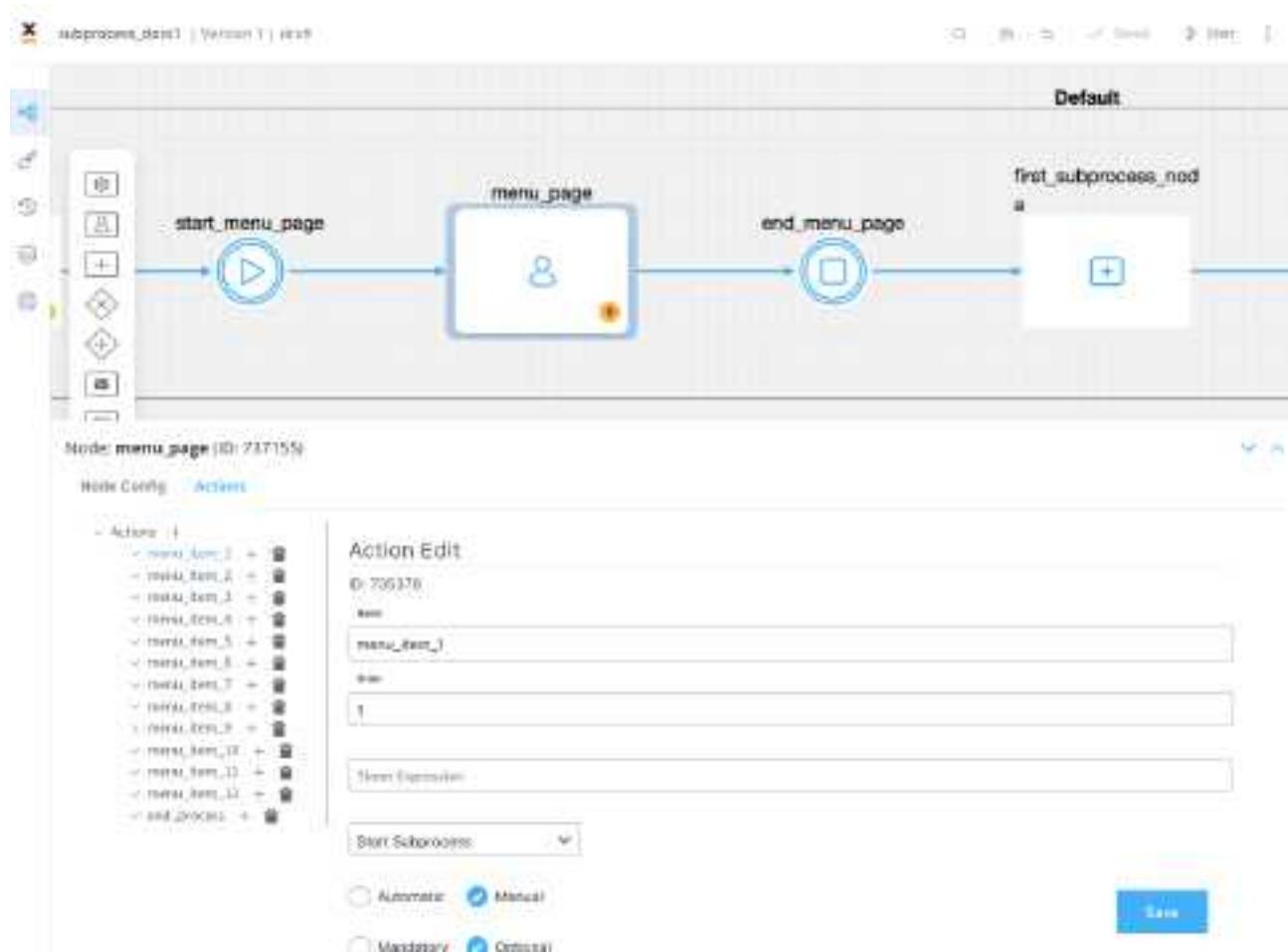
2023-07-26

Page 202 / 847

Let's create a main

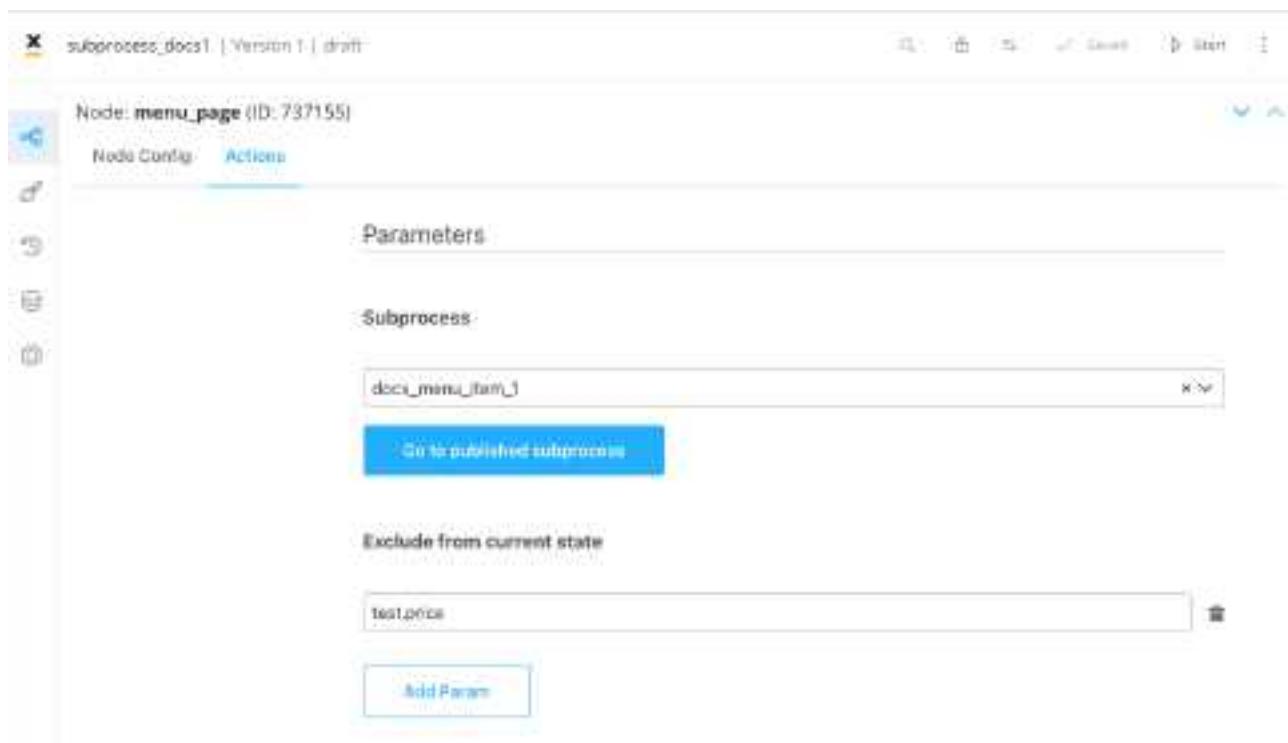
The fallback content to display on prerendering

, 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



- **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 a configuration interface for a subprocess named "menu\_page". The top bar indicates the file is saved. The main area has tabs for "Node Config" and "Actions", with "Actions" selected. A "Copy from current state" button is present. Below are several input fields for parameters:

- application (selected)
- application1
- application2
- application3
- application4
- application5
- webSocketAddress
- webSocketPath

An "Add Param" button is at the bottom.

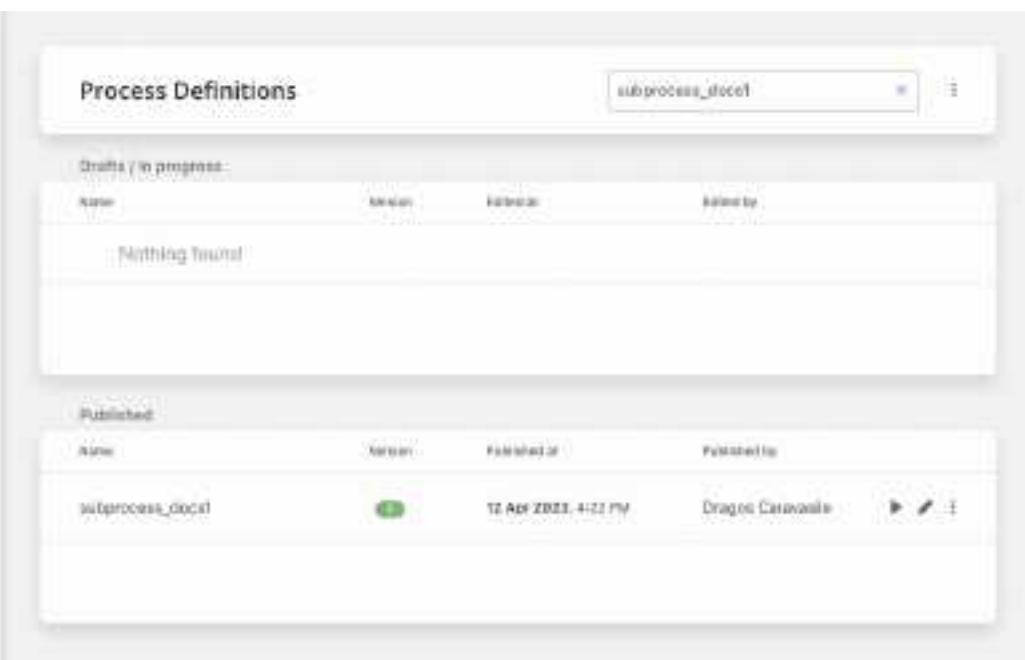
### ⚠ CAUTION

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 interface with the sidebar menu open. Under 'Processes', 'Definitions' is selected. The main area displays 'Process Definitions' with a search bar containing 'subprocess\_docet'. Below it, there are two sections: 'Drafts / In progress...' and 'Published'. The 'Published' section contains one item:

| Name             | Version | Published at        | Published by      |
|------------------|---------|---------------------|-------------------|
| subprocess_docet | 1.0.0   | TZ ADF 2023-4-22 PM | Dragos Caravastie |

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 Details    Actions

Actions: 0  
+ appendToParent

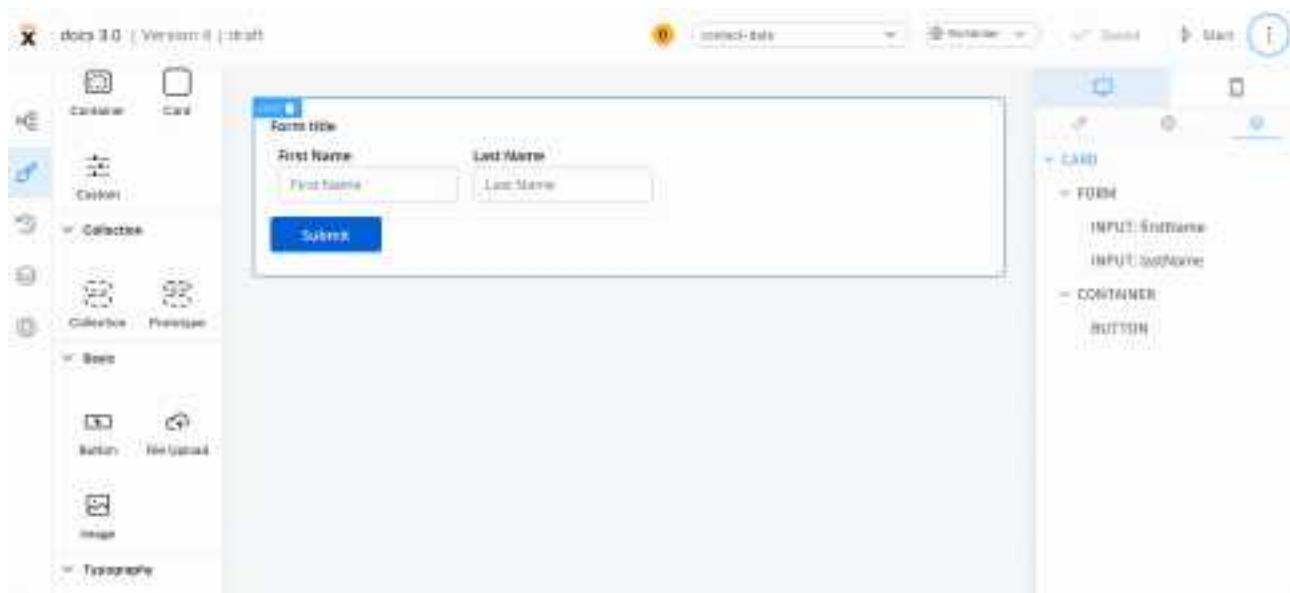
### Action Edit

ID: 546052  
Name:  
appendToParent  
Description:  
1  
Time Expression:  
Append Params to Parent Process  
 Automatic    Manual  
 Mandatory    Optional  
 Repeatable  
Automl Children?   
**Parameters**  
Copy from current state  
1. ~~FlowxTaskDataStart.bpmn~~  
 Replace Values:  
Destination in the parent state  
application  
 Replace Values  
Advanced configuration Save

Was this page helpful?

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

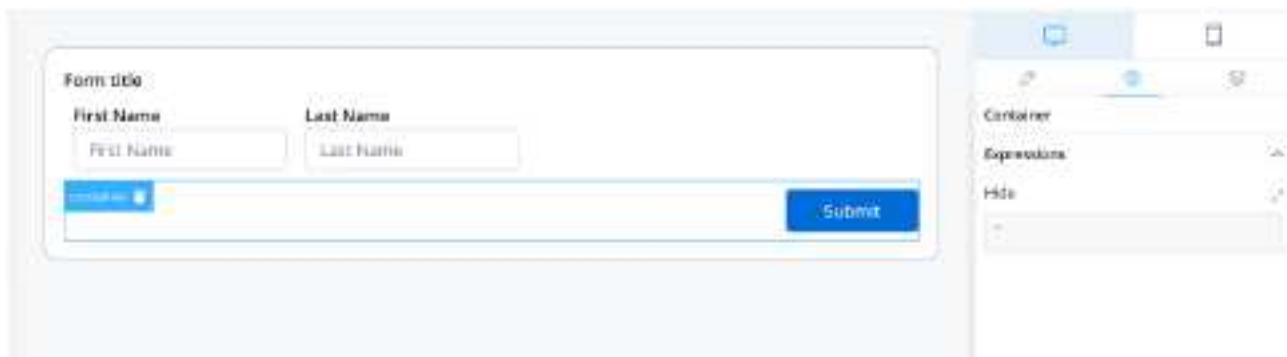
A container is a versatile element that allows you to group components and align them as desired.



The following properties can be configured in the container:

## Settings

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



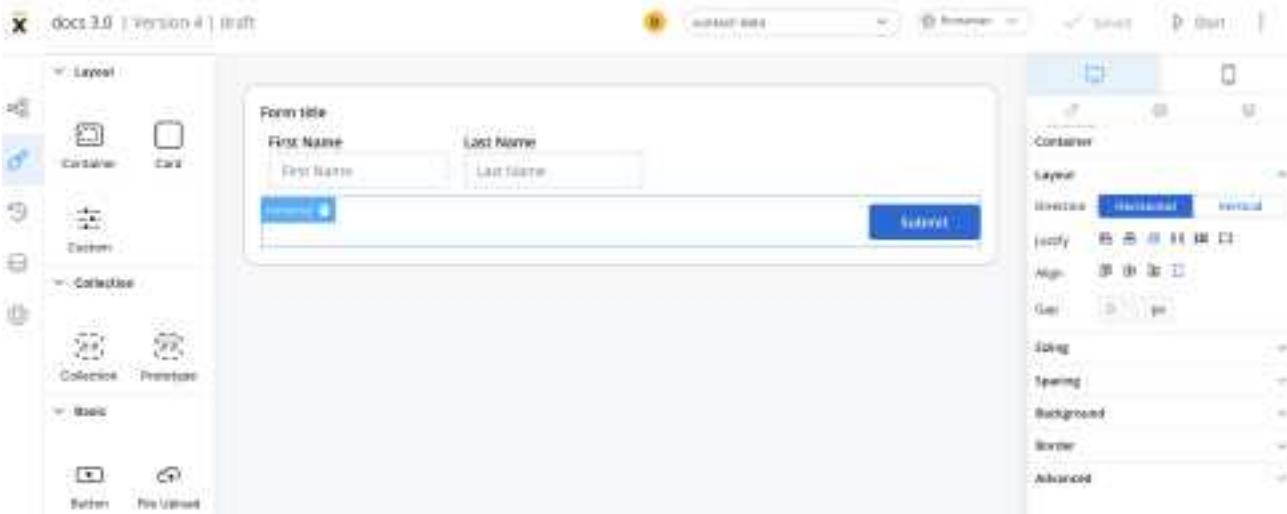
## Styling

- **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:



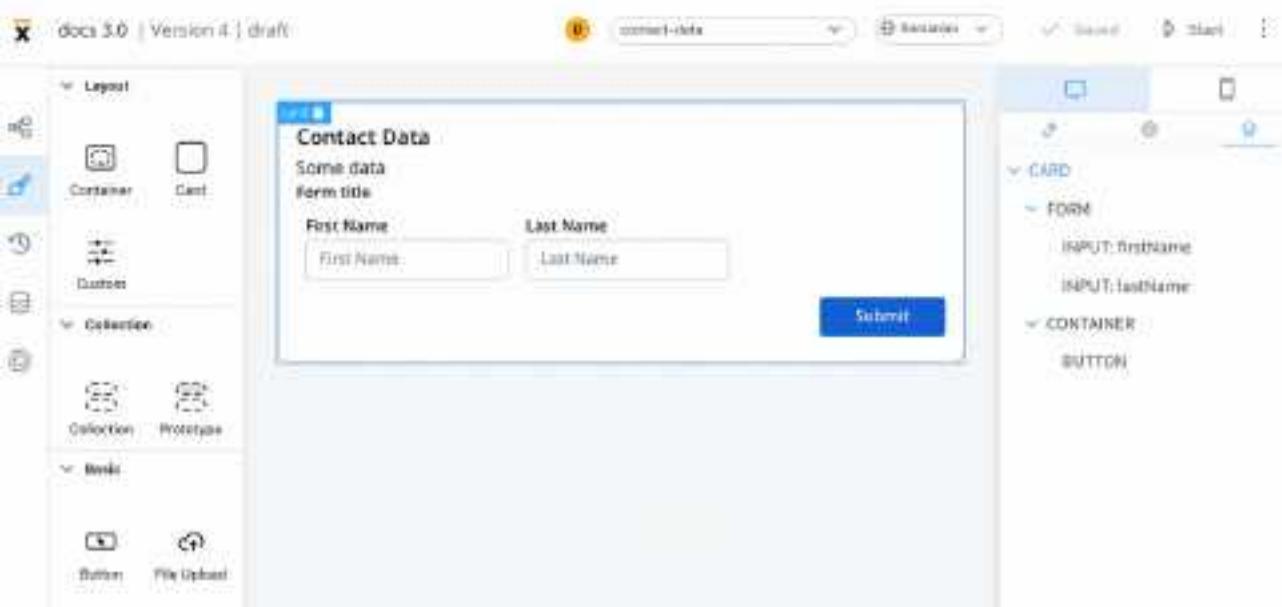
For more information about styling and layout configuration, check the following section:

» [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, settings, and share icons. Below these are two sections: "Card" and "Message". The "Card" section contains a title input field with "eg. title" placeholder text. The "Message" section contains a message input field with a small "x" icon in the top right corner. A large "Properties" section is expanded, containing fields for "Title" (placeholder "eg. title") and "Subtitle" (placeholder "eg. subtitle"). Under "Card style", the "border" option is selected, highlighted in blue. There is also a "raised" option. At the bottom of the properties section is a checkbox labeled "Has Accordion" with an unchecked state.

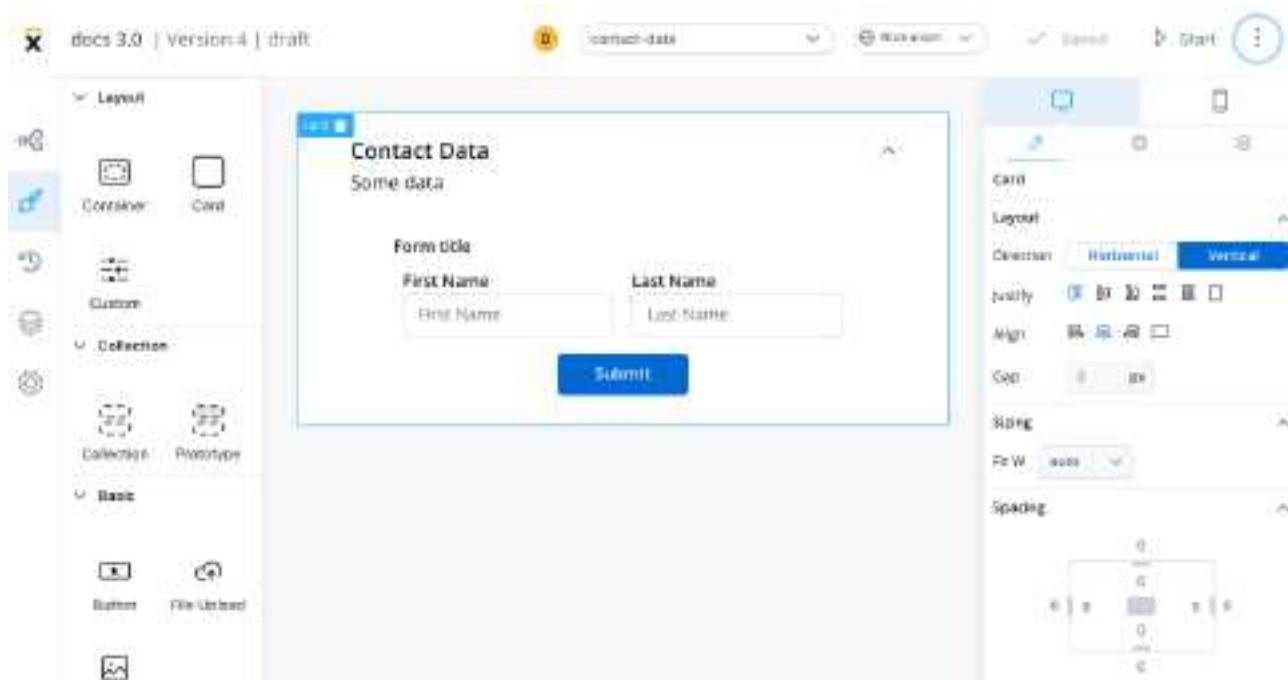
## Styling

- **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:

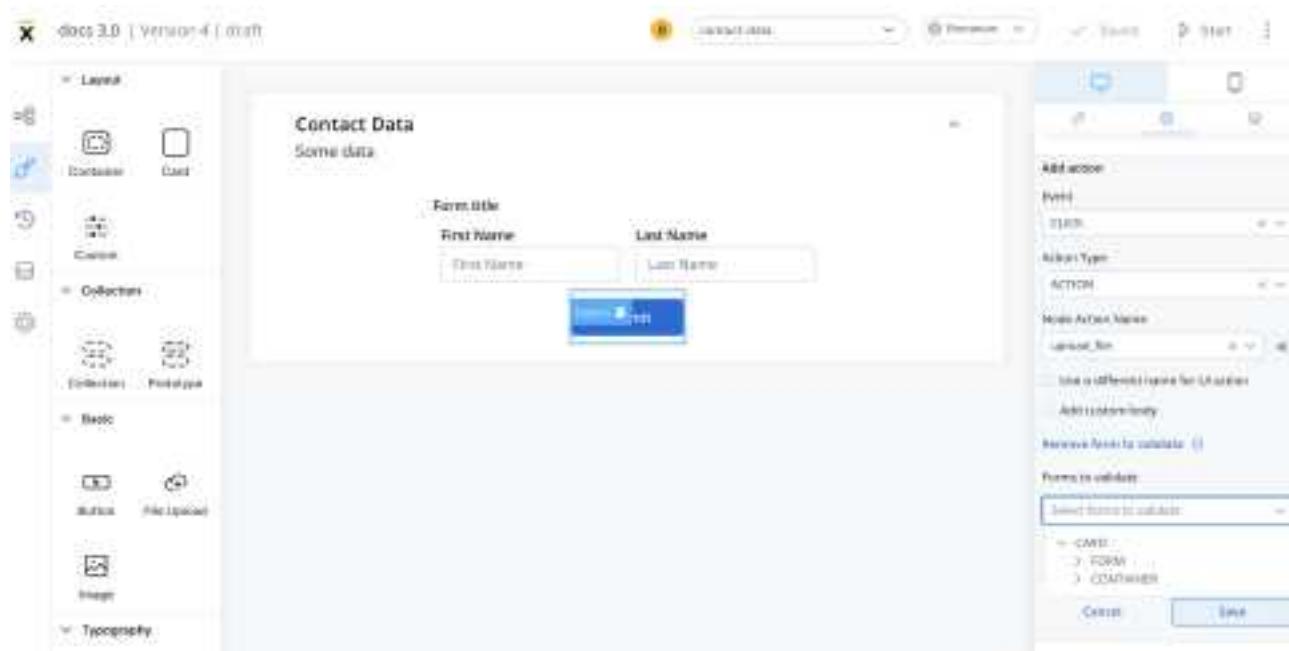


For more information about styling and layout configuration, check the following section:

» [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*.

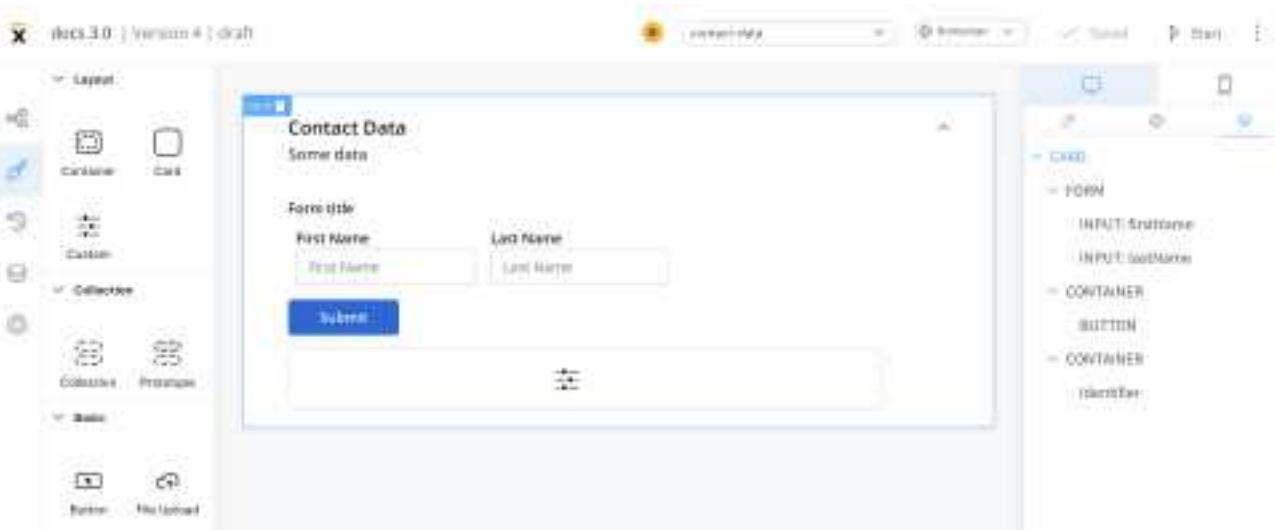


Was this page helpful?

# 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 a configuration interface for a component. At the top, there are icons for a computer monitor and a smartphone, followed by a gear icon and a downward arrow icon. Below this, the word "Custom" is displayed in bold. A section titled "Identifier" contains a text input field with the placeholder "Identifier". To the right of the input field are two small icons: a pencil and a trash can. Below this section is a header "Input Keys" with a collapse/expand arrow to its right. Under "Input Keys", there is a list item "CustomComponent" with edit and delete icons next to it. Below this is a button labeled "Add an option" with a plus sign icon. Another section titled "UI Action" is shown below, featuring 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,
```

```
"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

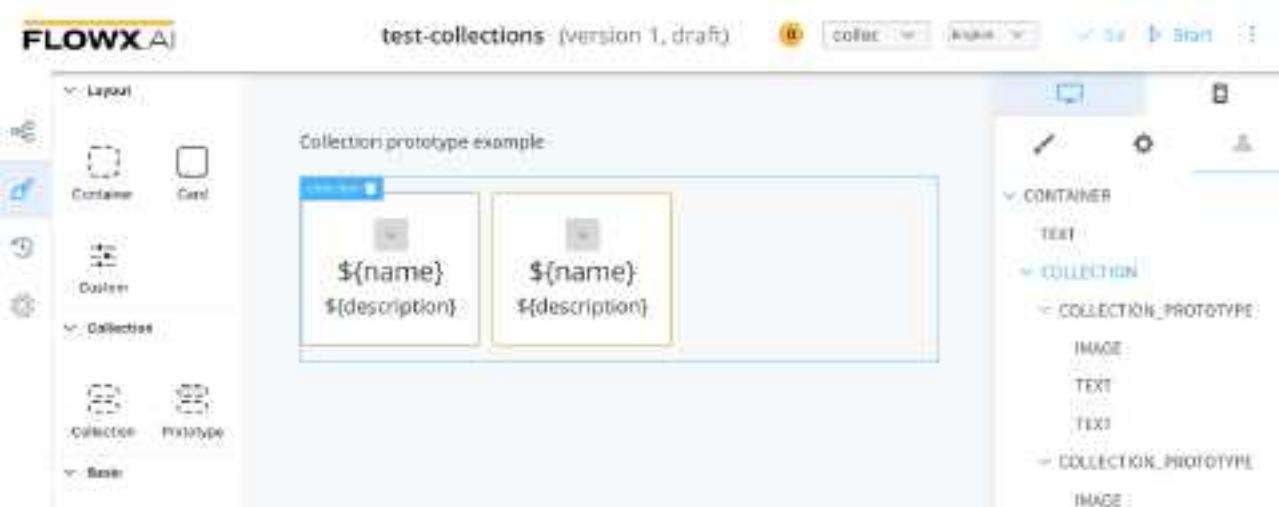
This additional container type is needed to allow multiple prototypes to be defined for a single **Collection**. This allows elements from the same collection to be displayed differently.

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

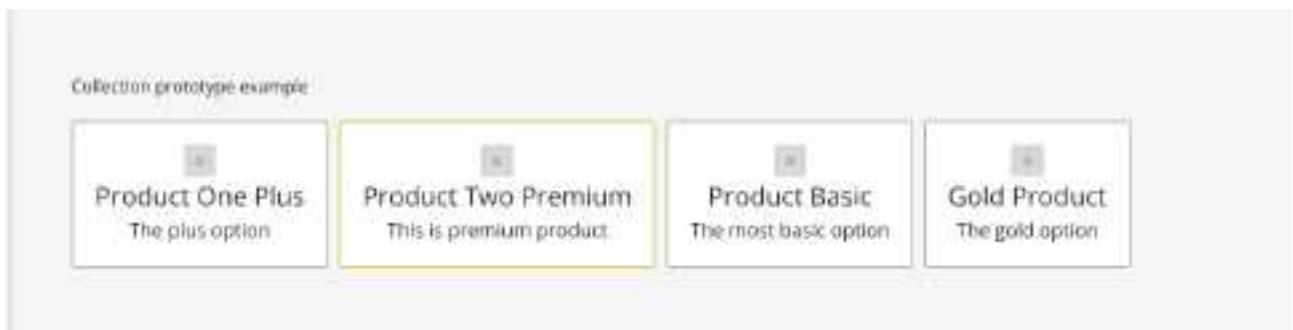


Source collection data example for products:

```
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',
}
]
```

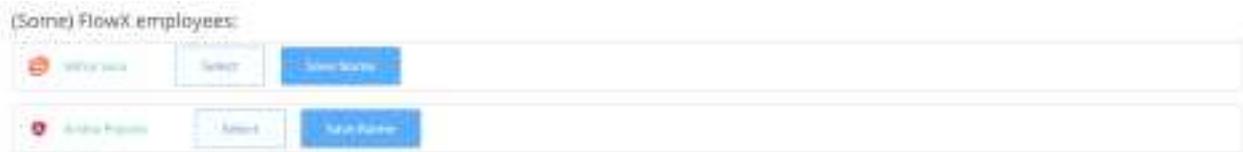
The above configuration will render:



## 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**.

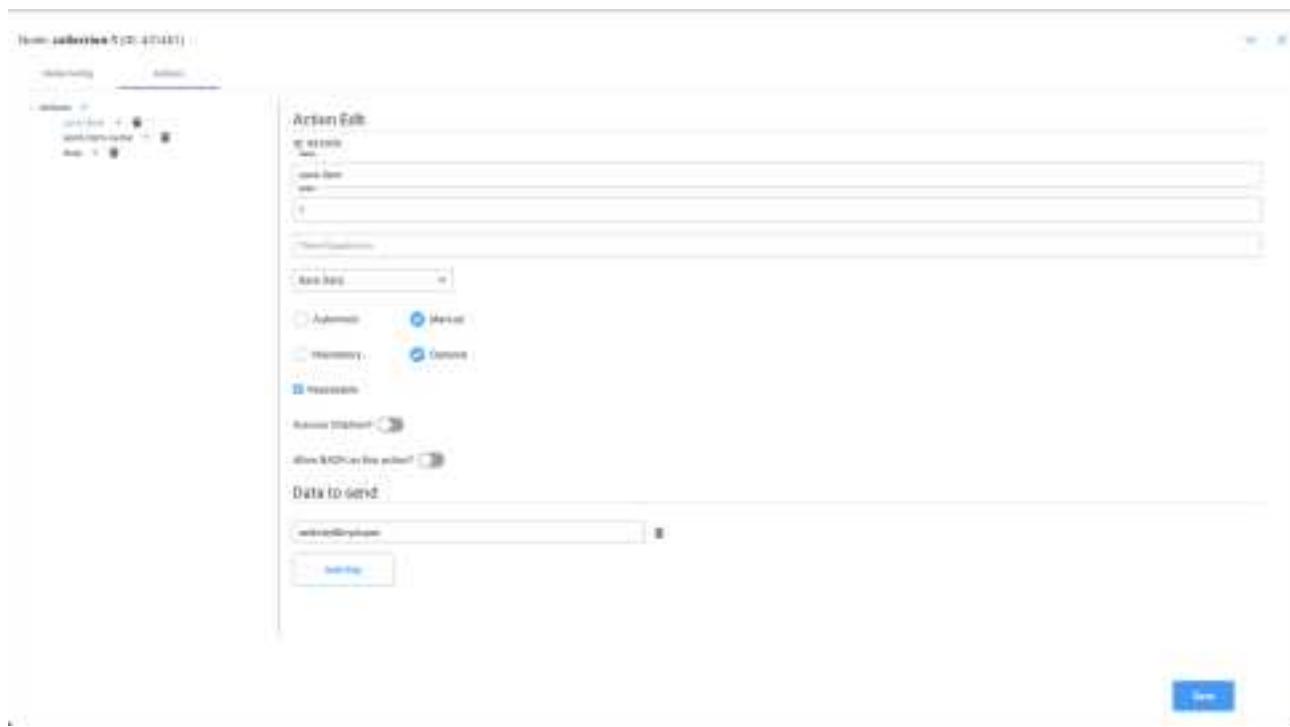
To showcase these differences, we'll use the next example:



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:



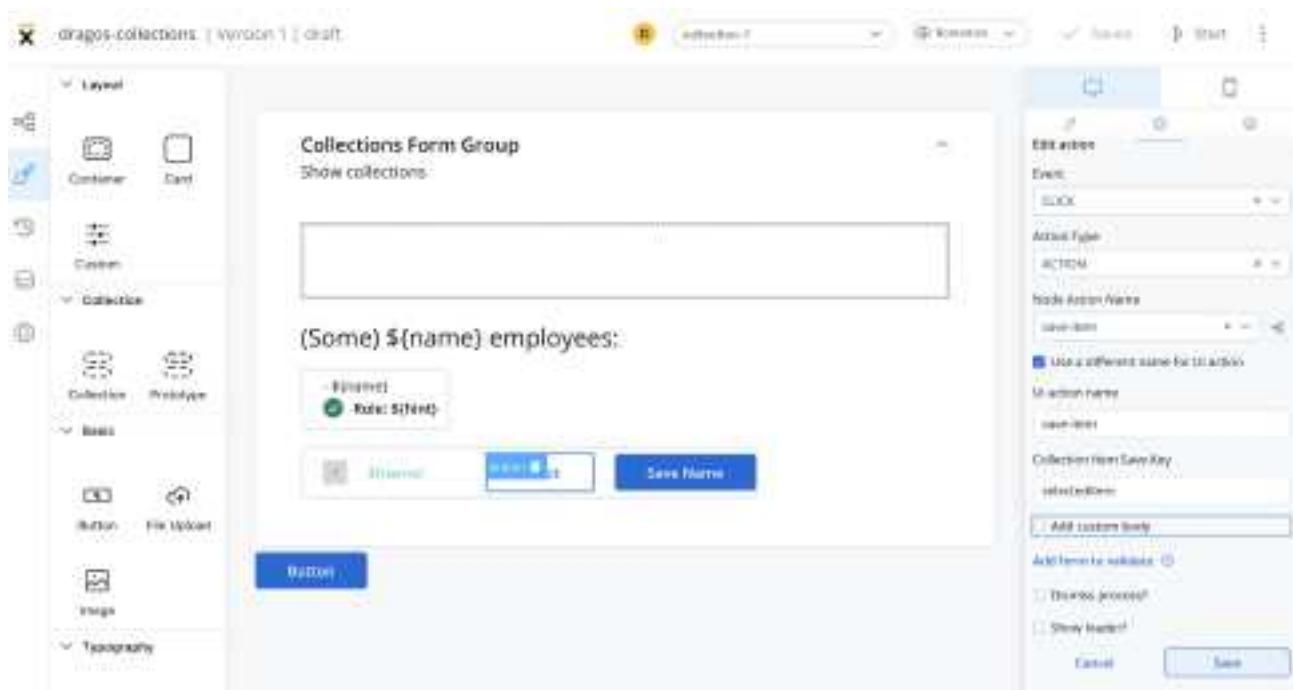
This **save-item** action is **manual** (since it will be triggered by the user) and **optionally** (since selecting an employee is not a requirement to go to the next **Node** in the process).

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:

The screenshot shows the 'Process status' screen with the following details:

- Process status:** Shows the process is in progress.
- Date:** The process started at 01/01/2023 10:00:00.
- Variables:** A list of variables:
  - processItemIndex: 0
  - processData: [{"id": 1, "name": "John Doe", "age": 30, "city": "New York", "status": "Active"}, {"id": 2, "name": "Jane Smith", "age": 28, "city": "Los Angeles", "status": "Active"}, {"id": 3, "name": "Mike Johnson", "age": 32, "city": "Chicago", "status": "Active"}, {"id": 4, "name": "Sarah Davis", "age": 29, "city": "Houston", "status": "Active"}, {"id": 5, "name": "David Wilson", "age": 31, "city": "Phoenix", "status": "Active"}, {"id": 6, "name": "Emily Davis", "age": 27, "city": "San Francisco", "status": "Active"}, {"id": 7, "name": "Aaron Williams", "age": 33, "city": "Seattle", "status": "Active"}, {"id": 8, "name": "Olivia Green", "age": 26, "city": "Austin", "status": "Active"}, {"id": 9, "name": "Jordan White", "age": 35, "city": "Philadelphia", "status": "Active"}, {"id": 10, "name": "Ava Brown", "age": 24, "city": "Portland", "status": "Active"}]
  - selectedEmployee: null
- Actions:** A list of actions:
  - Get Employee: [{"id": 1, "name": "John Doe", "age": 30, "city": "New York", "status": "Active"}]
  - Set Selected Employee: [{}]
- Logs:** No logs present.
- Buttons:** Buttons for 'Start/Stop', 'Reset Current Task', and 'Stop Task'.
- Status:** EXECUTED (Normal).
- Time:** 00:00:00.000

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



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:

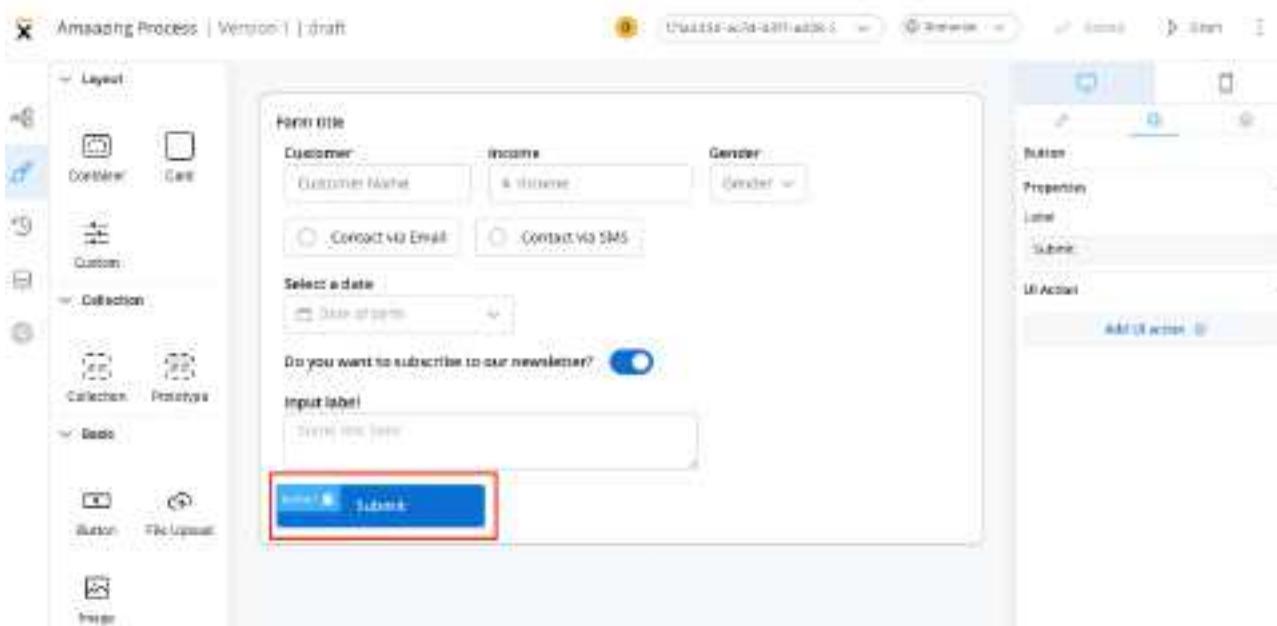
### Properties

- **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



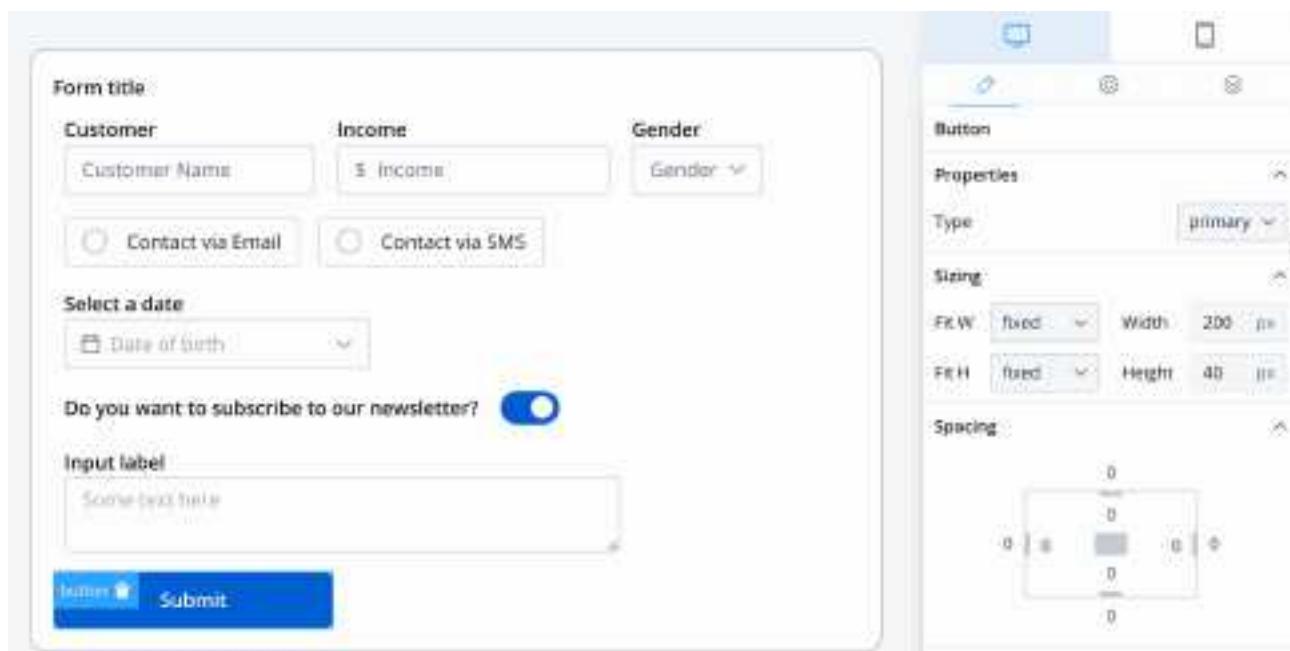
More details on how to configure UI actions can be found [here](#).

## 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

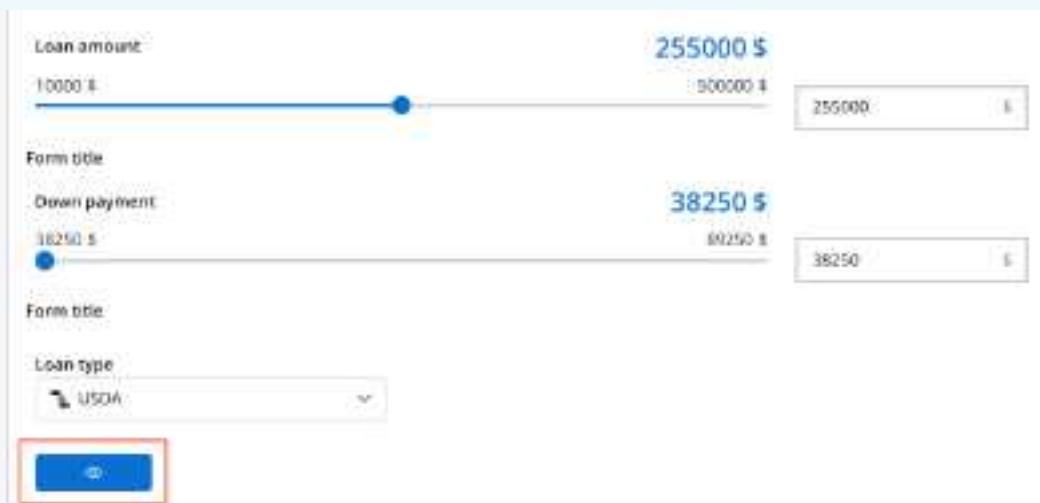
**!** INFO

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.



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

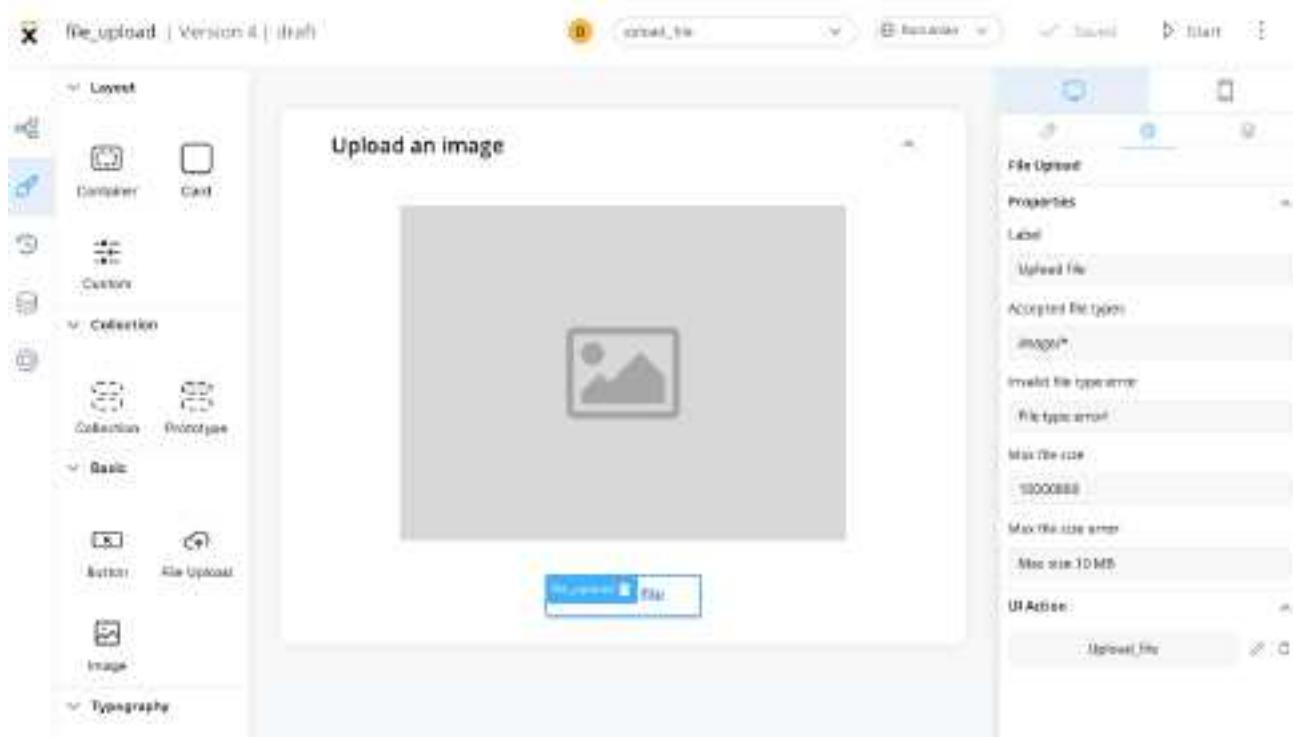
- **Label** - it allows you to set the label that appears on the button
- **Accepted file types** - the accept attribute takes as its value a string containing one or more of these unique file type specifiers, **separated by commas**, may take the following forms:

| 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                             |
| <b>MIME type</b> 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**

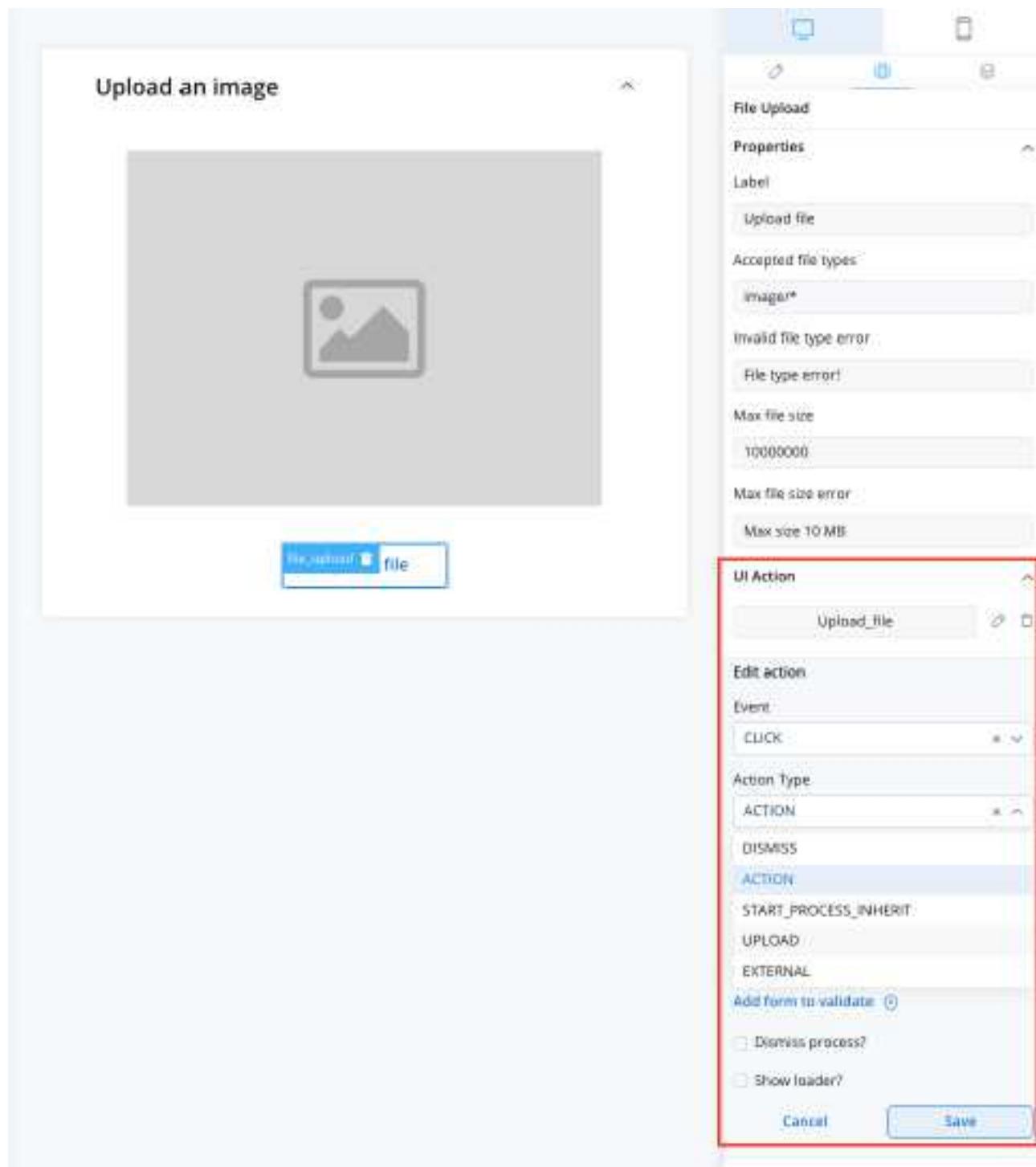
Example of an upload file button that accepts image files:



## UI action

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

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



## !(INFO)

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

## Button styling

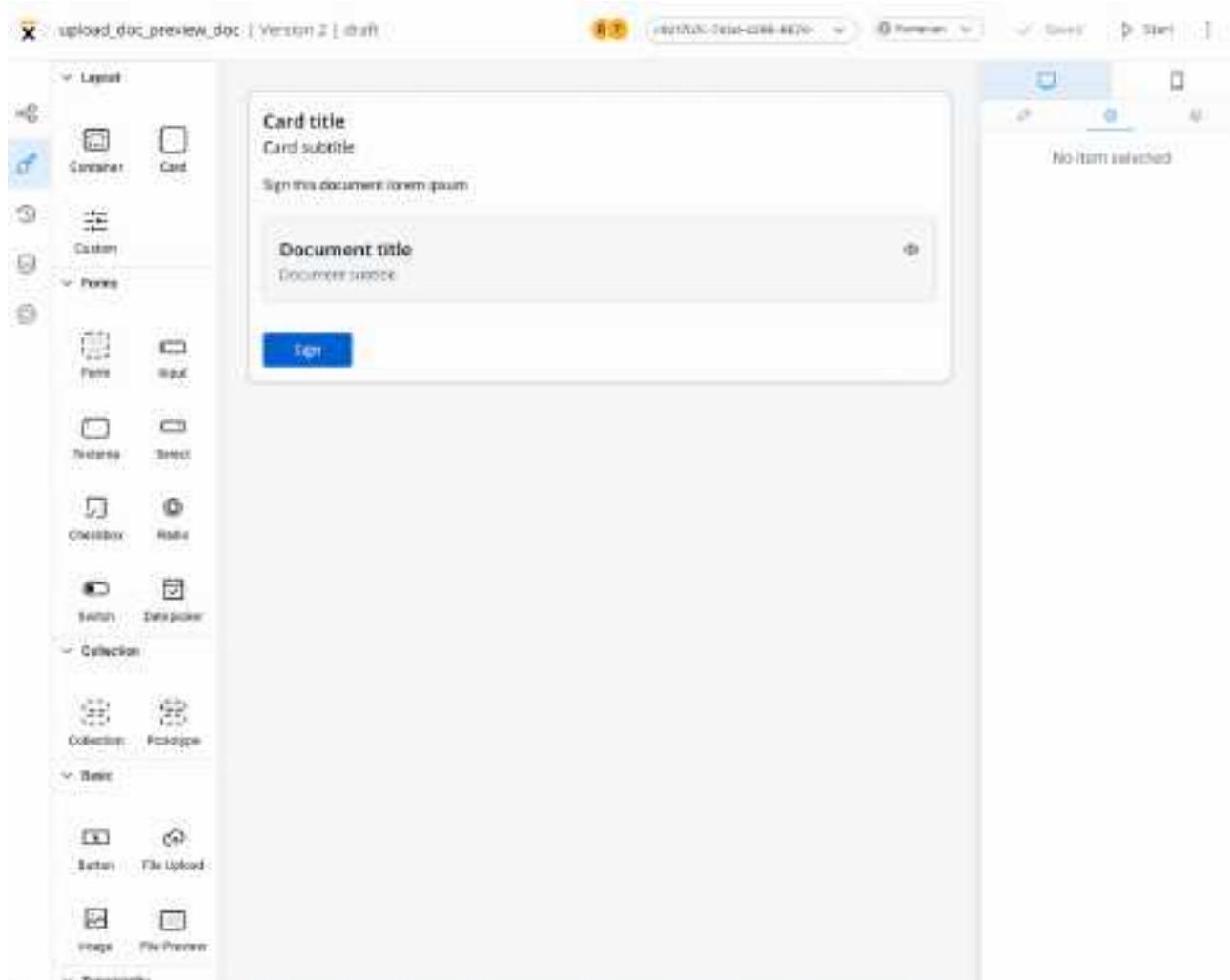
The file upload button can be styled using valid CSS properties (more details [here](#))

[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.

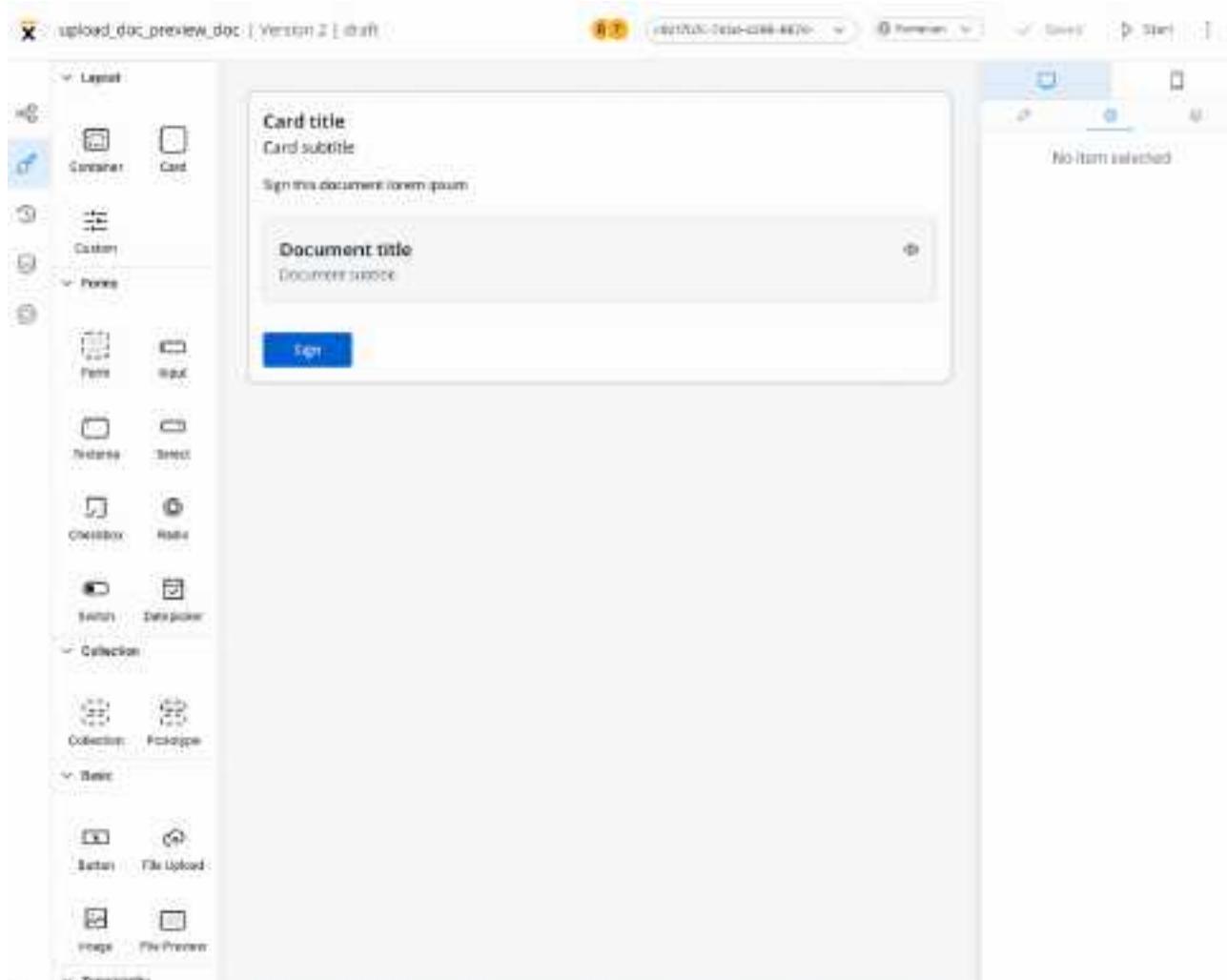


File Preview UI elements offer various benefits such as conveying information, improving the aesthetic appeal of an interface, providing visual cues and feedback or presenting complex data or concepts in a more accessible way.

## 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

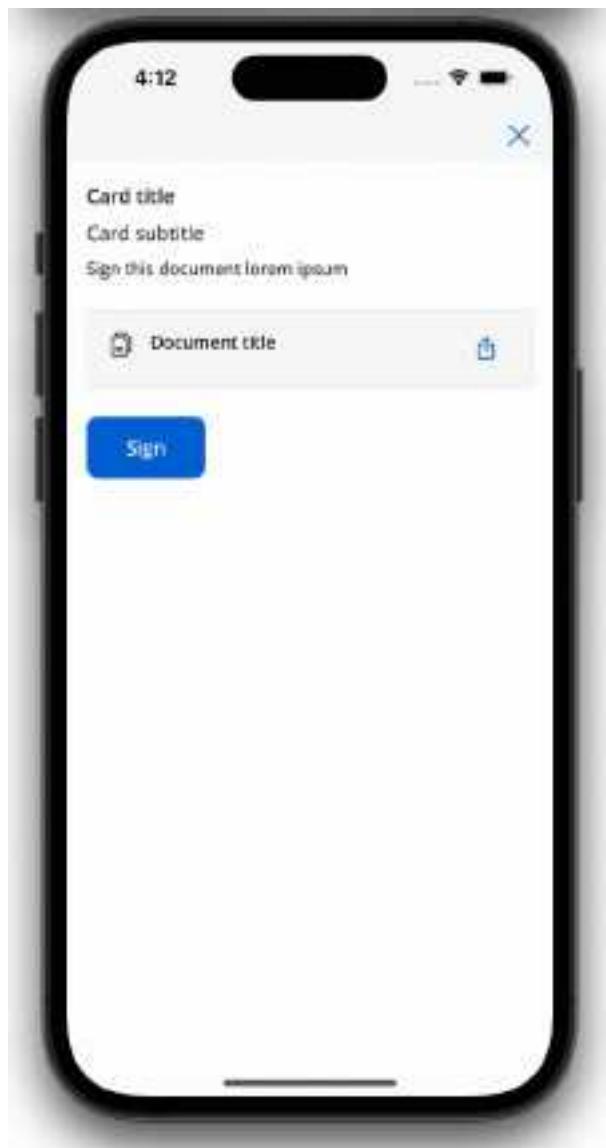
It's worth noting that the inline modal view can raise accessibility issues if the file preview's height exceeds the screen height.

## File Preview properties (mobile)

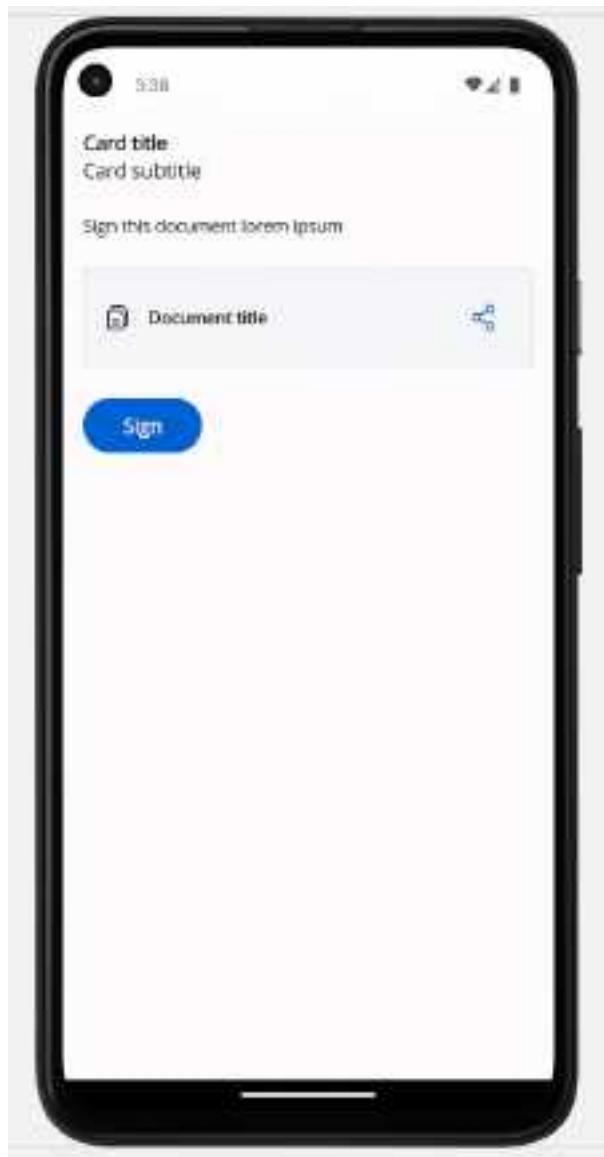
### INFO

Both iOS and Android devices support the share button.

## iOS



## Android



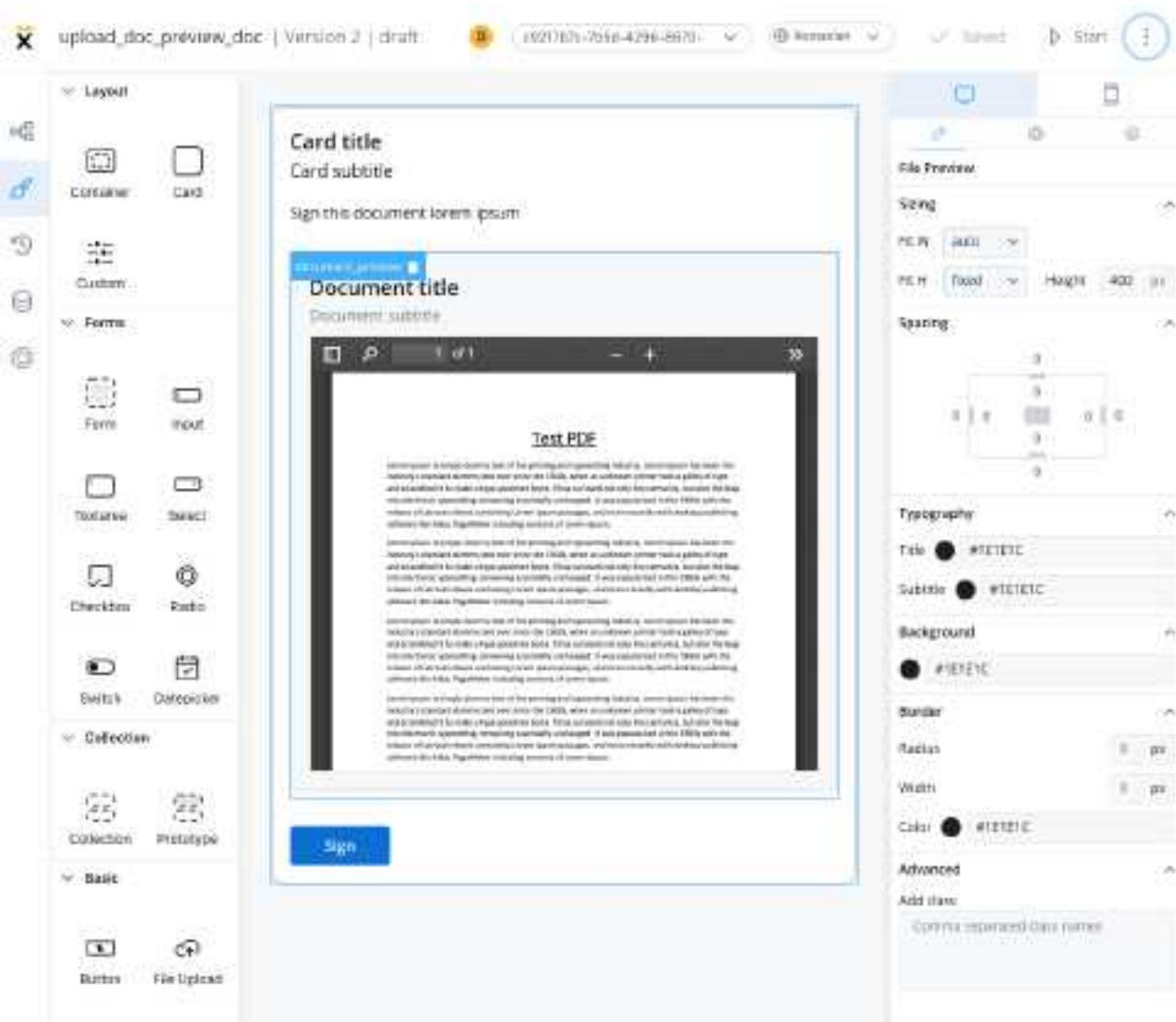
## File preview styling

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

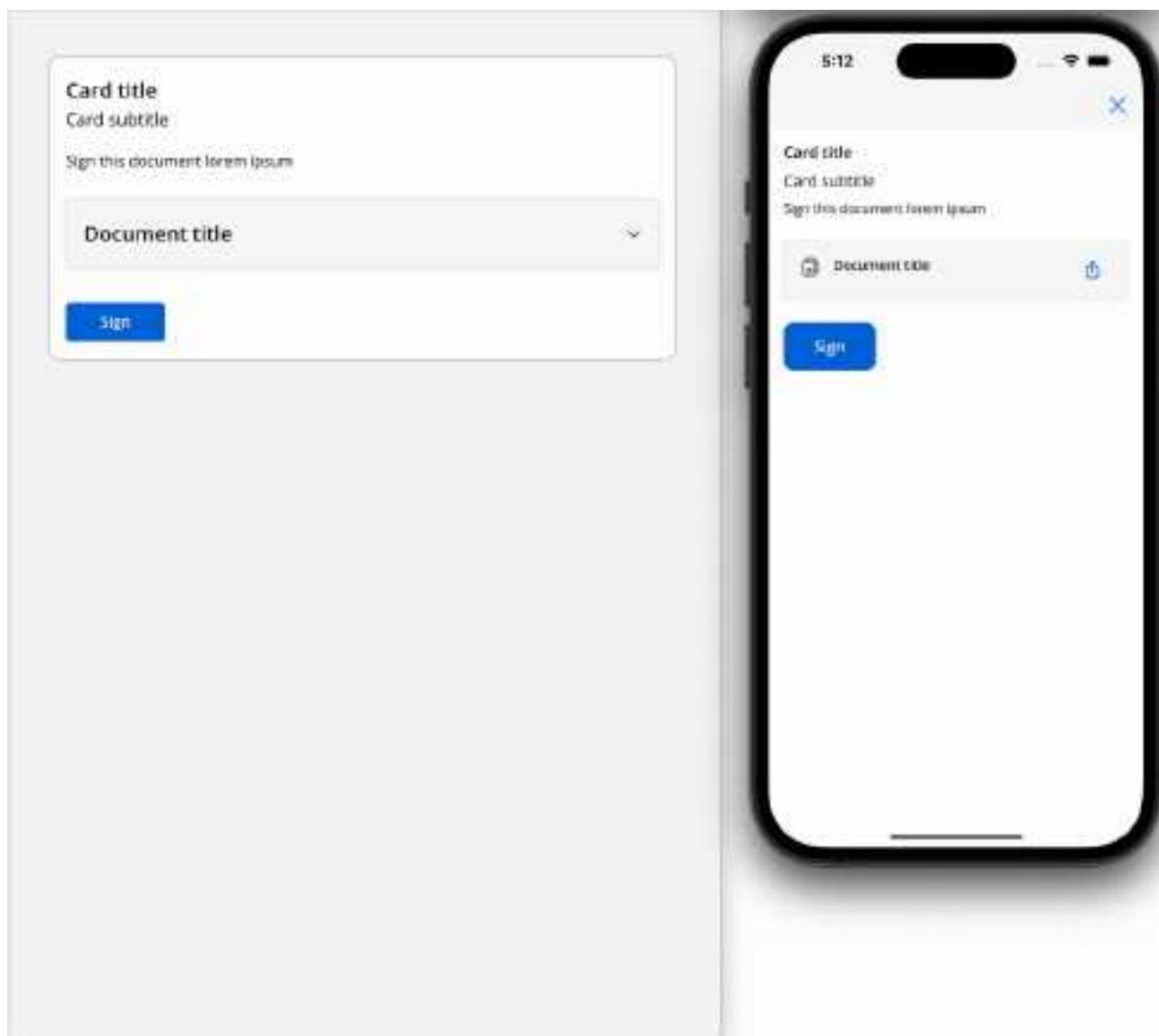
- **Fit W - auto**
- **Fit H - fixed / Height - 400 px**



## File Preview example

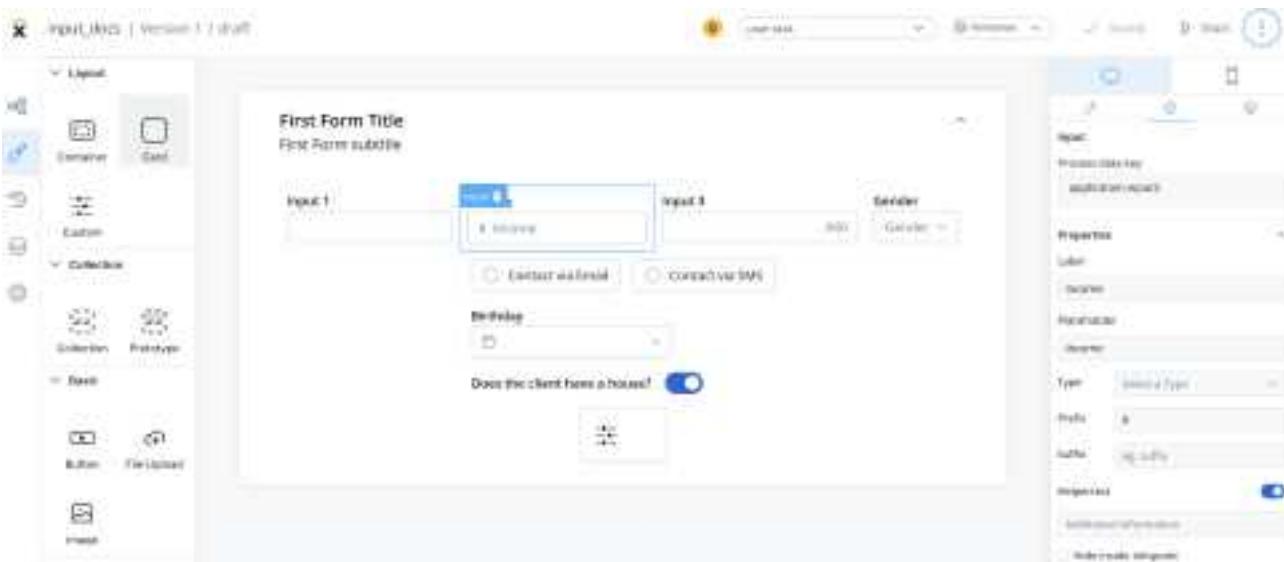
Below is an example of a File Preview UI element with the following properties:

- **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
- **Helptext** - 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 is a sidebar with icons for Input, Typography, Text, Form, Buttons, Checkboxes, and Images. The main area displays a form titled "First Form Title" with a subtitle "First Form subtitle". The form contains several input fields: "Input 1" (with a default value of "Hello"), "Input 2" (with a default value of "100"), and a "Gender" dropdown menu. There are also two radio buttons labeled "Contact via Email" and "Contact via SMS", and a "Birthday" date picker. A toggle switch labeled "Does the client have a header?" is turned on. On the right side, there is a properties panel with sections for "Report", "Properties", "Validation", and "Expressions". The "Report" section shows "Process (800 MB)" and "single-document". The "Properties" section includes "Label", "Name", "Placeholder", "Icon", "Type" (set to "Text"), "Prefix", "Suffix", and "Input type" (set to "text"). The "Validation" section has a "Default value" field set to "100".

## 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 shows the same FLOWX.AI process builder interface as the previous one, but with a different configuration for the "Input 1" field. In the properties panel on the right, the "Default value" field for "Input 1" is now set to "HelloWorld". All other elements and configurations remain the same as in the first screenshot.

**First Form Title**  
First Form subtitle

Income: \$ 555

Mortgage: 000

Gender: Gender ▾

This is a helper text.

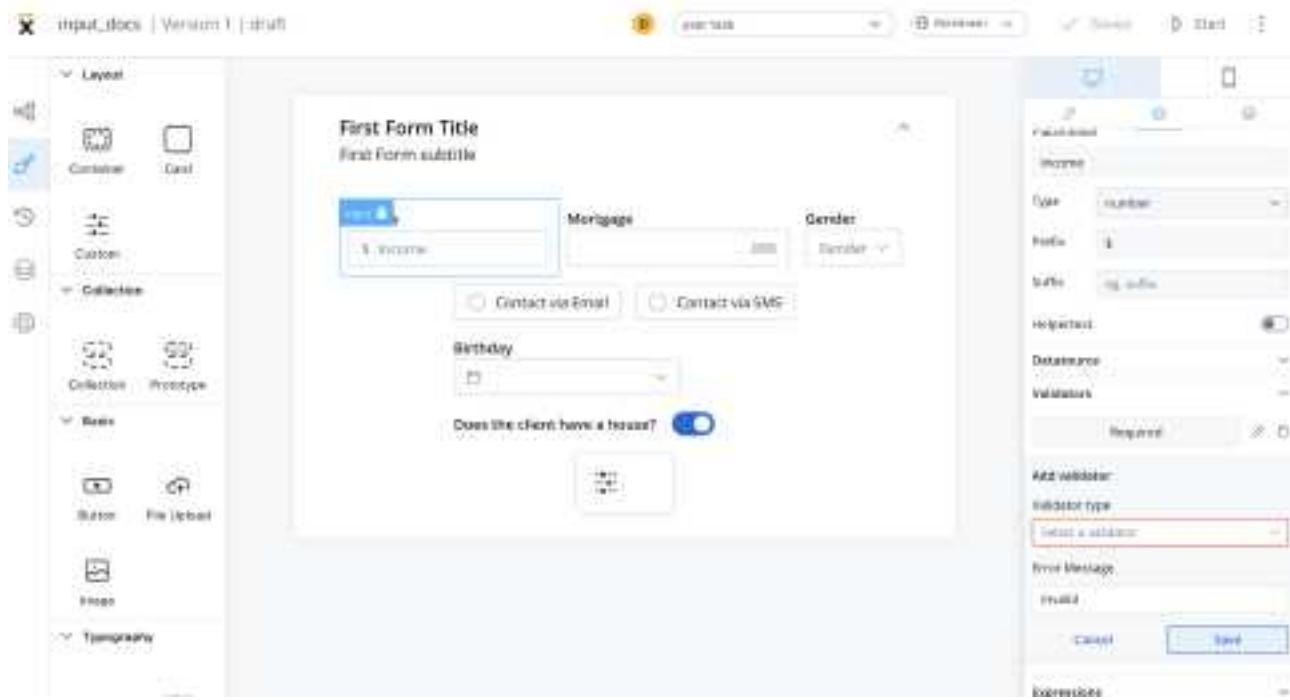
Contact via Email  Contact via SMS

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.

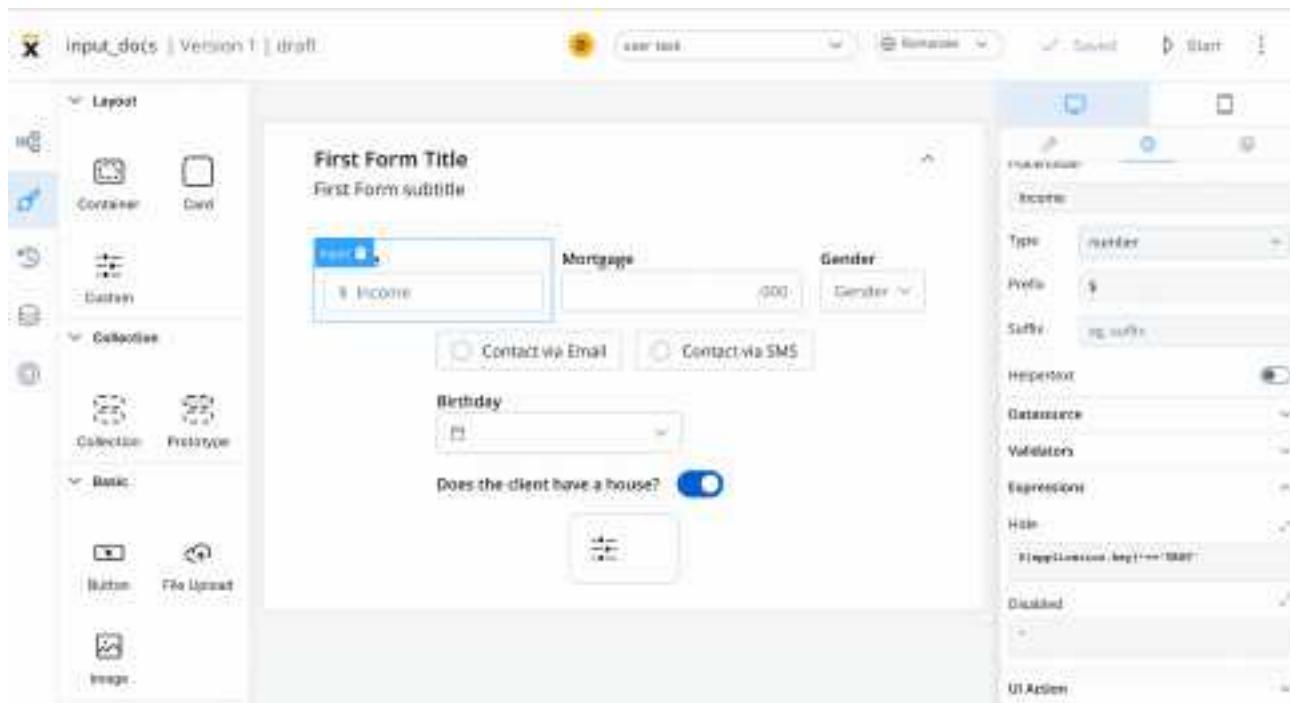
The screenshot shows the FLOWX.AI platform's form builder. On the left, there is a sidebar with various UI component icons grouped under categories like Layout, Collection, and Basic. The main canvas displays a form titled "First Form Title" with a subtitle "First Form subtitle". The form contains several input fields: a dropdown menu labeled "Mortgage", a "Gender" dropdown, and two radio buttons for "Contact via Email" and "Contact via SMS". Below these are a date input field for "Birthday" and a toggle switch for "Does the client have a house?". To the right of the form, there is a detailed configuration panel with sections for "Properties", "Validation", and "Expressions". The "Expressions" section is currently active, showing a code editor with the following JavaScript-like code:

```
if (appContext.page === 'page1') {  
    hide();  
}  
disabled
```

## 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



### (!) 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

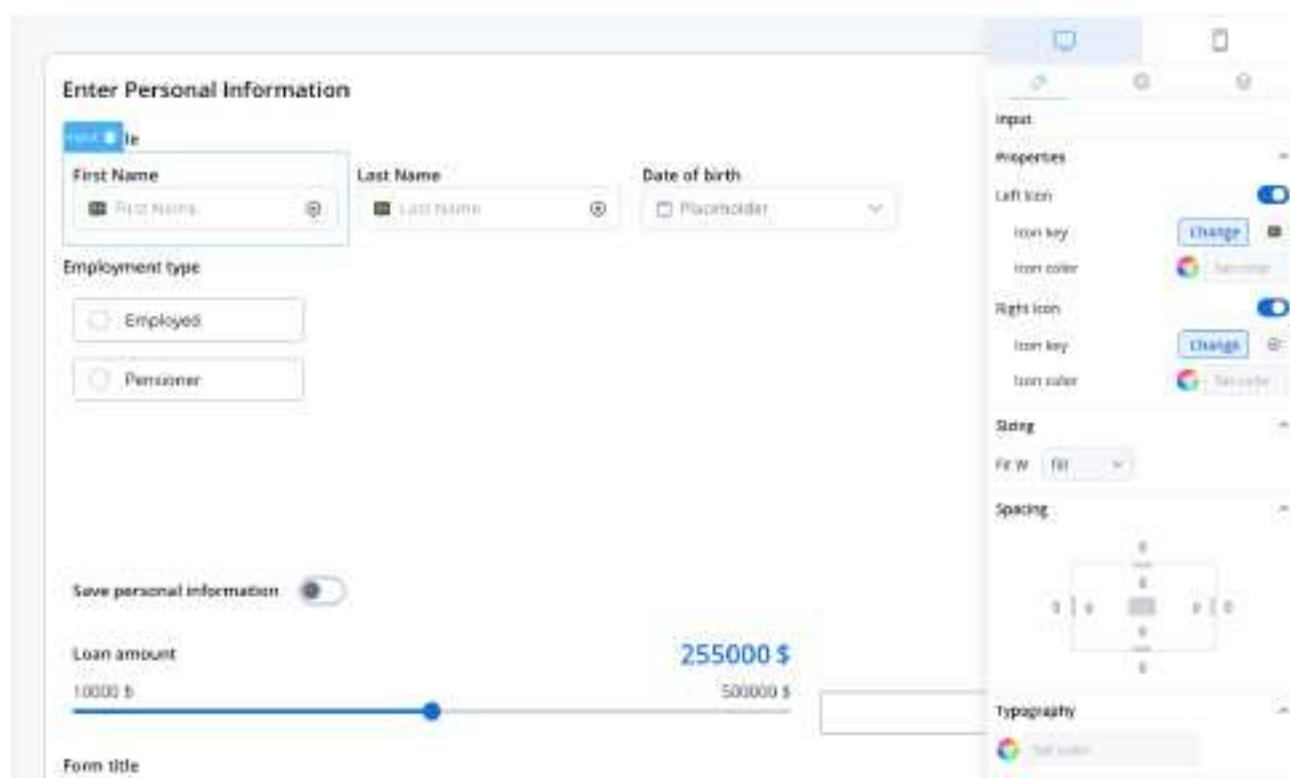
### (!) INFO

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

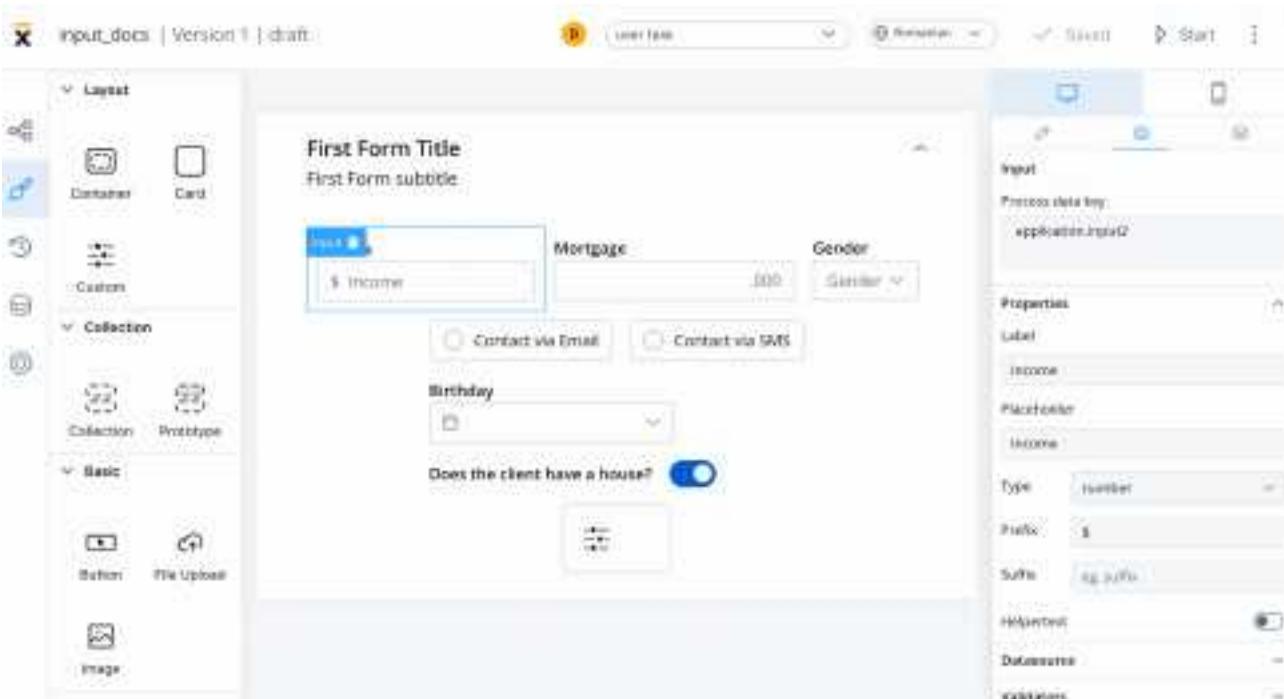
You have the option to enhance the Input element by incorporating two types of icons:

- **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.

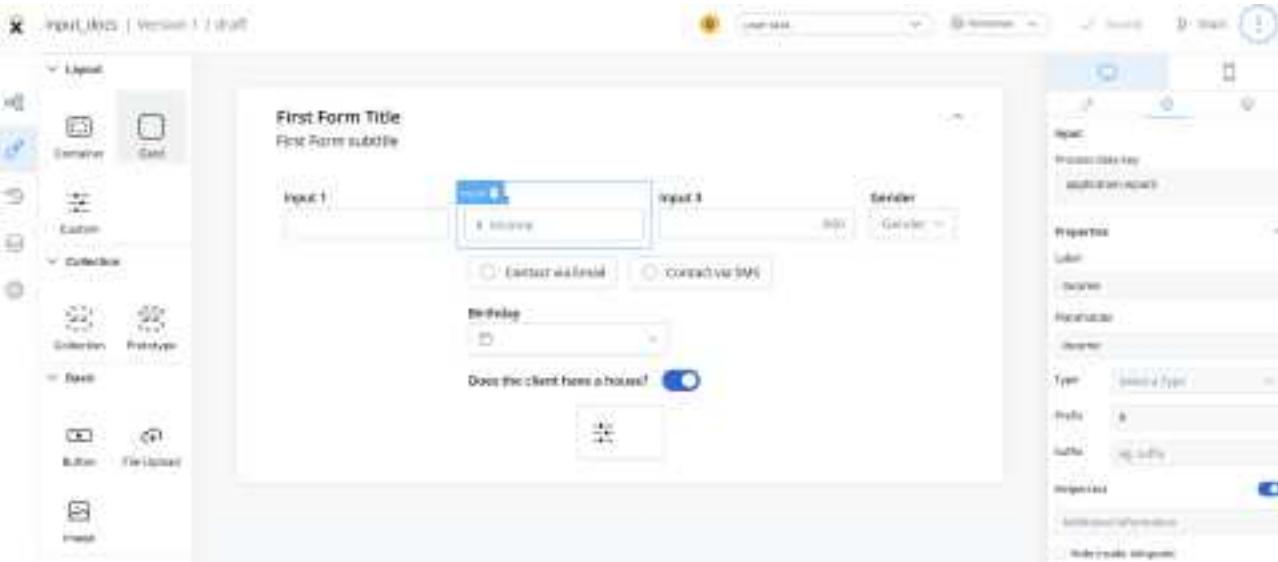


- The Input Field can be styled using valid CSS properties (more details [here](#))



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

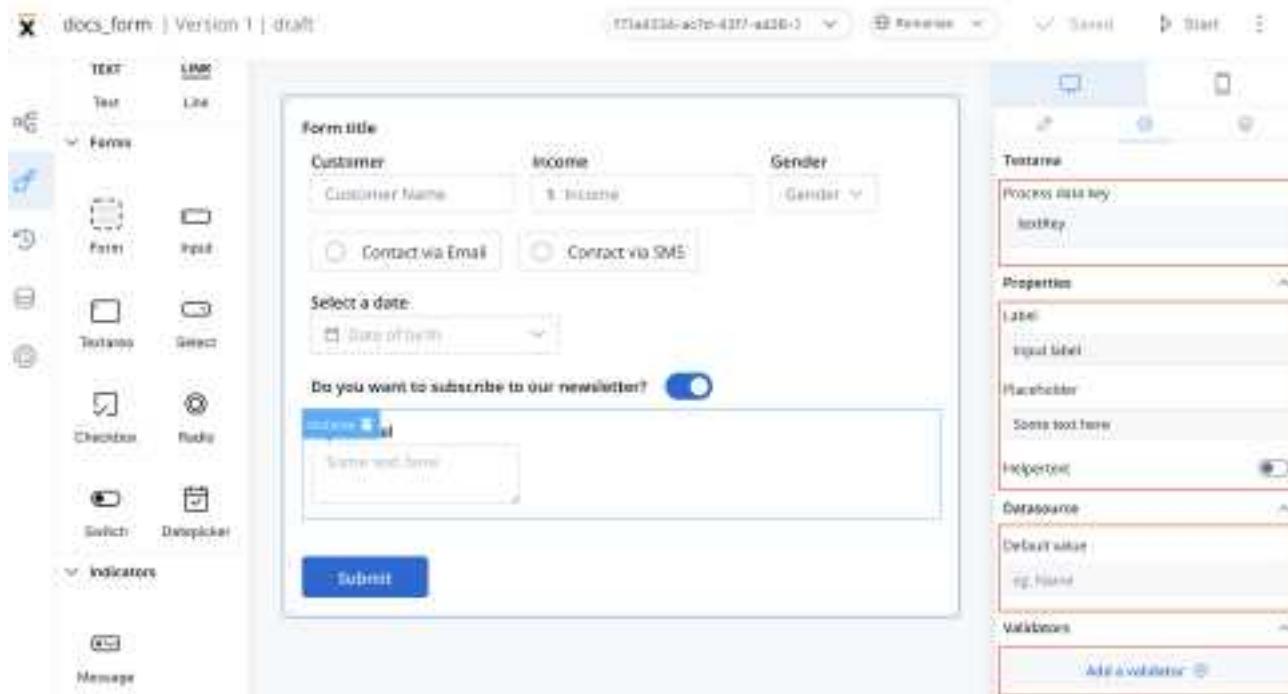
- **Label** - the label of the text area
- **Placeholder** - the placeholder text that appears in the text area
- **Helpertext** - additional information about the text area field (can be hidden inside an infopoint)

## 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](#)).



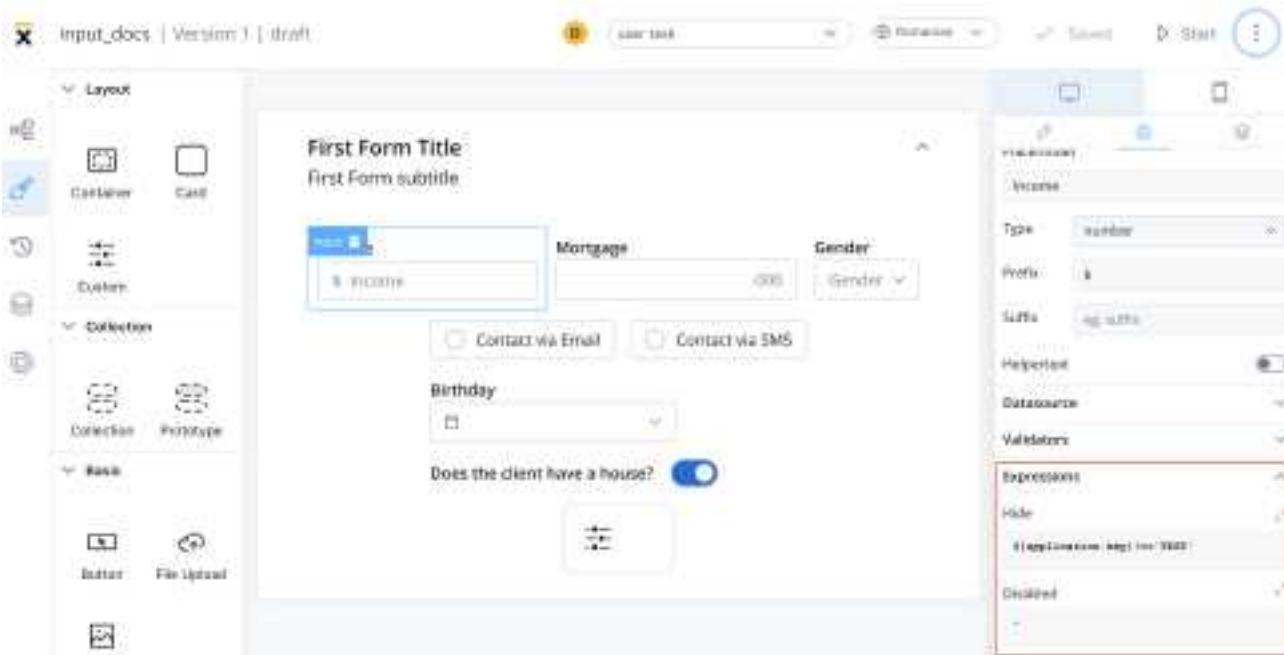
## Expressions

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

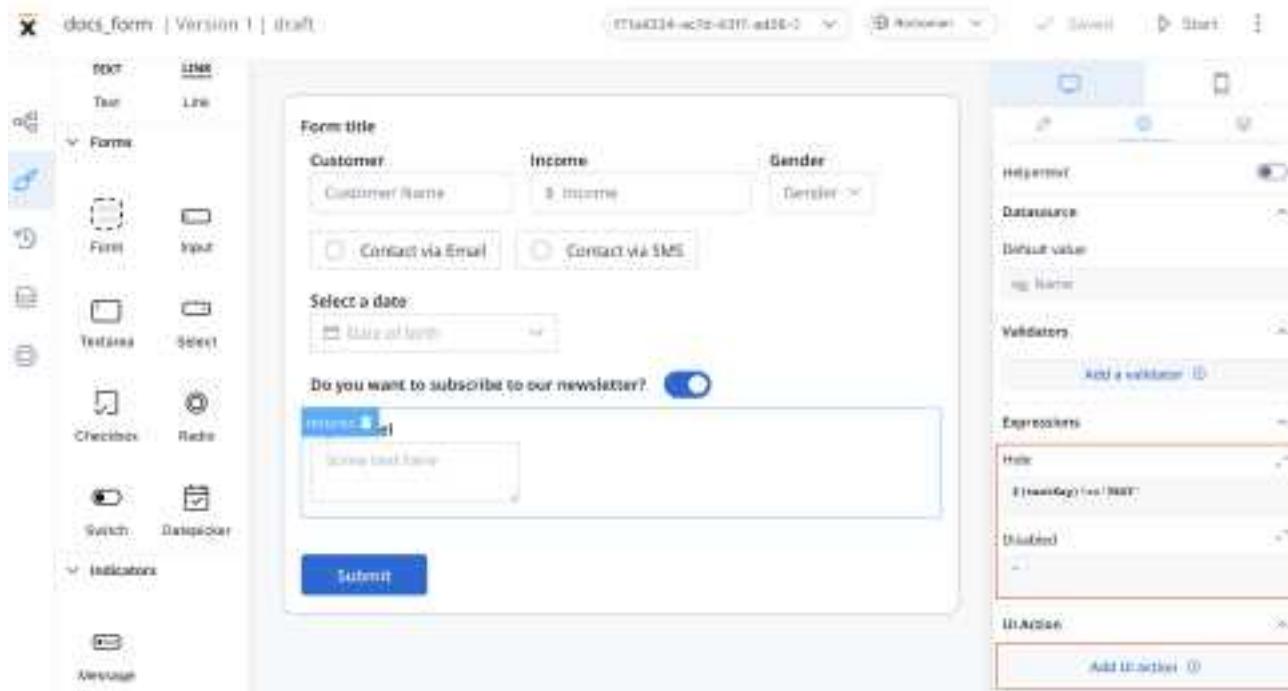
It's important to make sure that disabled fields have the same expression configured under the path expressions → hide.



## 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



## INFO

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

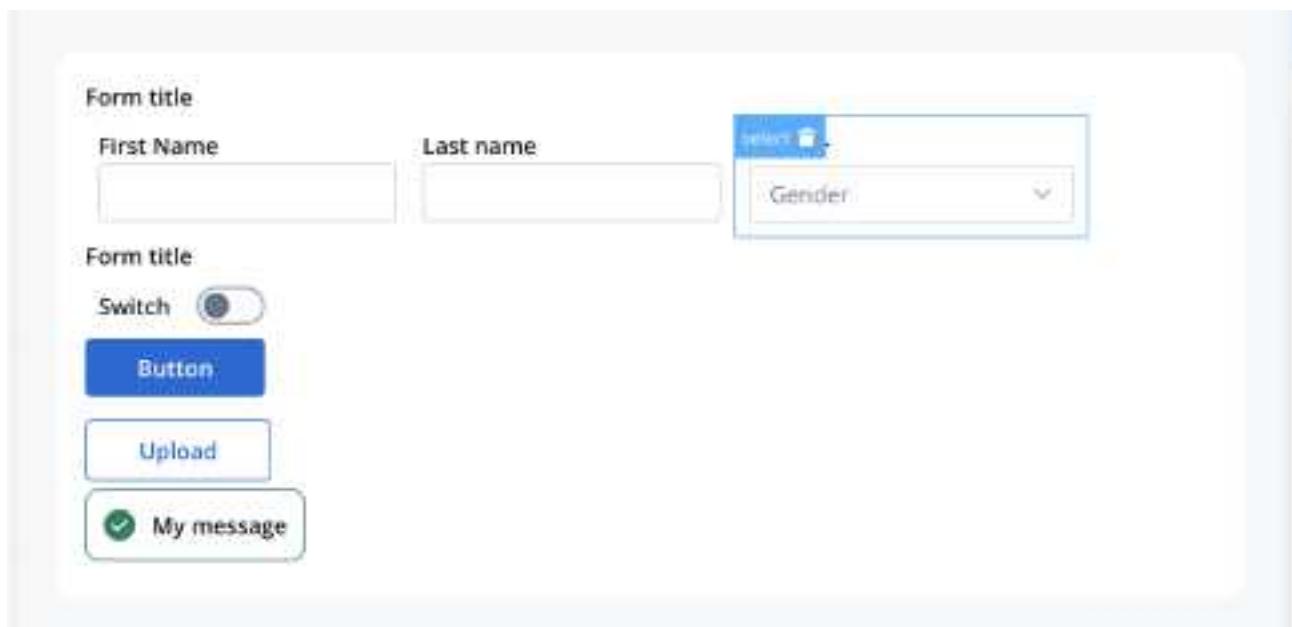
## Text area styling

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](#)).

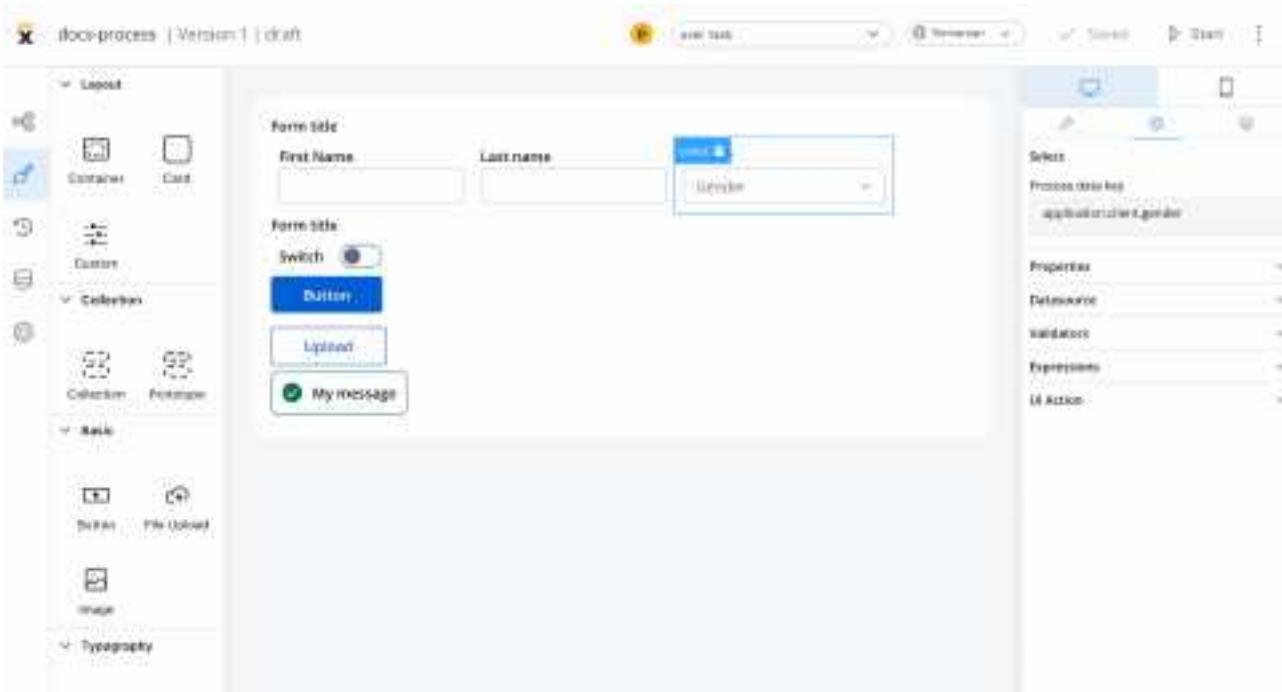
## Expressions

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

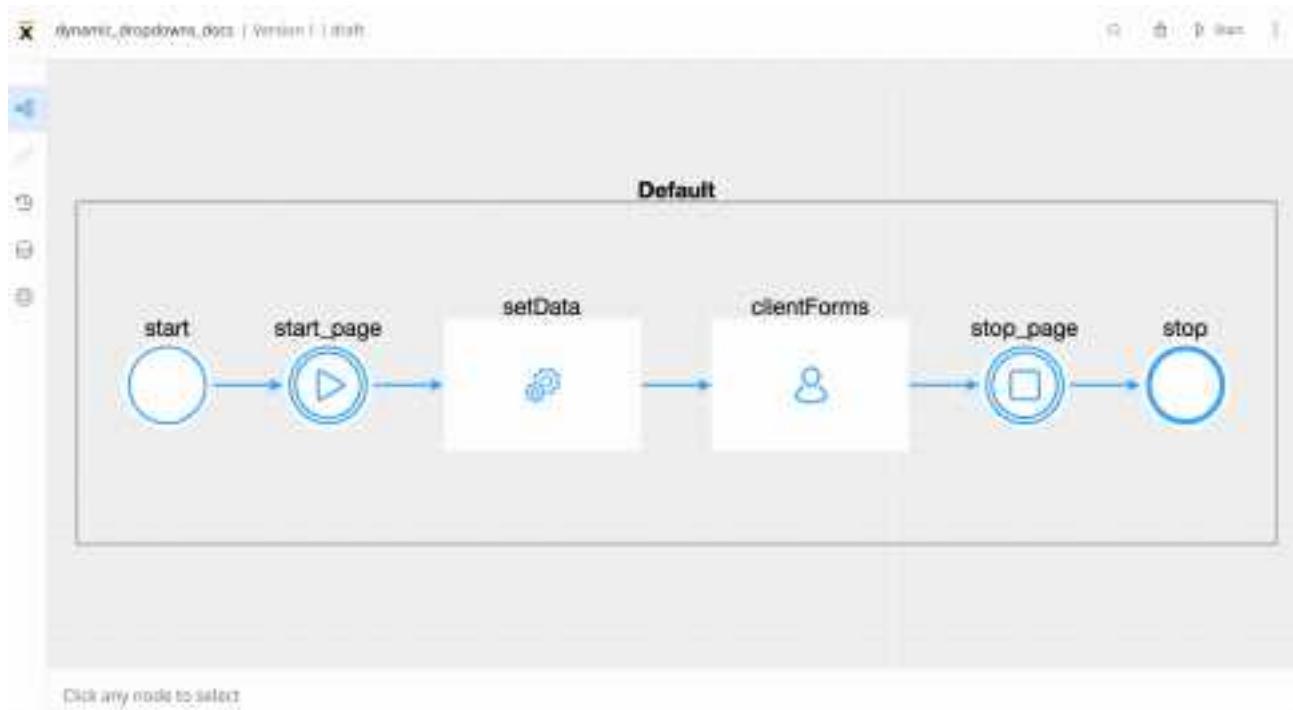
When setting the color, the entire icon is filled with that color, the SVG's fill. Avoid changing colors for multicolor icons.

## Example - Dynamic dropdowns

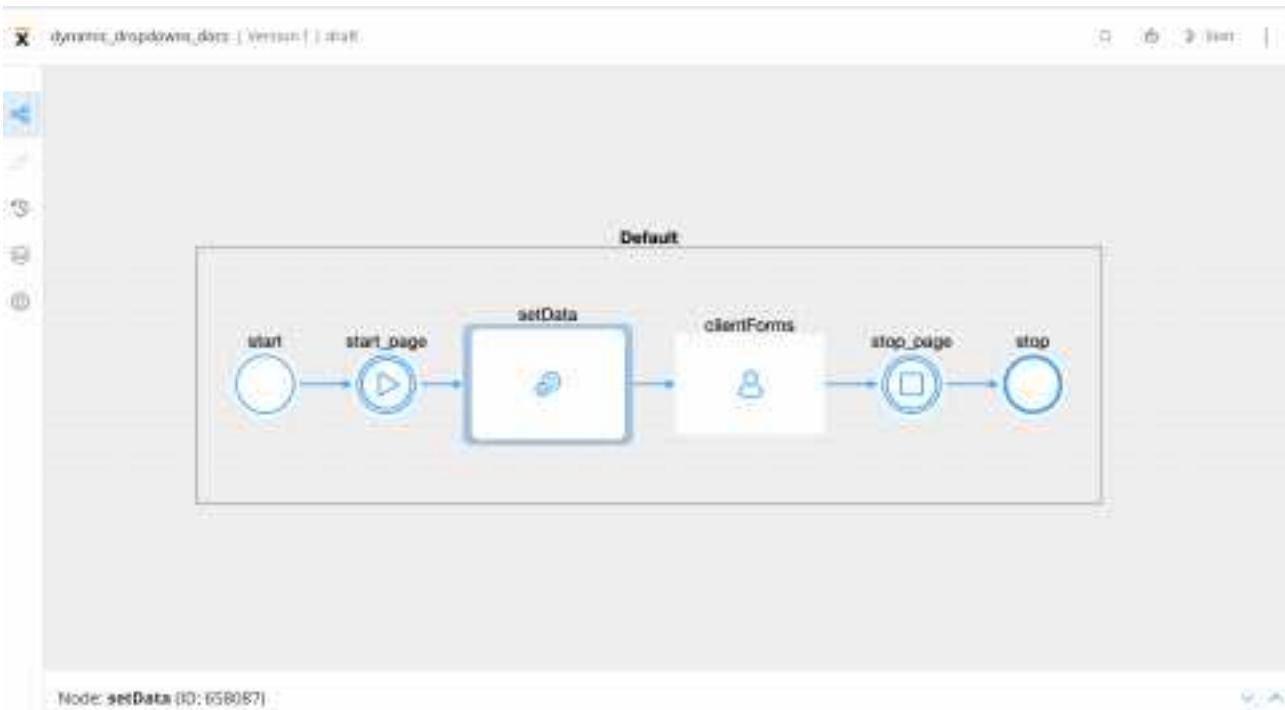
As mentioned previously, you can create dropdowns including static data, enumerations, or **process data**. Let's create an example using **process data** to create a process that contains **dynamic dropdowns**.

To create this kind of process, we need the following elements:

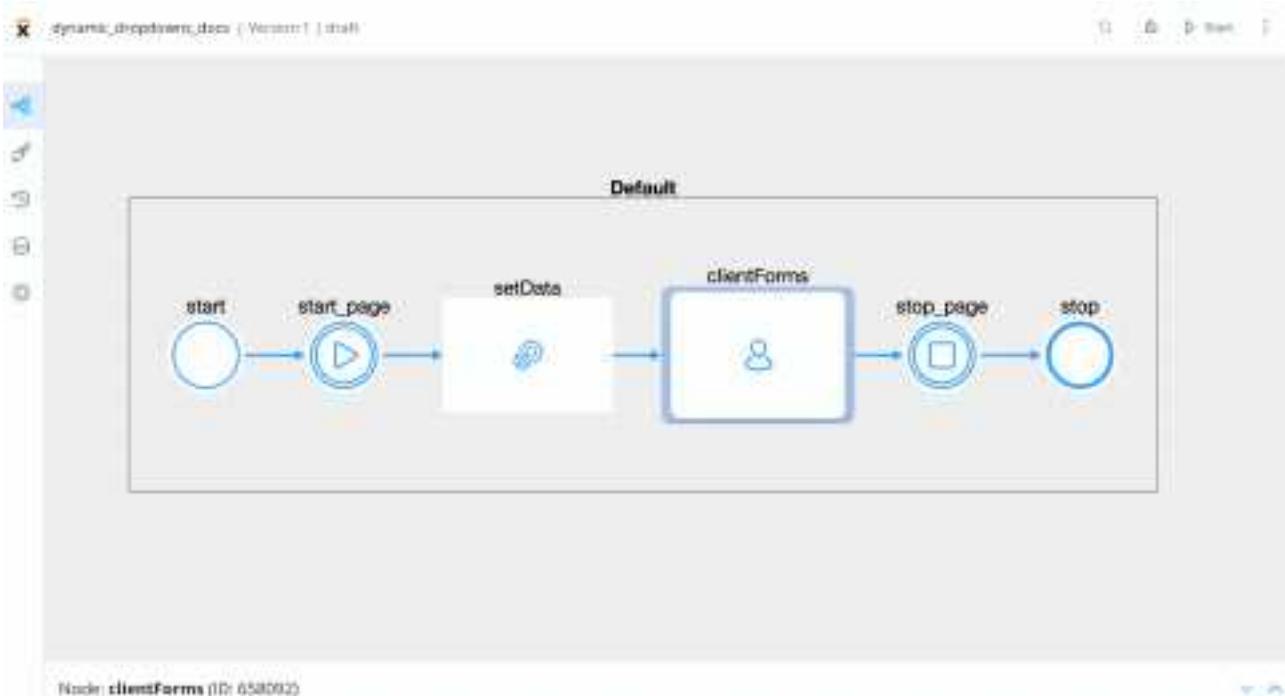
- a **start** node and an **end** node
- a **start milestone** UI element to it and an **end milestone** node



- 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)



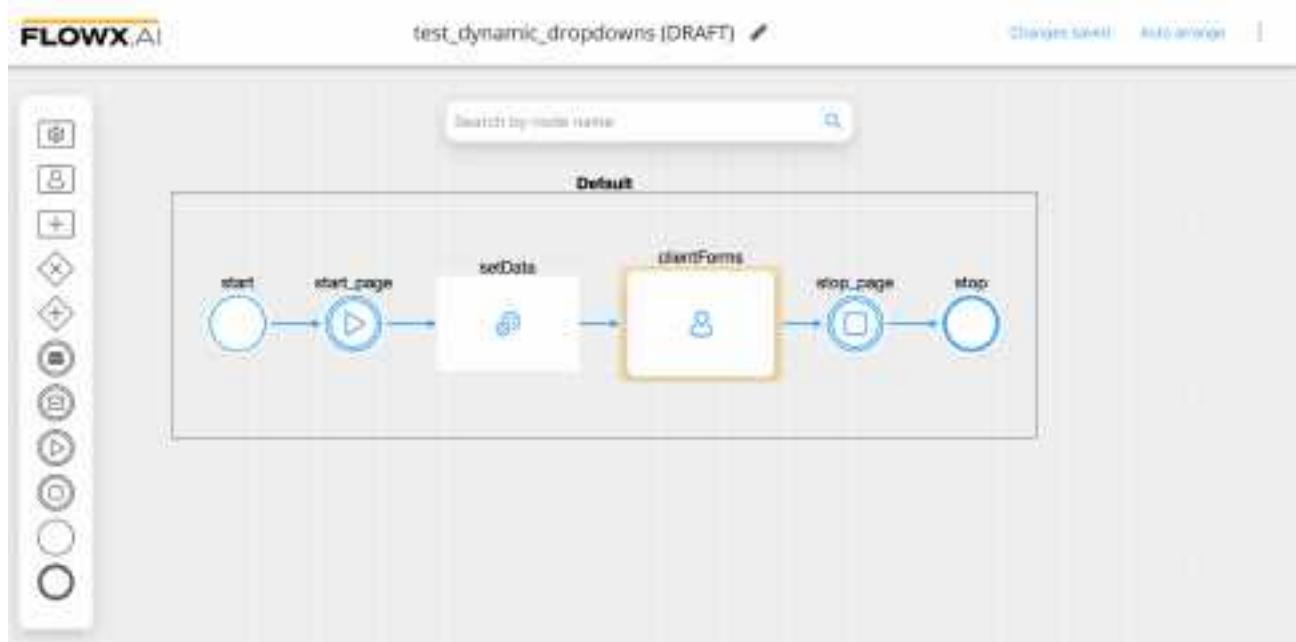
## Creating the process

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:

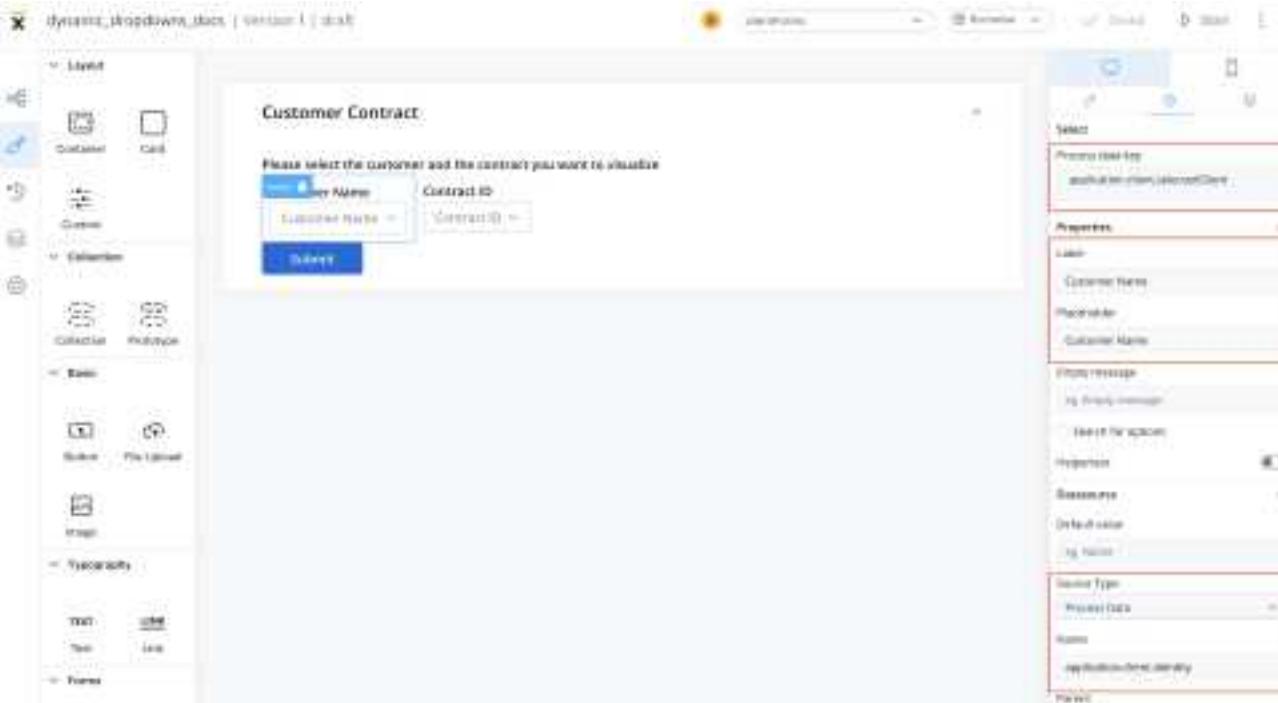
```
output.put("application",
{
    "client": {
        "identity": [
            {
                "value": "001",
                "label": "Eddard Stark"
            },
            {
                "value": "002",
                "label": "Sansa Stark"
            },
            {
                "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

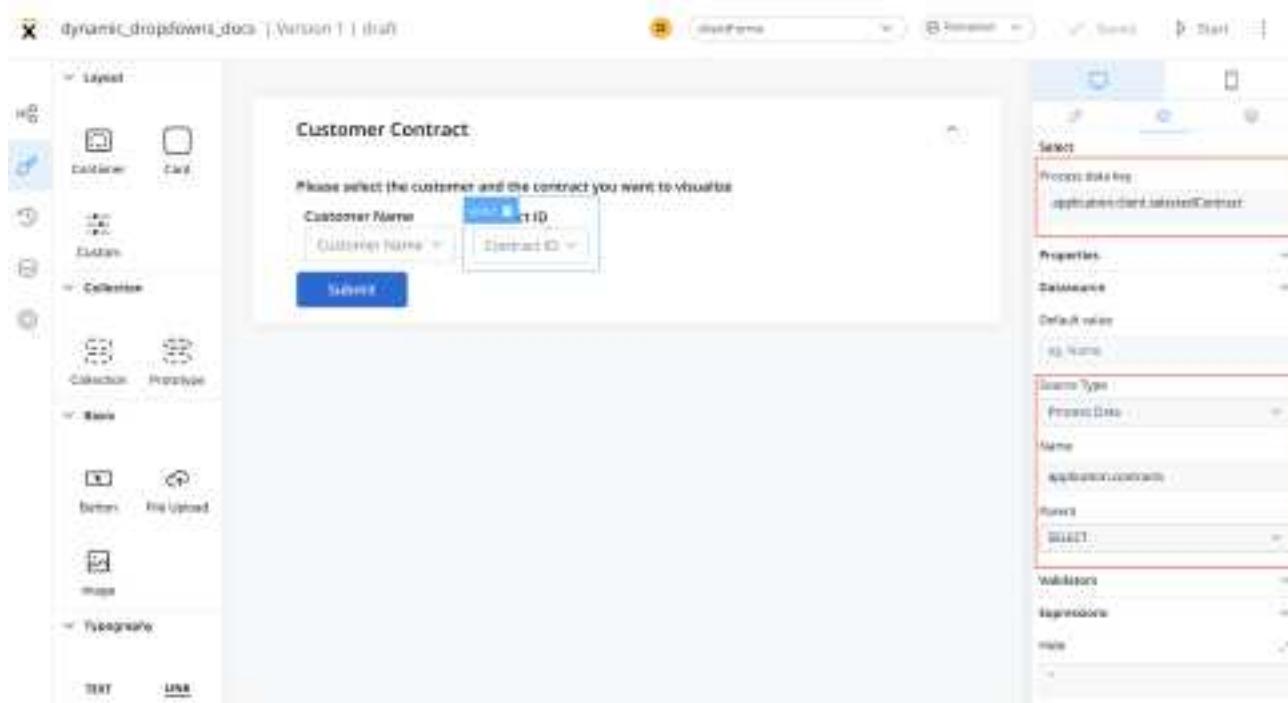
Follow the next steps to configure the UI needed:

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`



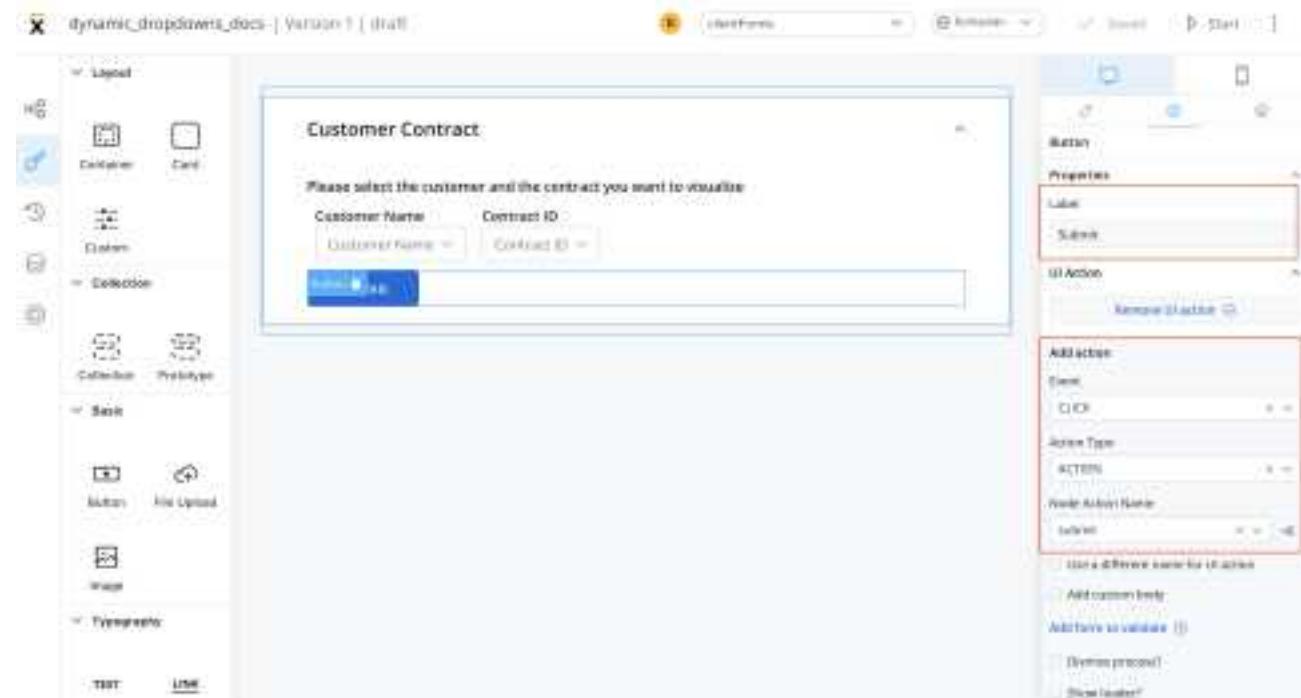
6. For the second select element (Contract ID) set the following properties:

- **Process data key** - application.client.selectedContract
  - **Label** - Contract ID
  - **Placeholder** - Contract ID
  - **Source Type** - Process Data
  - **Name** - application.contracts
  - **Parent** - SELECT (choose from the dropdown list)

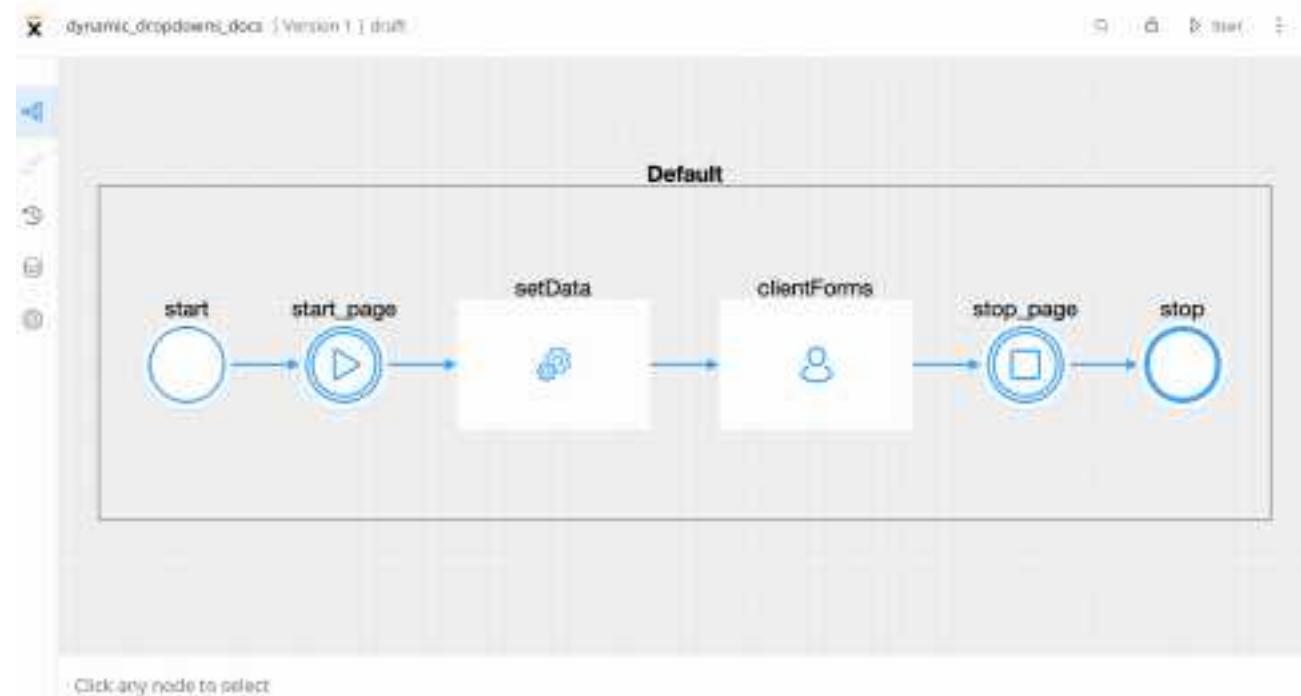


7. Add a button under the form that contains the select elements with the following properties:

- **Label** - Submit
- **Add UI action** - add the submit action attached earlier to the user task node

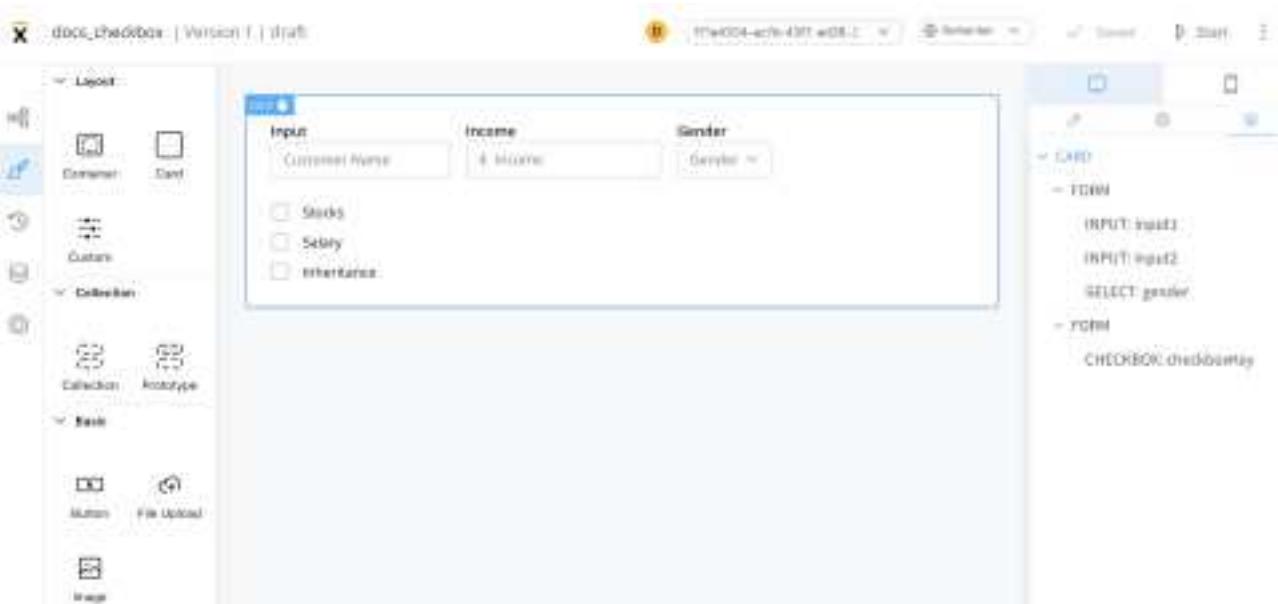


## 8. Test and run the process by clicking **Start process**.



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:

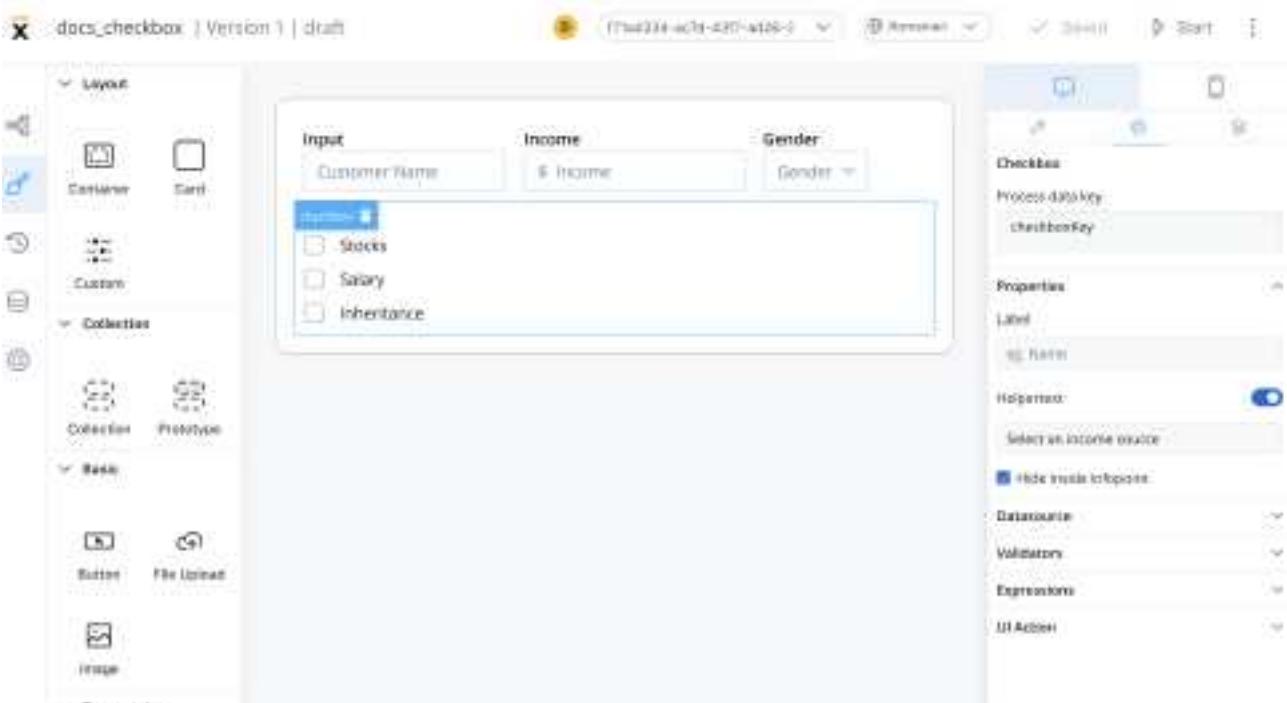
- [General](#)
- [Properties](#)
- [Datasource](#)
- [Validators](#)
- [Expressions](#)
- [UI actions](#)
- [Checkbox styling](#)

### 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)



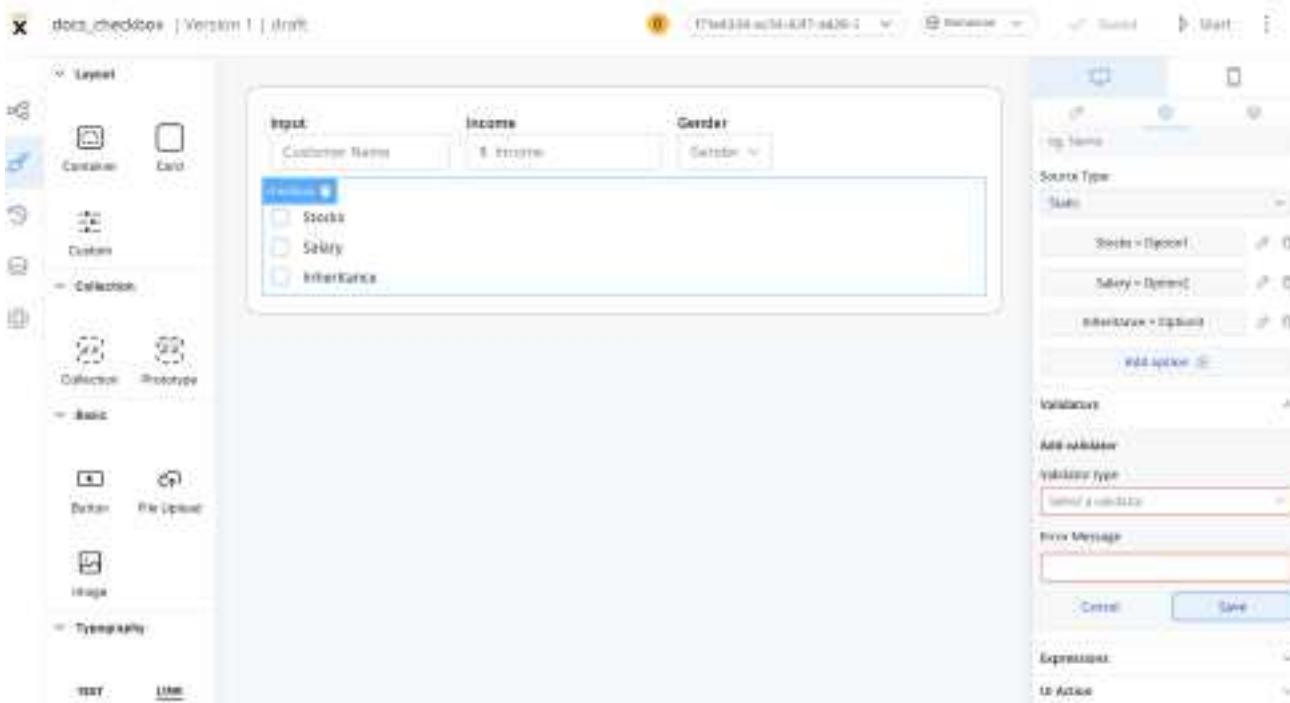
## Datasource

- **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 interface. On the left, there's a toolbar with various components like Container, Card, Collection, and Basic elements. The main workspace displays a form with fields for Customer Name, Income (with dropdown options Stocks, Salary, Inheritance), and Gender. To the right, the Properties panel is open, showing settings for the background, default values, and source type. Under Source Type, three options are listed: Stocks = Option1, Salary = Option2, and Inheritance = Option3. A red box highlights the 'Add option' button. Below this, there are sections for 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



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



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:

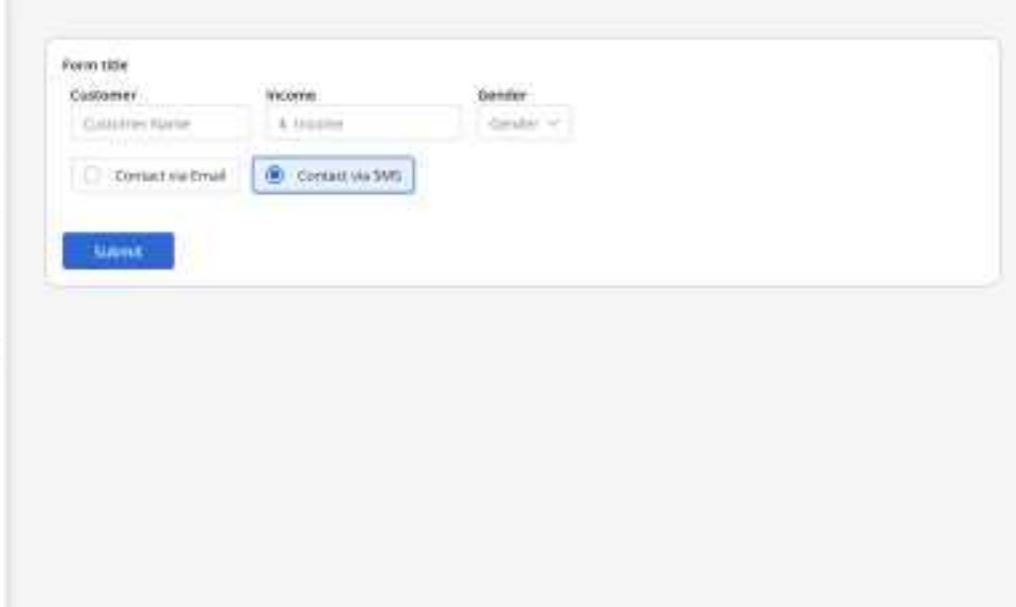
✓ First Form Title  
First Form subtitle

Input 1      Income      Input 2      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 interface. On the left, there's a sidebar with navigation links for 'Processes' (Definitions, Active processes, Process instances, Failed process start) and 'Content Management' (Enumerations, Metadata tags, Content models, Languages, Source systems, Media Library). The main area displays a form titled 'Customer' with fields for 'Income' and 'Gender'. Below these fields is a section for 'Contact' methods. There are two radio buttons: one for 'Contact via Email' (unchecked) and one for 'Contact via SMS' (checked). A blue 'Submit' button is located at the bottom of this section.

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

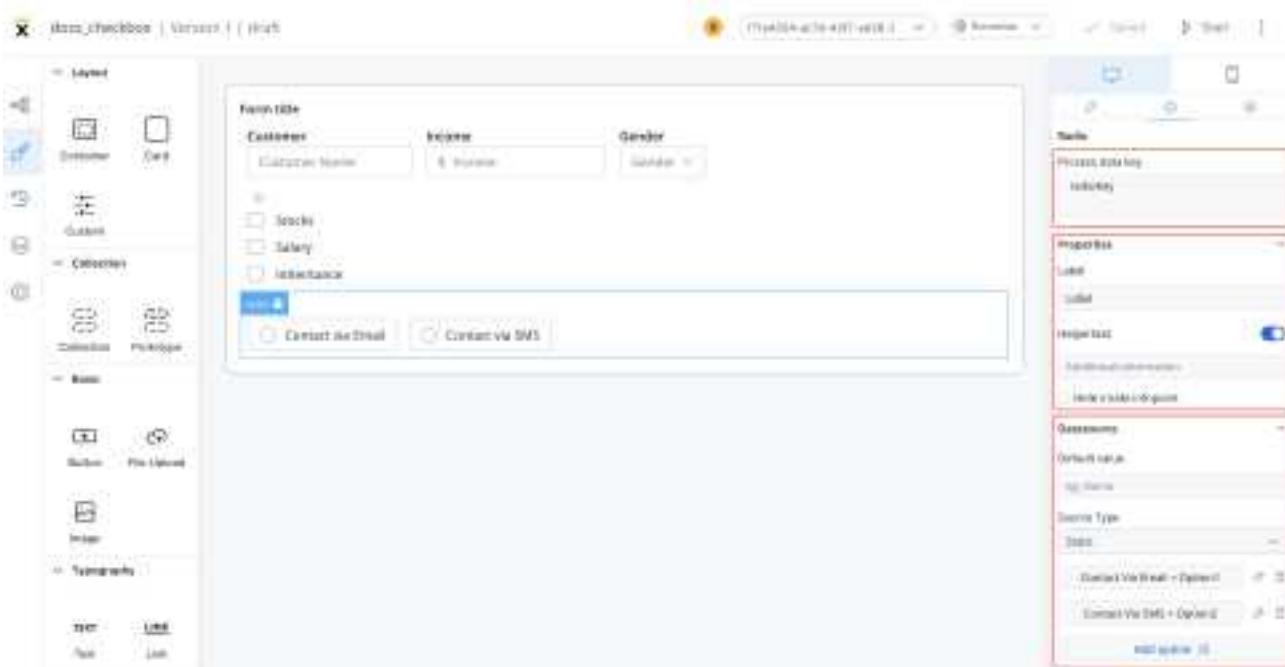
- **Process data key** - creates the binding between form element and process data so it can be later used in decisions, business rules or integrations

#### 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](#))

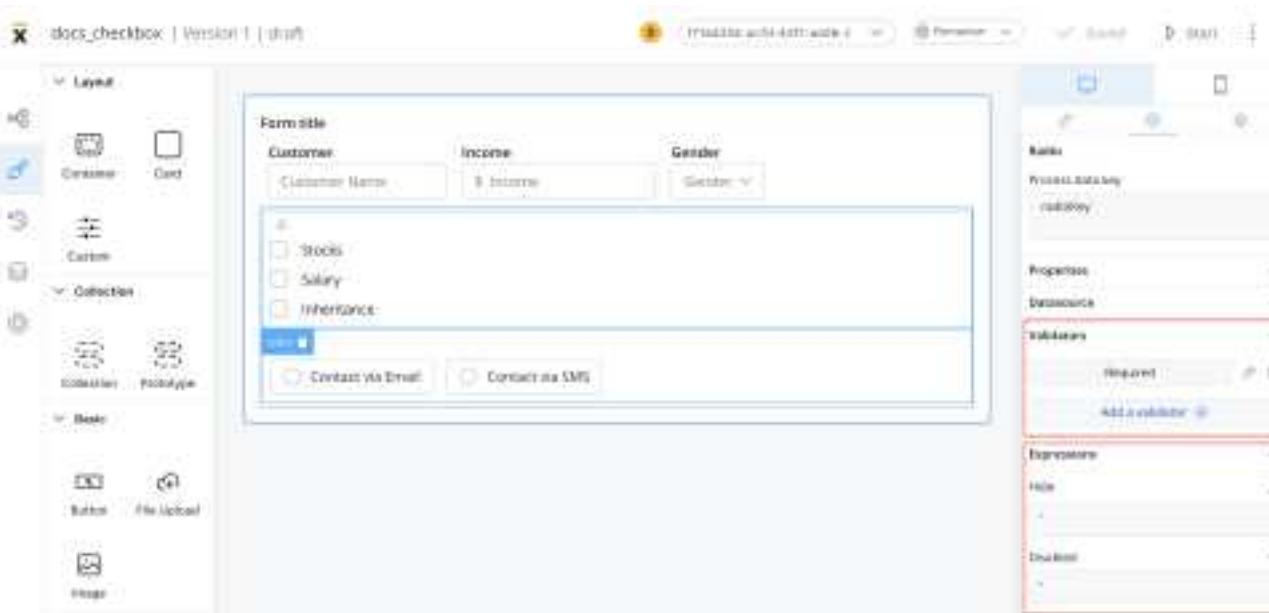
## Expressions

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.



## 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:

The screenshot shows a UI Designer interface for a form titled "First Form Title". The form includes several input fields: "Input 1" (text), "Income" (dropdown menu), "Input 2" (text), "000" (dropdown menu), and "Gender" (dropdown menu). Below these is a radio group with three options: "Stocks" (unchecked), "Salary" (checked), and "Inheritance" (unchecked). At the bottom are two buttons: a green "Contact me!" button with a checked checkbox and a grey "Contact via Email" button with an unchecked checkbox.

**Was this page helpful?**

# BUILDING BLOCKS / UI Designer / UI component types / Form elements / Switch

The image shows a user interface for a form titled "Customer". It includes fields for "Customer" (with value "Customer Customer"), "Income" (with value "\$ 999999999"), and "Gender" (with dropdown menu open). Below these are two radio buttons: "Contact via Email" (selected) and "Contact via SMS". A date picker shows "09.02.1972". A toggle switch labeled "Do you want to subscribe to our newsletter?" is shown in the "on" position. At the bottom is a blue "Submit" button.

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

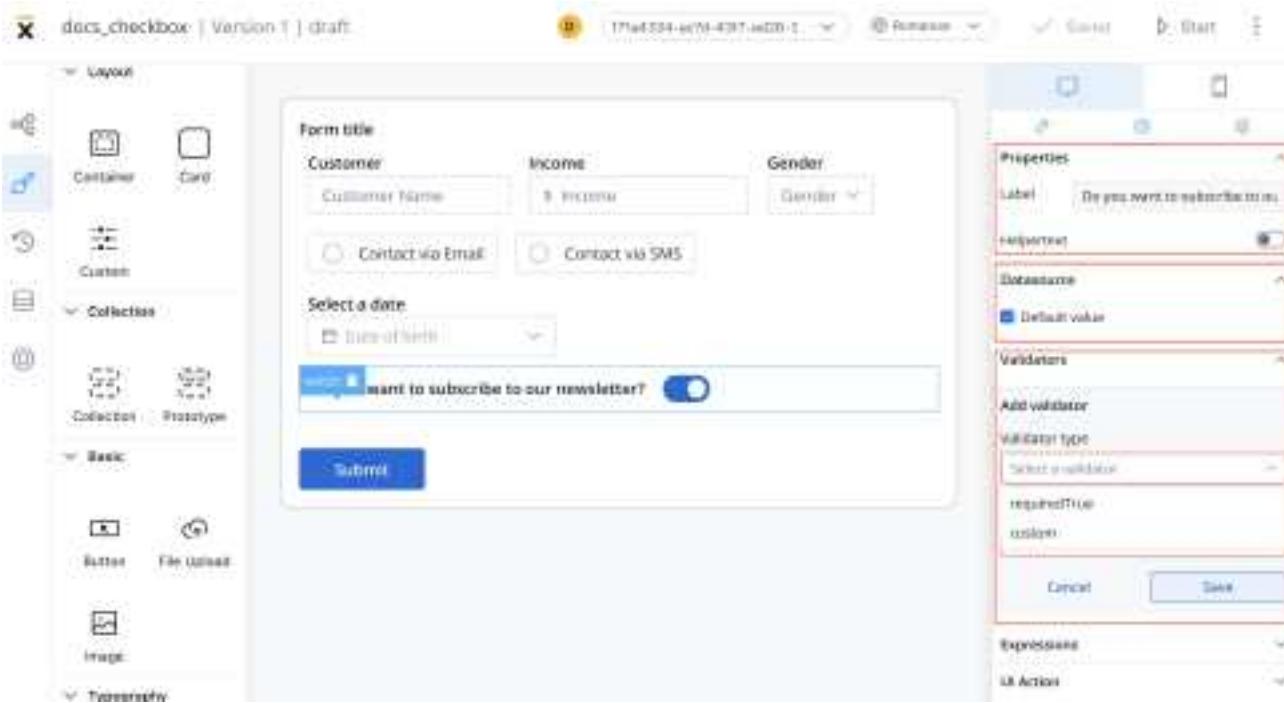
- **Label** - the label of the switch
- **Helpertext** - additional information about the switch element (can be hidden inside an infopoint)

## 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](#)).



## 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.

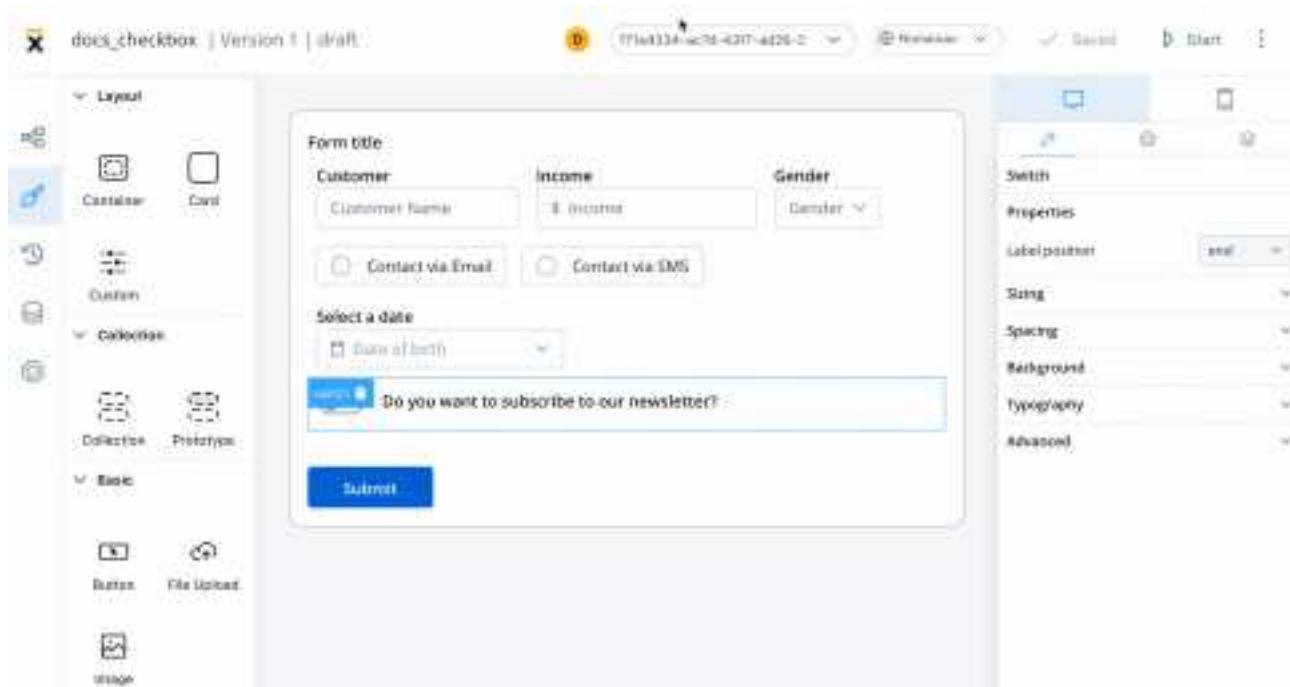
- Event - possible value: CHANGE
- Action Type - select the action type

### INFO

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`.



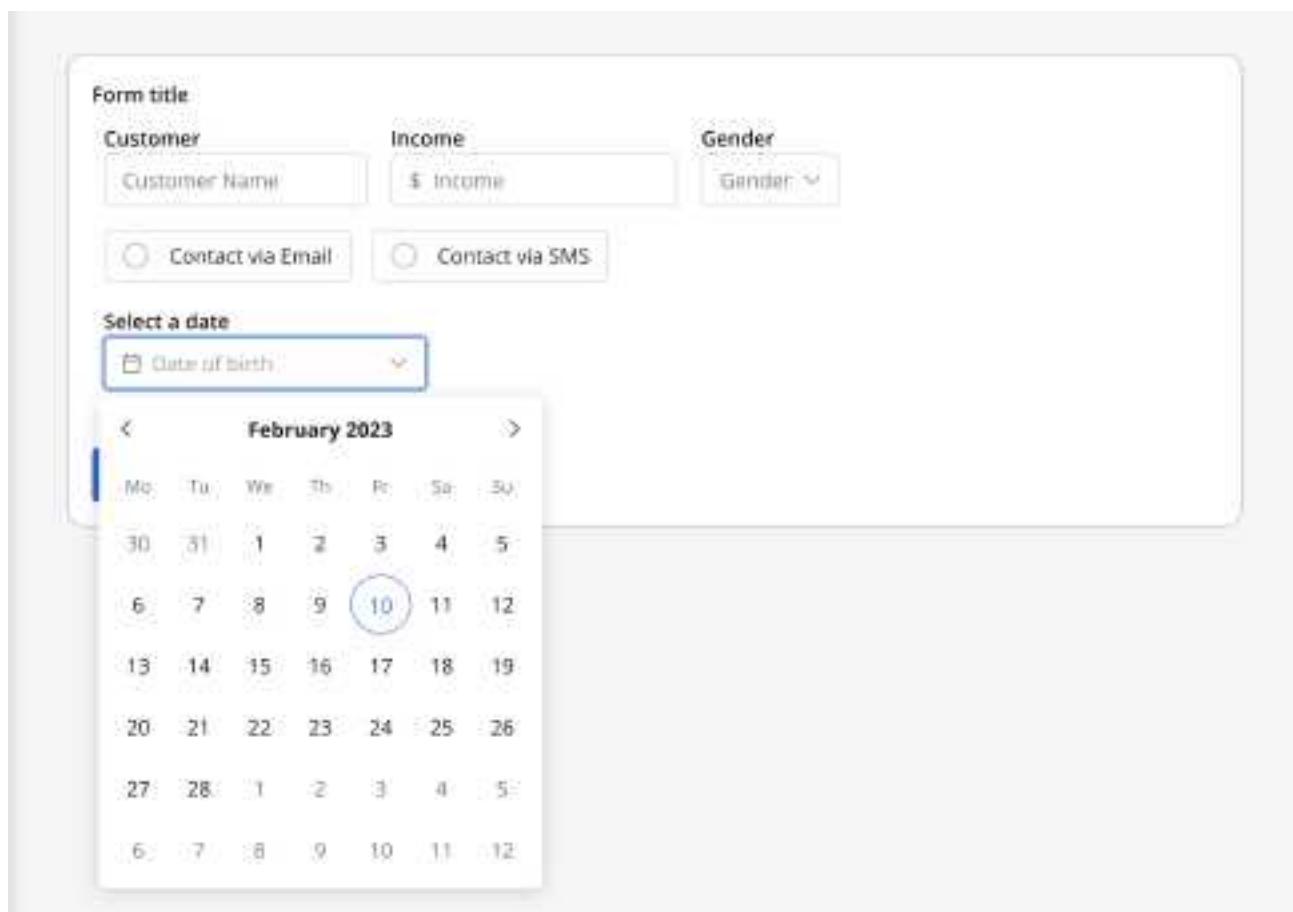
### INFO

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:

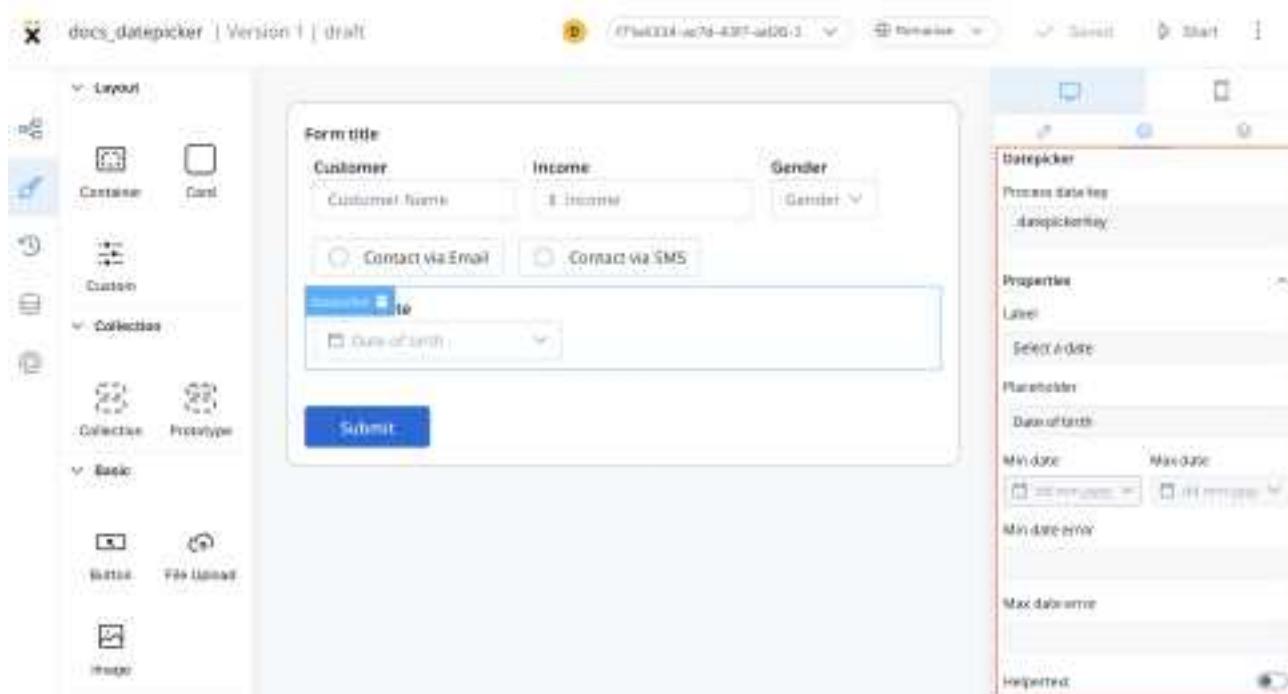
- **General**
- **Properties**
- **Datasource**
- **Validators**
- **Expressions**
- **UI actions**
- **Datepicker styling**

### 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)

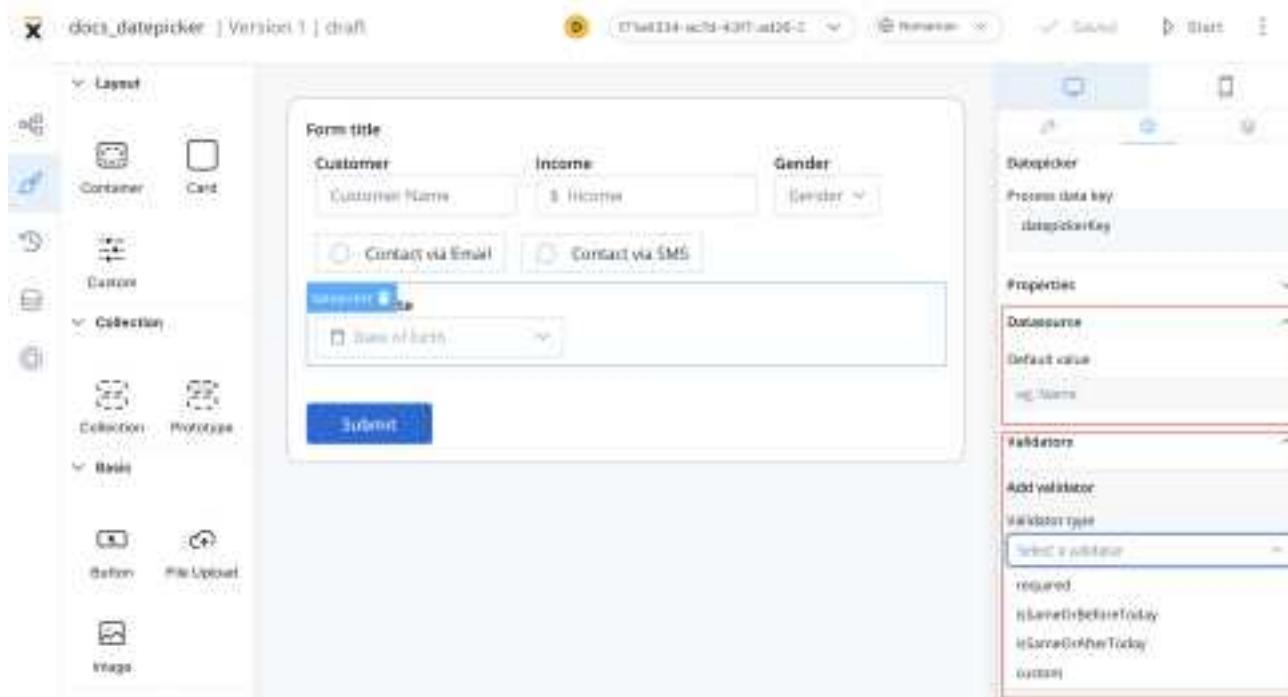


## Datasource

- **Default Value** - the default values of the datepicker element, this will autofill 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](#)).



## 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.

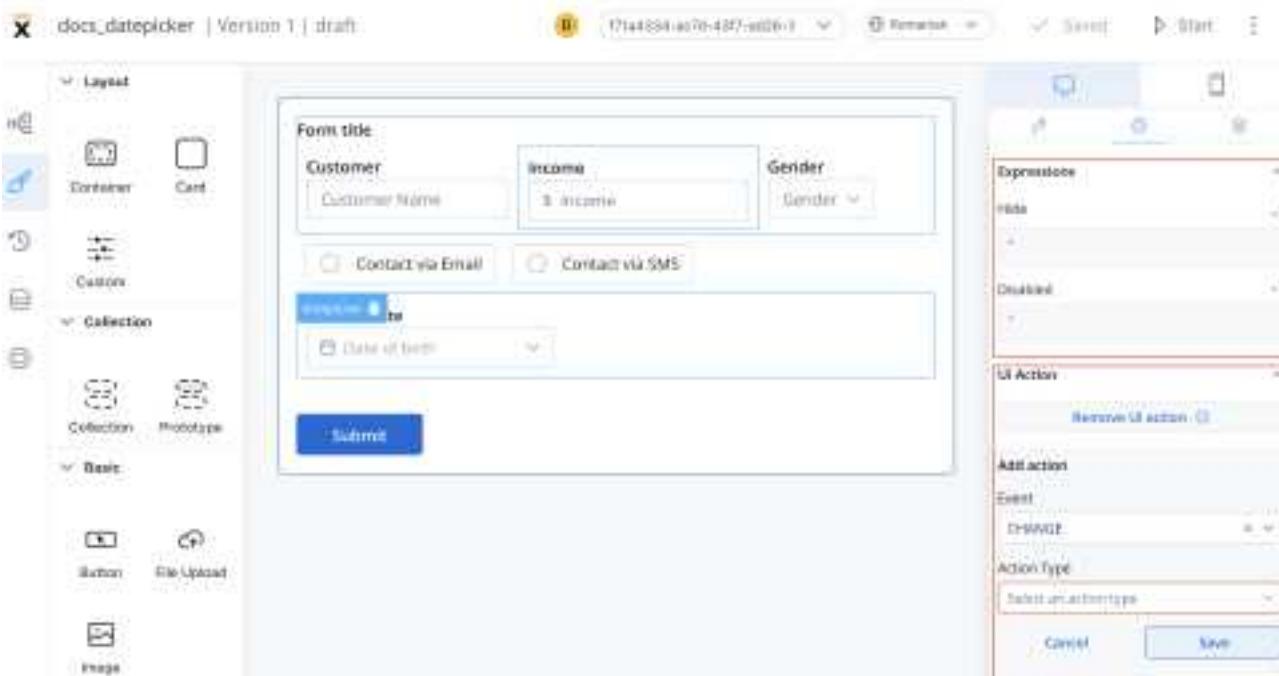
## UI actions

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](#).



## 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 'Form builder' interface with a template for 'Enter Personal Information'. The form fields include:

- Form title: Enter Personal Information
- First Name: John
- Last Name: Doe
- Date of birth: 18.12.1995
- Income range: \$50,000 - \$100,000
- Rate your experience with us: A horizontal slider with values 1 and 5, where the slider is positioned between them.
- Save personal information: A toggle switch that is currently turned on.
- Submit: A blue button at the bottom left.

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

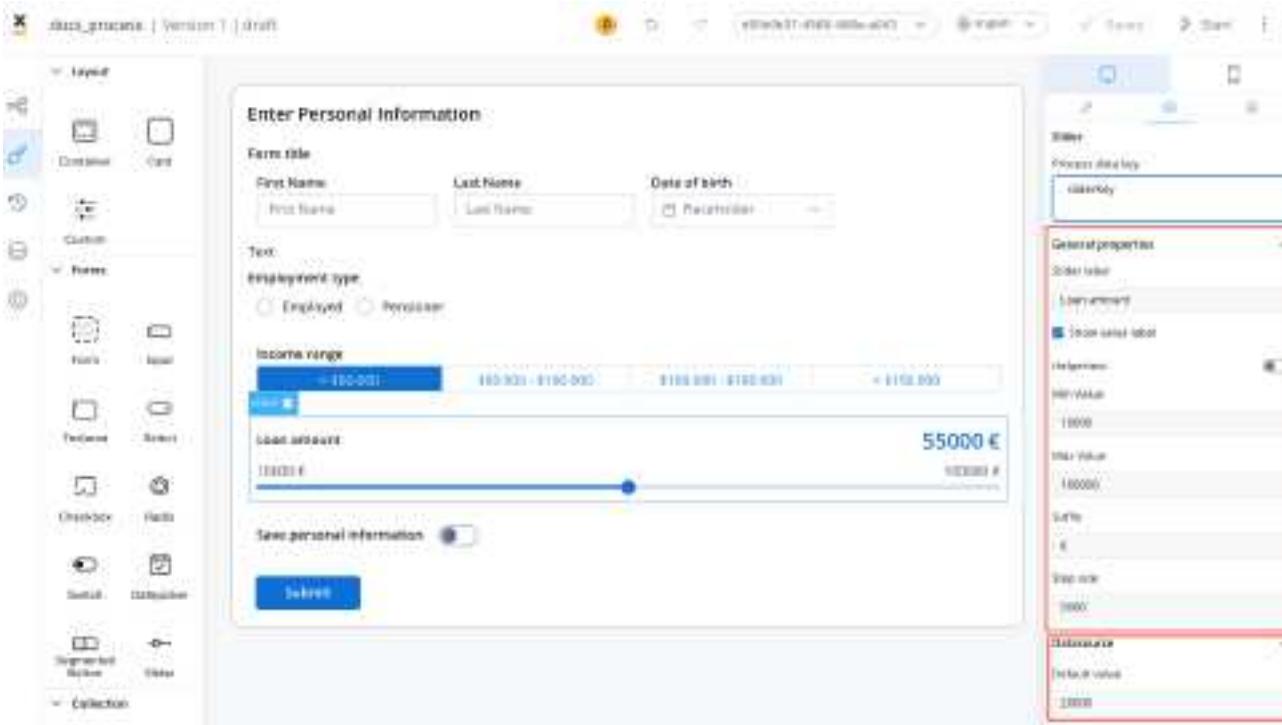
- **Label** - the label of the slider
- **Show value label** - a toggle option that determines whether the current selected value of the slider is displayed as a label alongside the slider handle
- **Helptext** - an optional field that provides additional information or guidance related to the usage 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 form consists of several input fields and interactive elements:

- First Name:** Input field containing "First Name".
- Last Name:** Input field containing "Doe".
- Date of birth:** Input field containing "18.12.1995".
- Employment type:** Two radio buttons: "Employed" (selected) and "Pensioner".
- Save personal information:** A toggle switch that is turned on.
- Loan amount:** A slider with a value of 255000 \$. The input field next to it shows "255000 \$".
- Down payment:** A slider with a value of 38250 \$. The input field next to it shows "38250 \$".
- Form title:** A placeholder text "Form title".
- Loan type:** A dropdown menu showing "Conventional".
- Submit:** A blue button labeled "Submit".

### 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

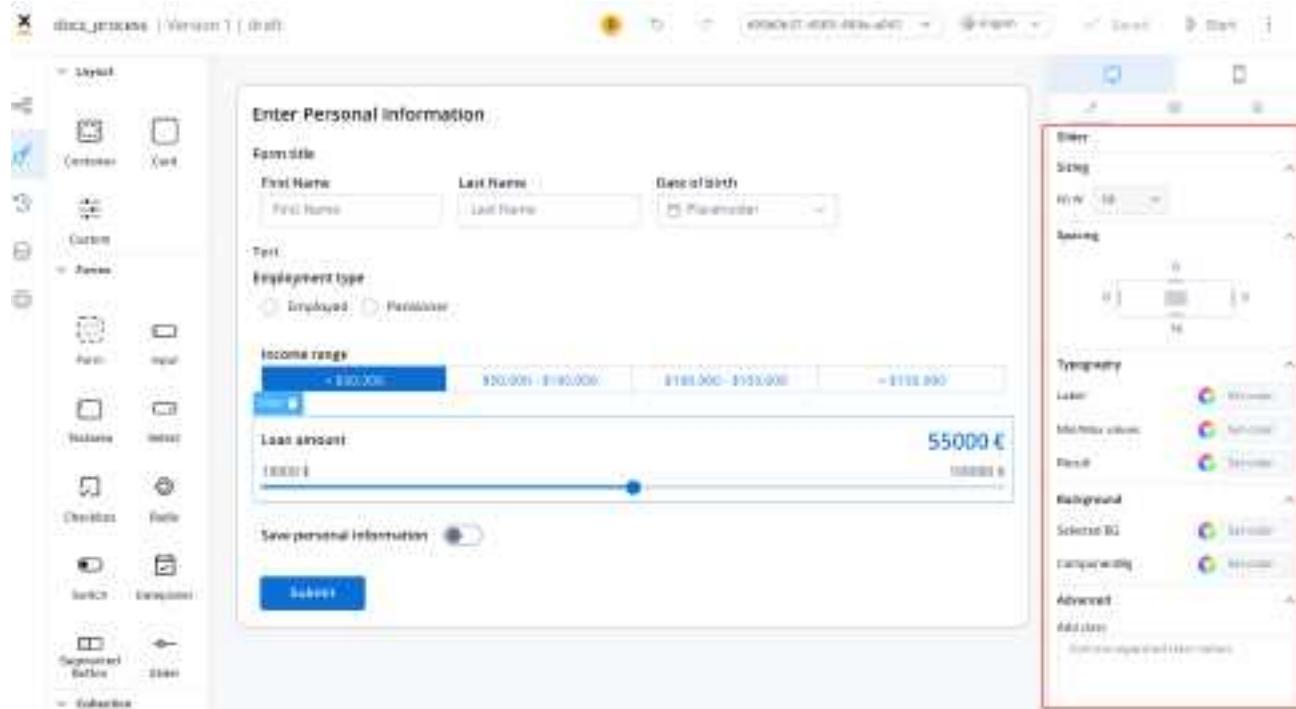
## Typography

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.



### 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 a segmented button containing four options: '< \$50.000', '\$50.000 - \$100.000', '\$100.000 - \$150.000', and '> \$150.000'. There is also a 'Save personal information' toggle switch (turned on) and a '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

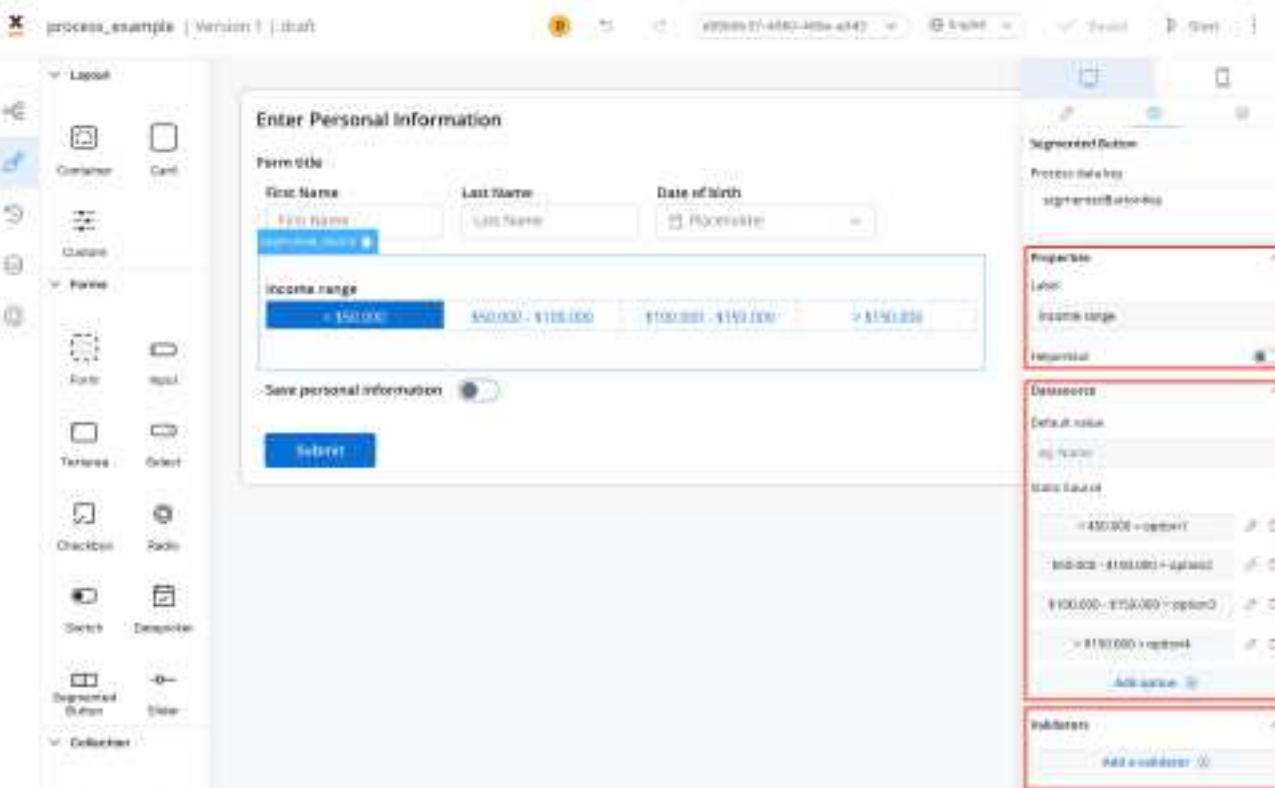
- **Label** - the label of the segmented button
- **Helpertext** - additional information about the segmented button (can be hidden inside an infopoint)

## 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](#)).



## 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

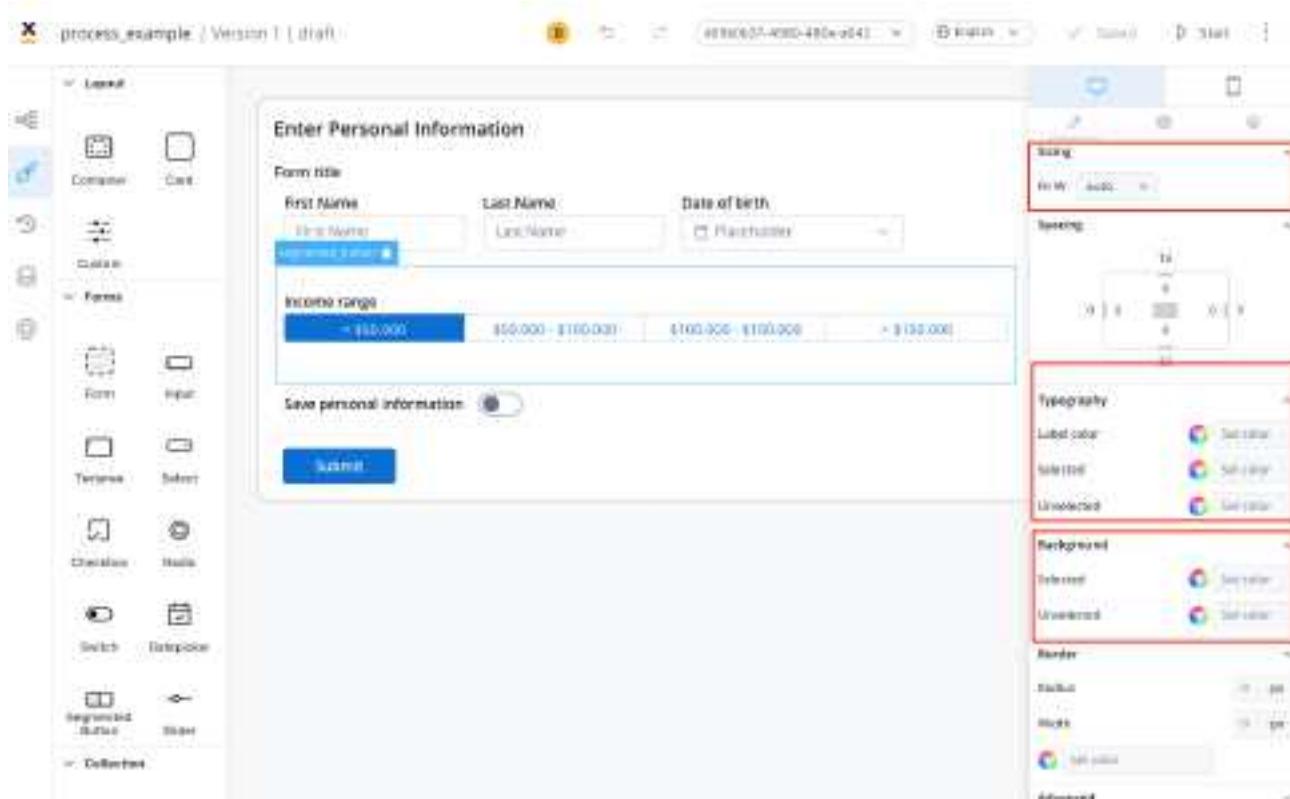
## Typography

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



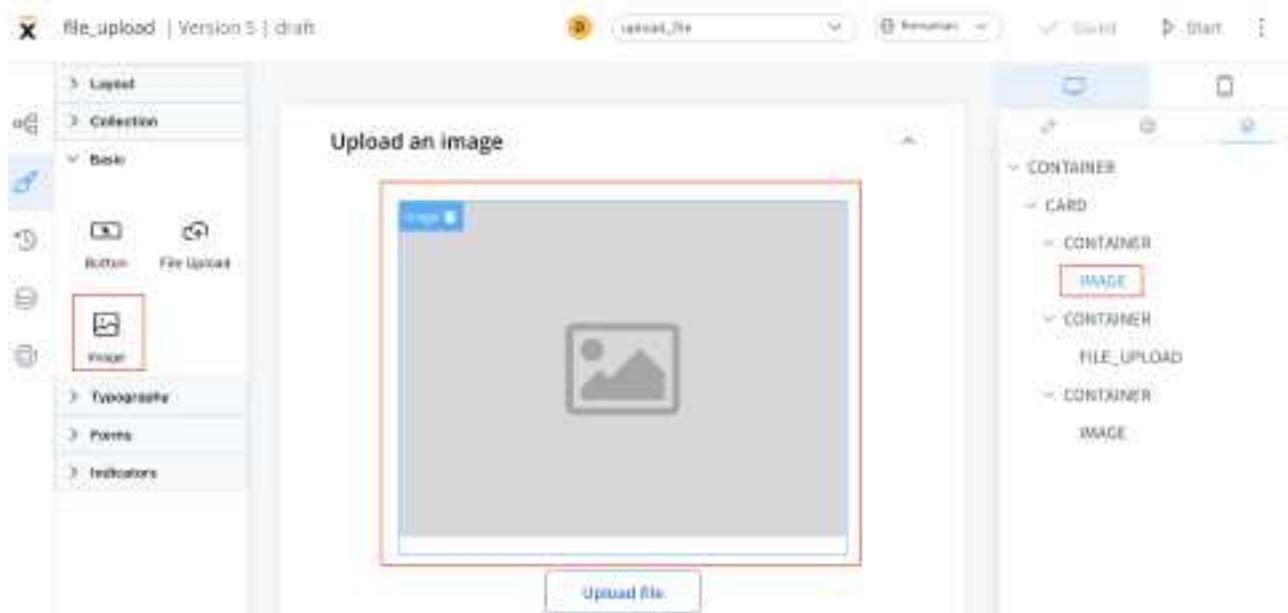
### INFO

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

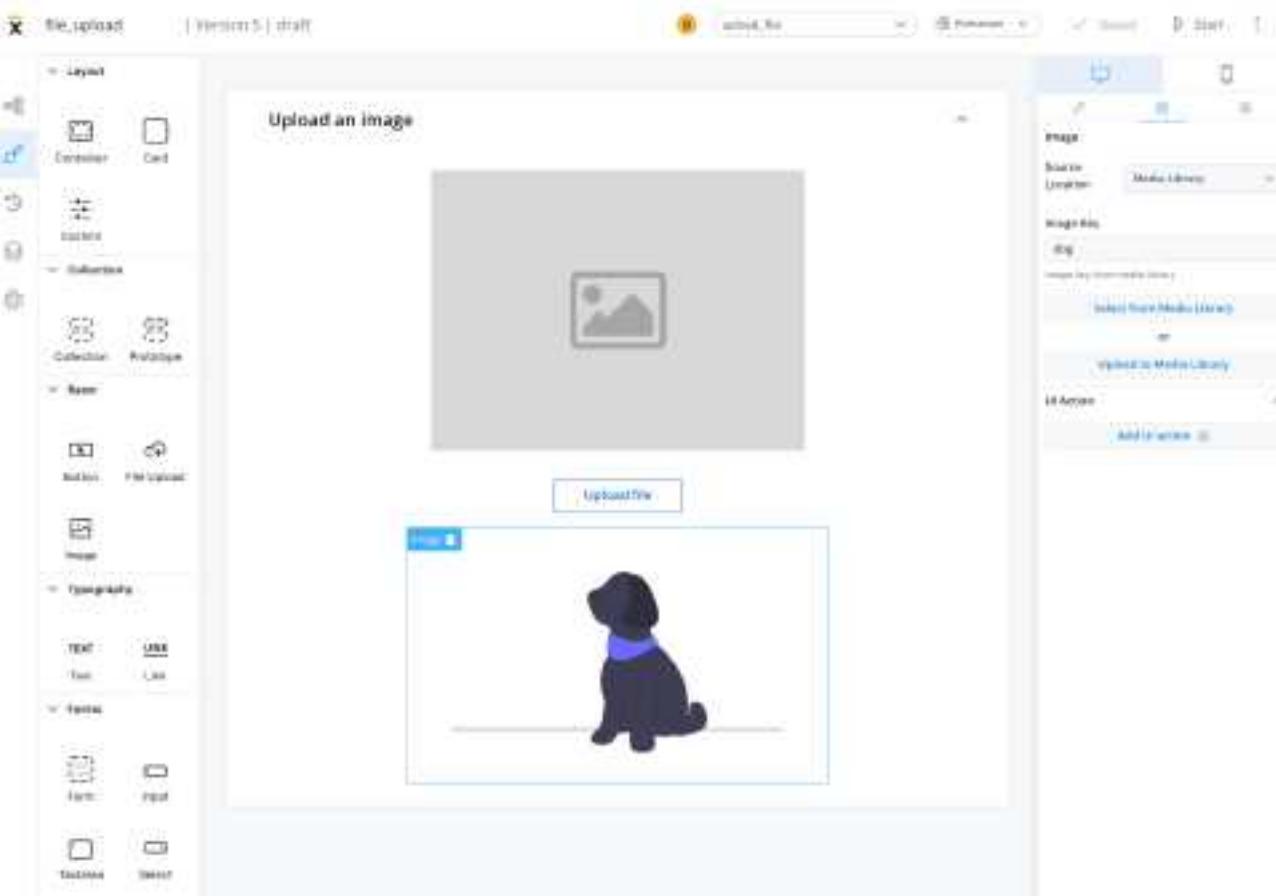
Was this page helpful?

## 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:

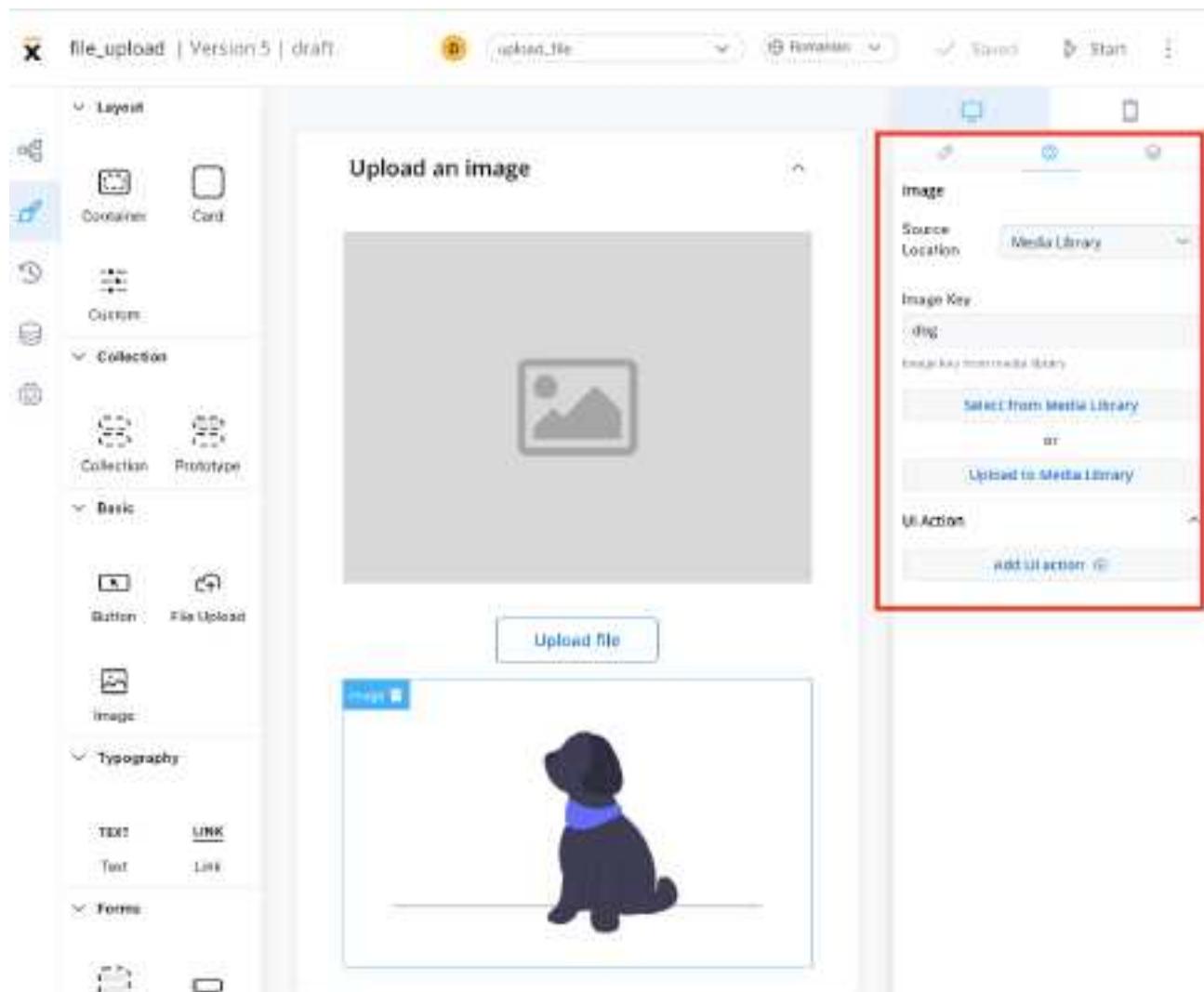
- **Source location** - the location from where the image is loaded:

- **Media Library**

- **Process Data**
- **External**

Depending on which **Source location** is selected, different configurations are available:

## Media library

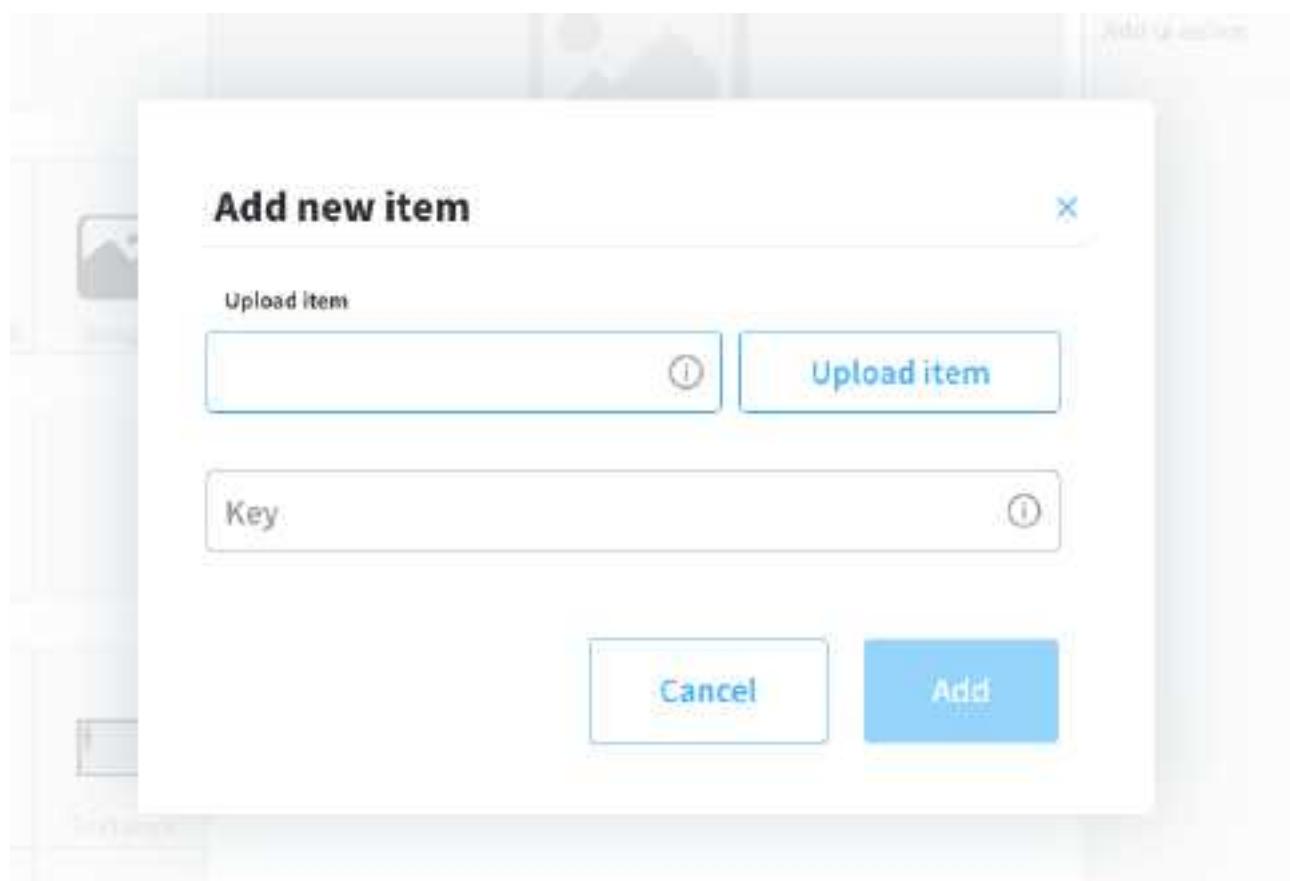


- **Image key** - the key of the image from the media library

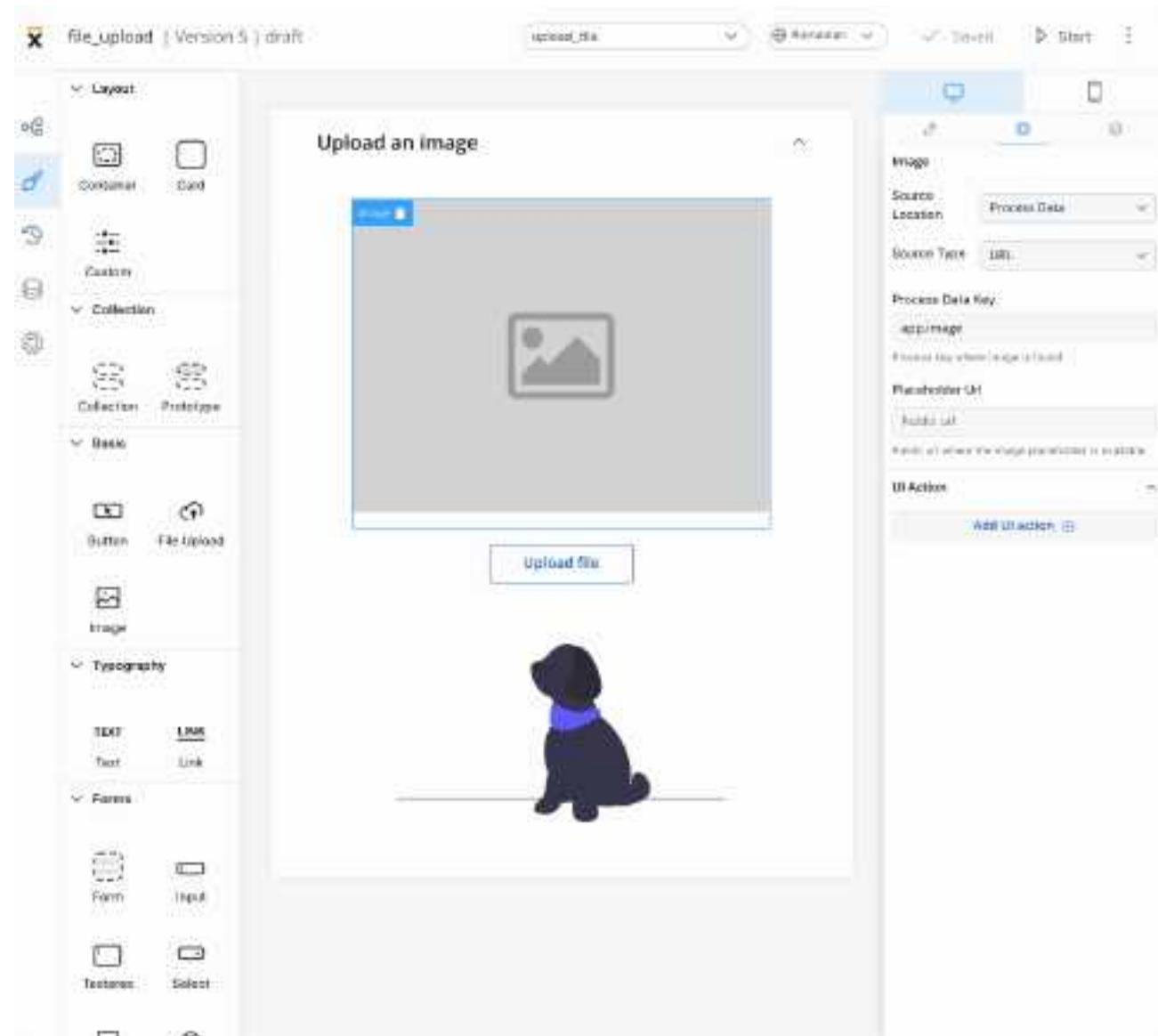
- **Select from media library** - search for an item by key and select it from the media library

| Prefix | Key          | Format | Size     | Created               | Edited   | Author |
|--------|--------------|--------|----------|-----------------------|----------|--------|
|        | video_file   | png    | 30.52 KB | 21 Feb 2023, 11:28 AM | John Doe |        |
|        | switches     | png    | 0.03 MB  | 21 Feb 2023, 11:28 AM | John Doe |        |
|        | form_example | png    | 22.29 KB | 21 Feb 2023, 11:28 AM | John Doe |        |
|        | avatard      | png    | 0.01 MB  | 21 Feb 2023, 11:28 AM | John Doe |        |
|        | avatard      | png    | 23.73 KB | 21 Feb 2023, 11:28 AM | John Doe |        |
|        | dog          | png    | 20.81 KB | 22 Feb 2023, 11:18 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



- 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.

X file\_upload | Version 5 | draft

upload\_file Runname: Saved Start

Layout

Container

Custom

Collection

Collection Prototype

Basic

File Upload

Image

Typography

Text Link

Forms

Upload an image

Process Data

URL

app.Image

Process key when image is loaded

Placeholder URL

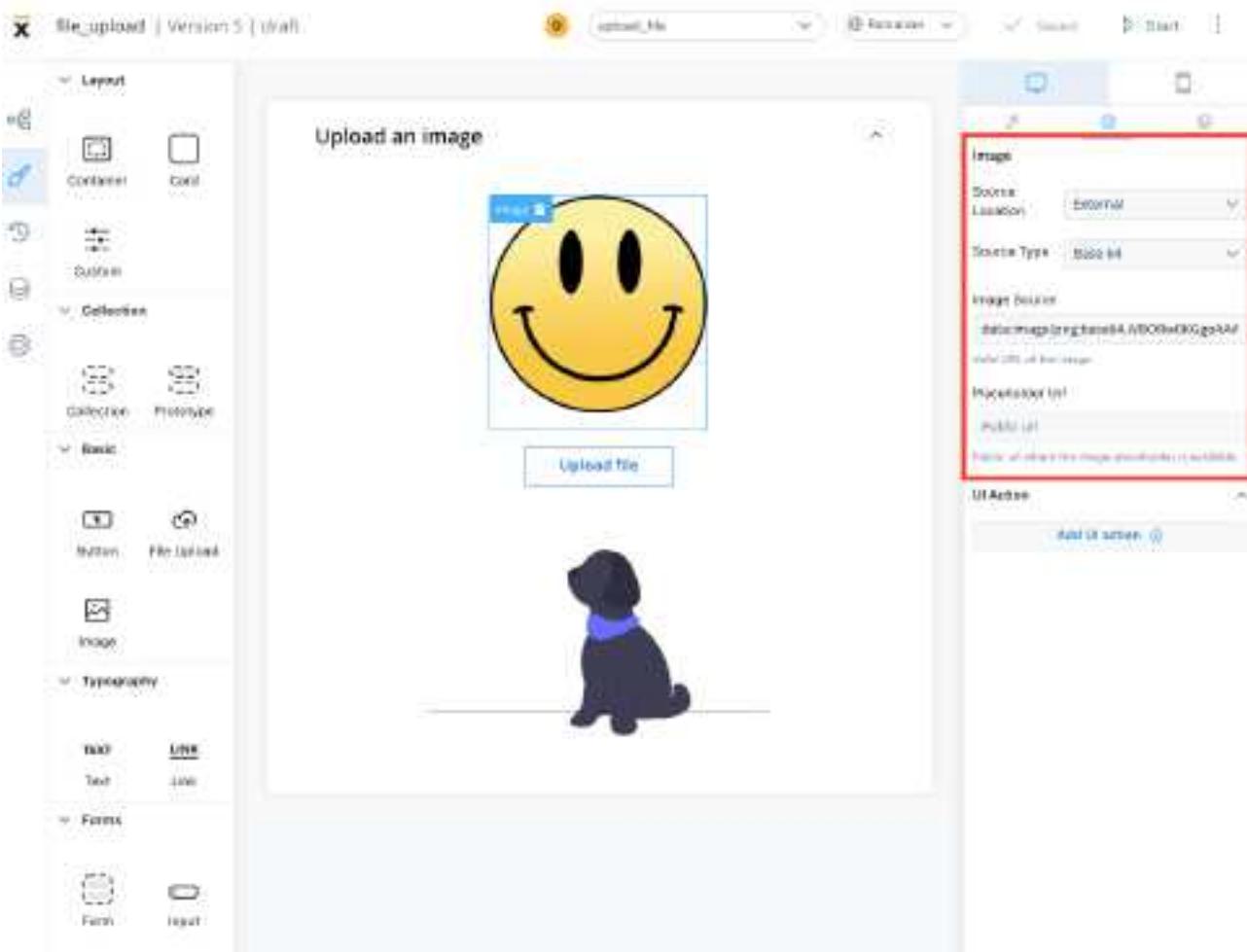
Public URL

Public url where the image placeholder is available

Add UI action

The screenshot displays the FLOWX.AI interface for creating a file upload component. On the left, a sidebar lists various UI elements: Layout, Container, Custom, Collection, Collection Prototype, Basic, File Upload, Image, Typography, Text (selected), Link, and Forms. The main workspace shows a component titled "Upload an image" with a placeholder image of a dog and a "Upload file" button. To the right, a detailed configuration panel is open, showing fields for "Source Location" (set to "Process Data"), "Source Type" (set to "URL"), "Process Data Key" (set to "app.Image"), and "Placeholder URL". A note indicates that the placeholder URL is used when the image is loaded. A "Public URL" field is also present, with a note stating it's where the image placeholder is available. A "UI Action" section at the bottom right allows for adding additional actions.

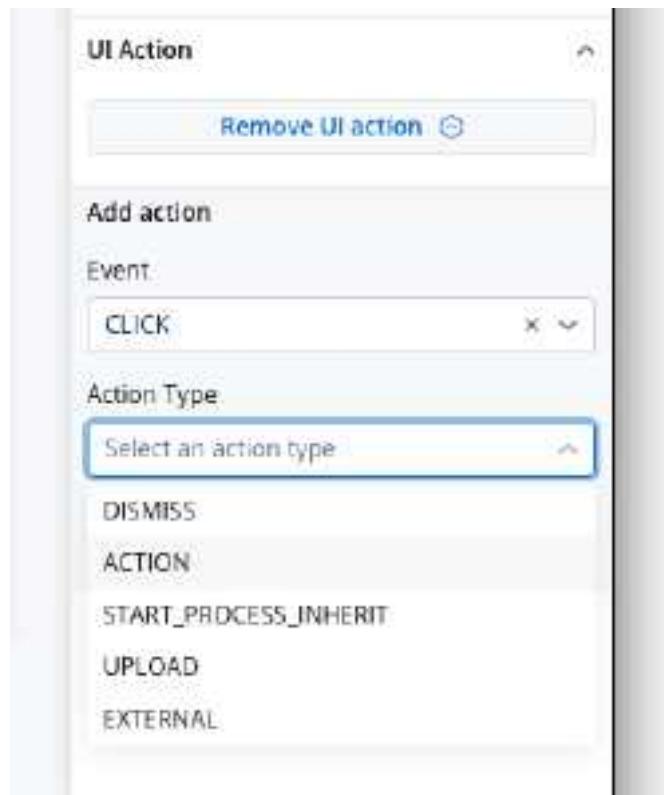
## External



- **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

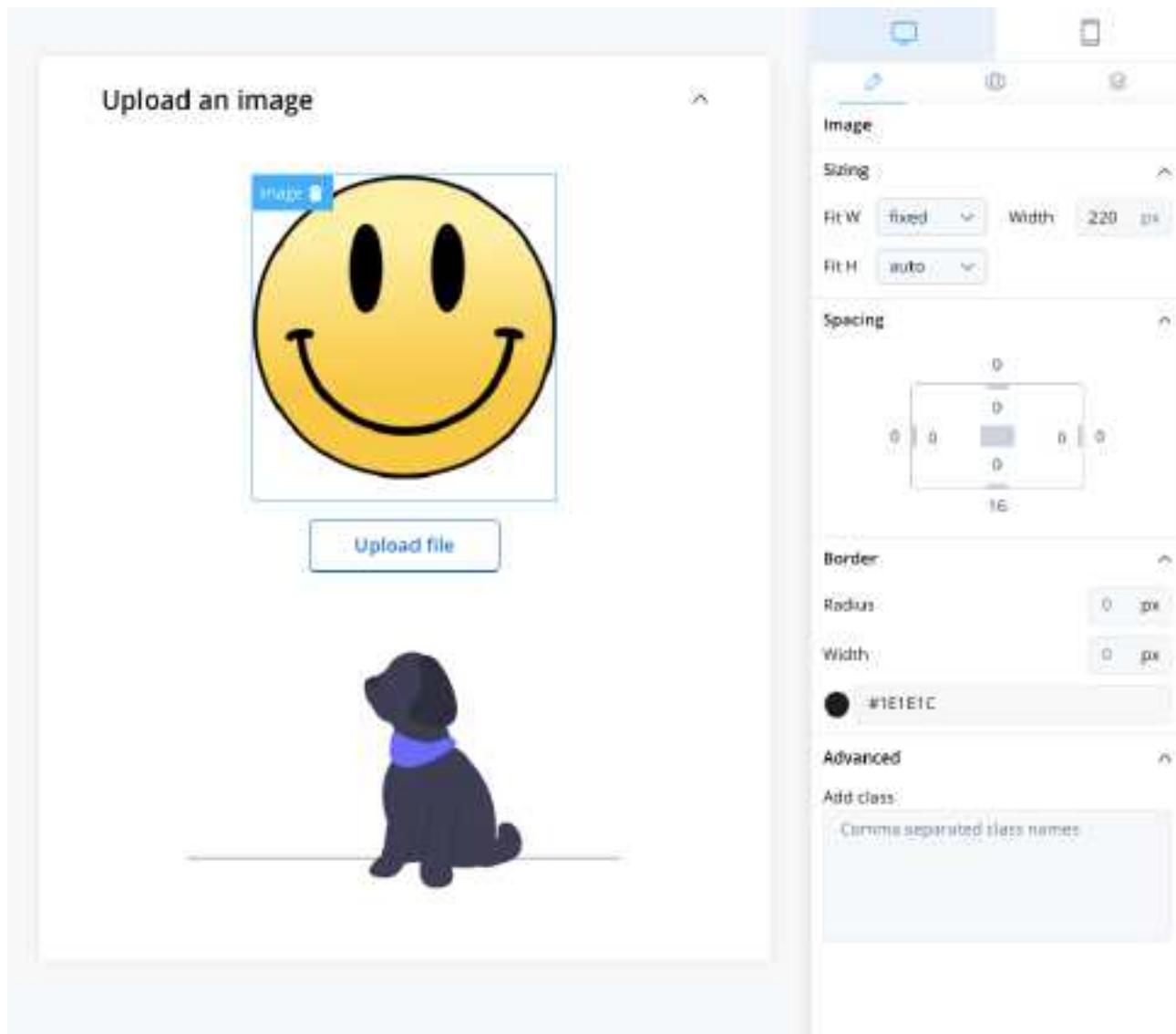
## UI actions

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](#).



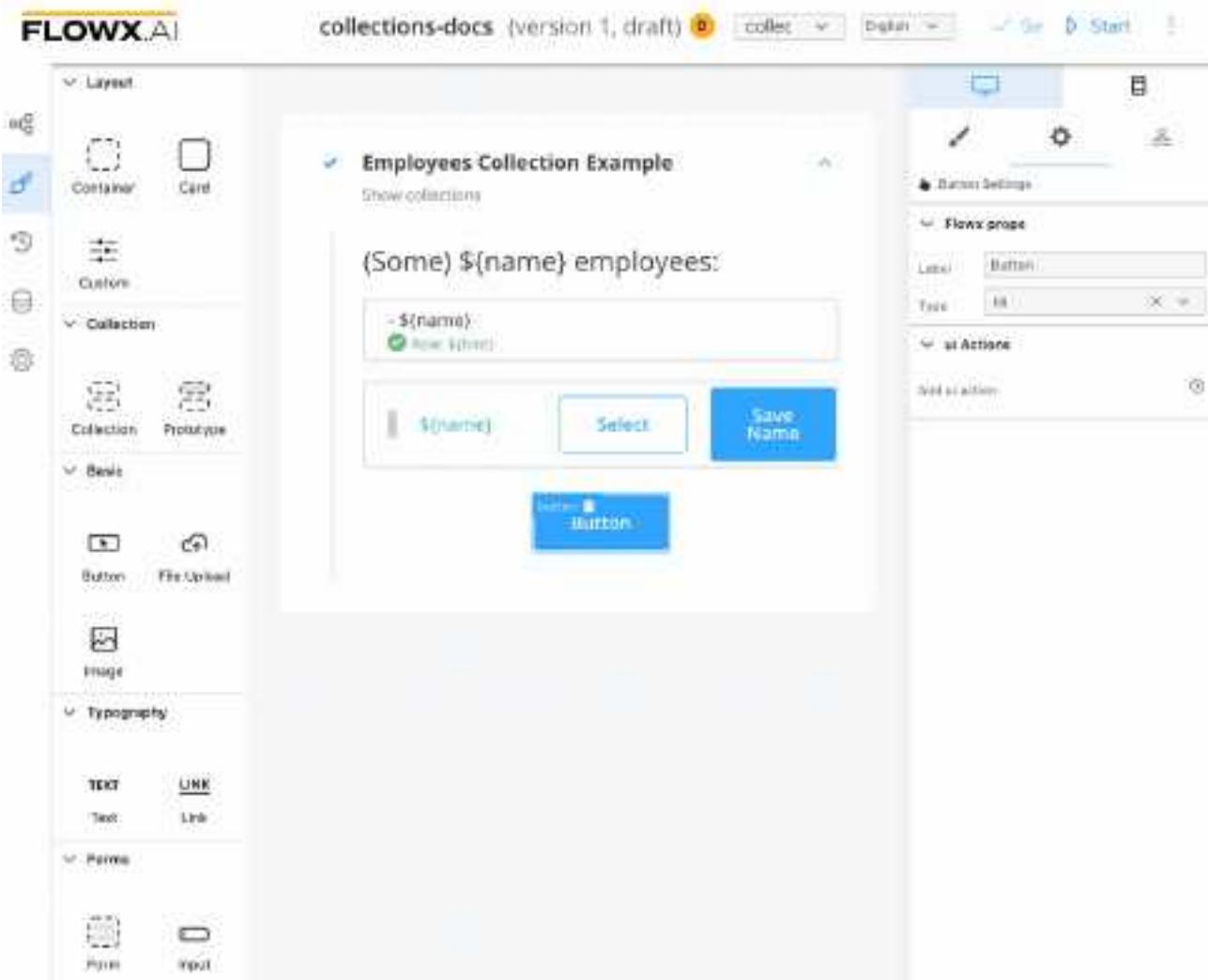
Was this page helpful?

## 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: CLICK

Type: ACTION

Node Action Name: saveUIaction

Use a different name for UI action

Custom body:

Forms to validate: FORM

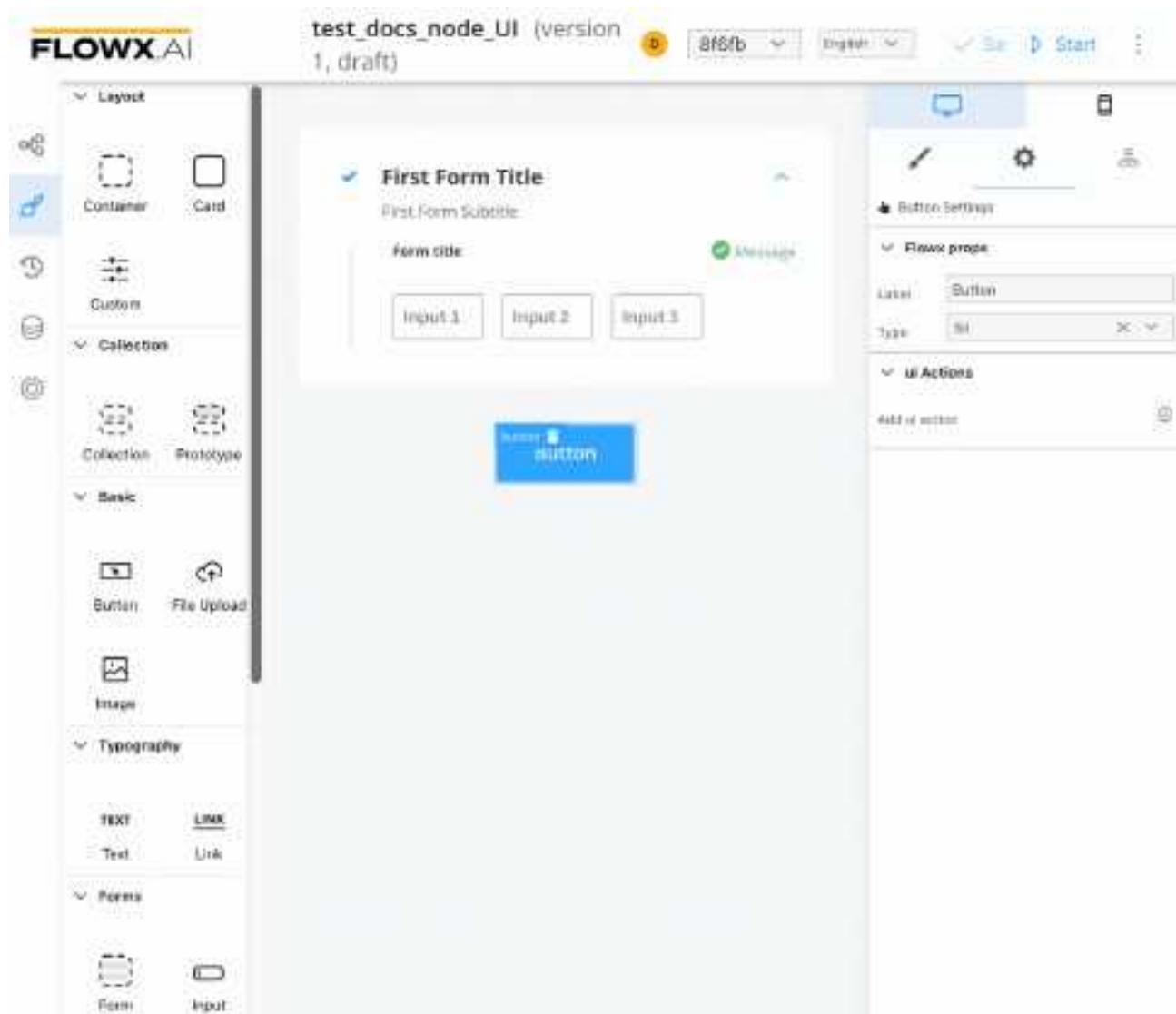
Dismiss process?  Show loader?

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**.



### INFO

! Not available for *UI Actions* on Custom Components.

## Action types

The **action type** dropdown will be pre-filled with the following UI action types:

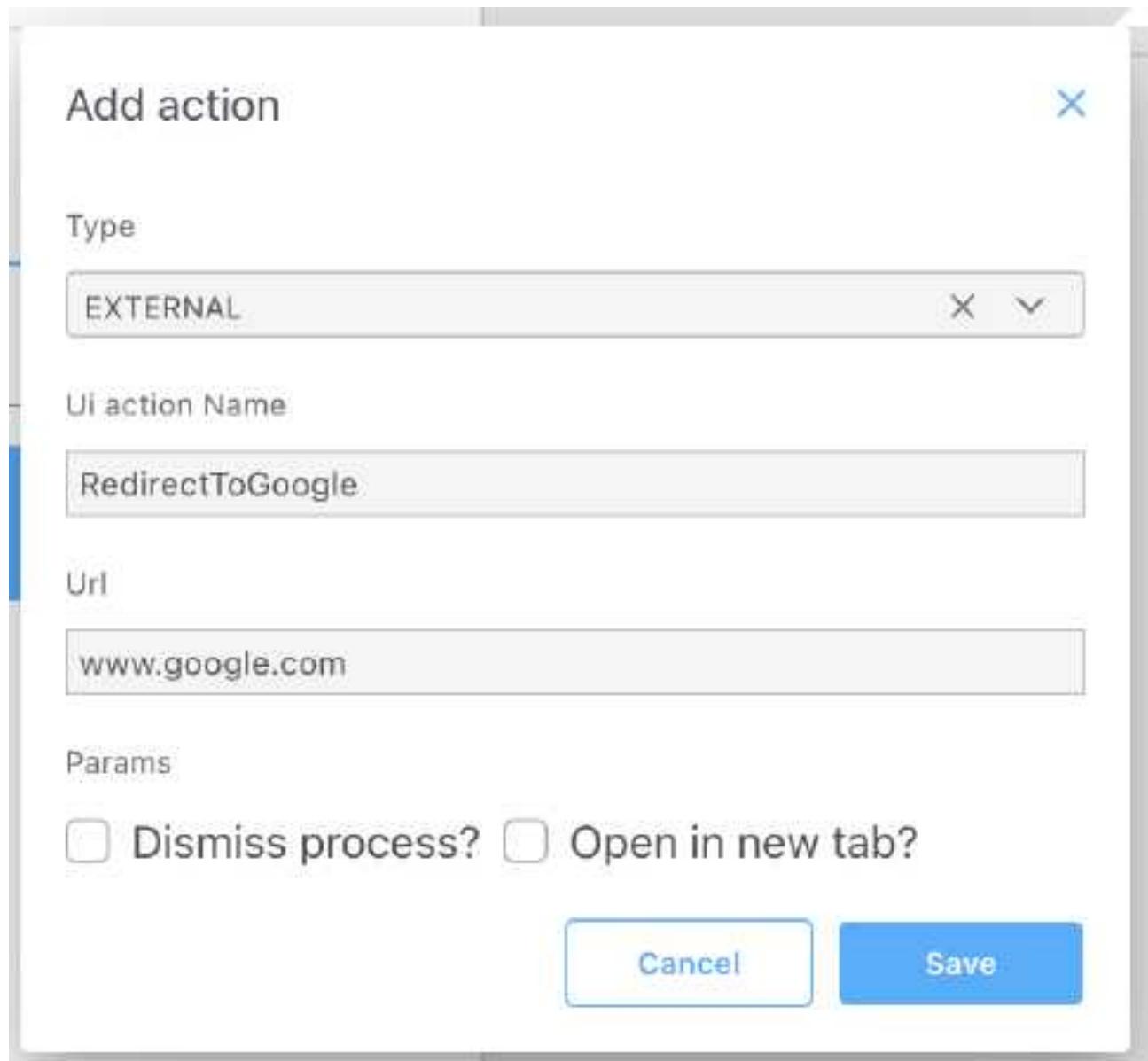
- DISMISS - used to dismiss a modal after action execution
- ACTION - used to link an action to a UI action
- START\_PROCESS\_INHERIT - used to inherit data from another process
- UPLOAD - used to create an un upload action
- EXTERNAL - used to create an action that will open a link in a new tab

## 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



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's form builder. On the left, there's a sidebar with categories like Layout, Collection, Basic, Typography, and Forms, each with corresponding icons. In the center, there's a preview area showing a simple form with a title input field containing 'abc' and a placeholder 'Placeholder', a label 'Input label' with a placeholder 'Placeholder', and a blue 'Submit' button. On the right, there's a panel for configuring form elements like suffixes, helpers, and data sources. A dropdown menu is open under the 'Validators' section, titled 'Add validator'. It lists several options: required, minlength, maxlength, min, max, email, pattern, and custom. The 'Select a validator' option is highlighted with a red border.

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.

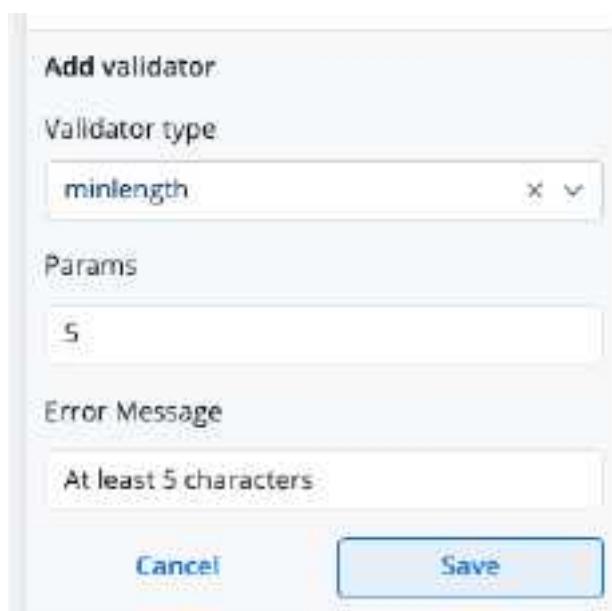
The screenshot shows the FLOWX.AI interface for building forms. On the left is a sidebar with categories like Layout, Collection, Basic, and Typography. The main area shows a form with a title input, two input fields, and a submit button. The properties panel on the right has sections for Type, Prefix, Suffix, HelpText, DataSource, Default value, Validators, Expressions, Hide, and Disabled. The 'Validators' section is highlighted with a red box and contains 'Required' and 'Minlength' validators.

It is recommended to use this validator with other validators like **minlength** to check if there is no value at all.



## 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.



## 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

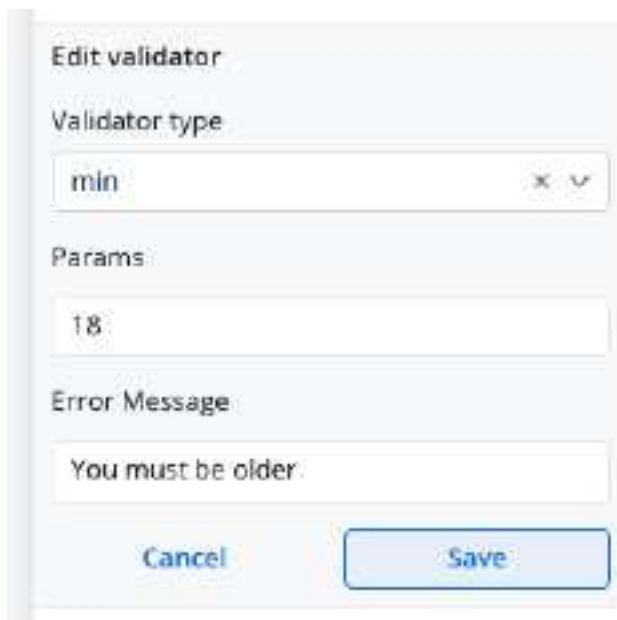
Params

18

Error Message

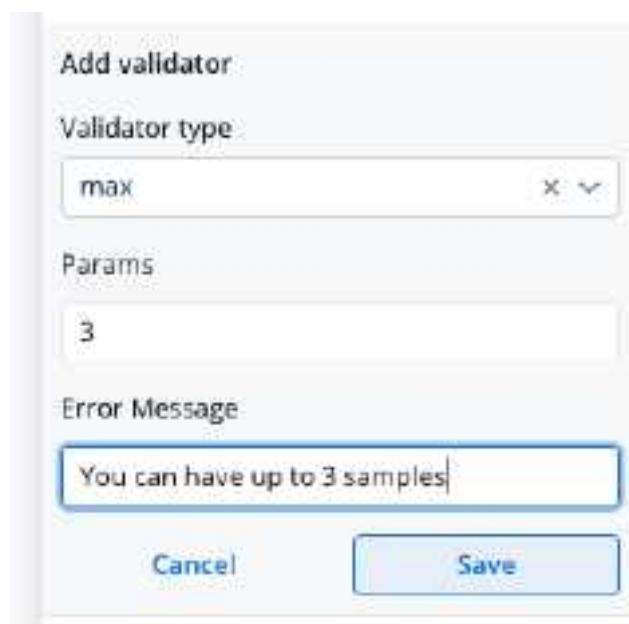
You must be older.

Cancel Save



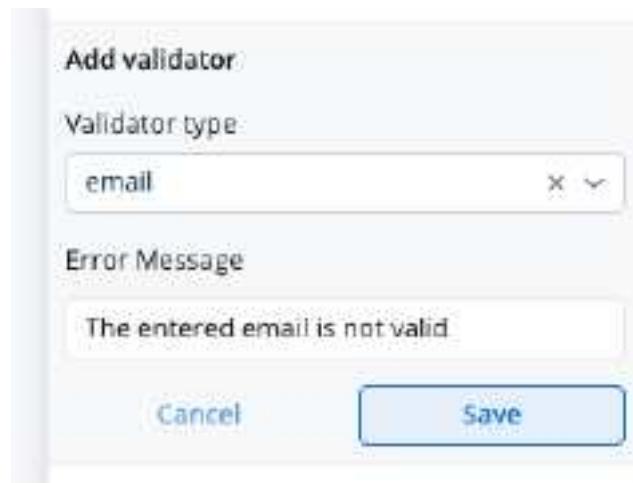
## max validator

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.



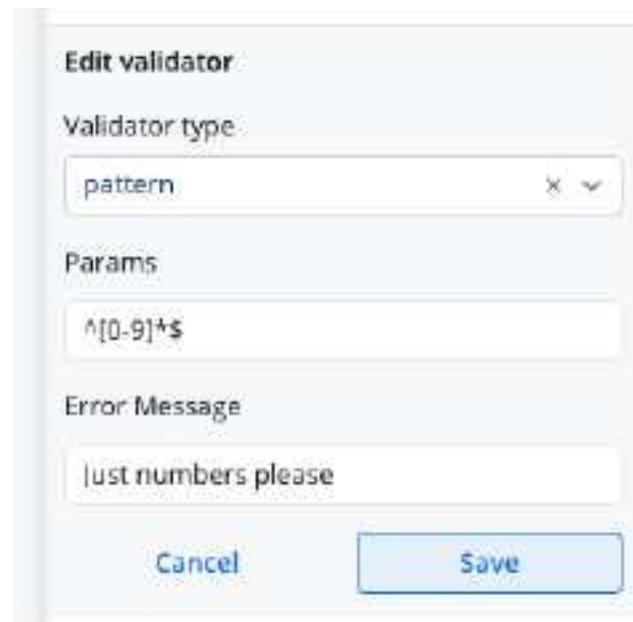
## email validator

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.



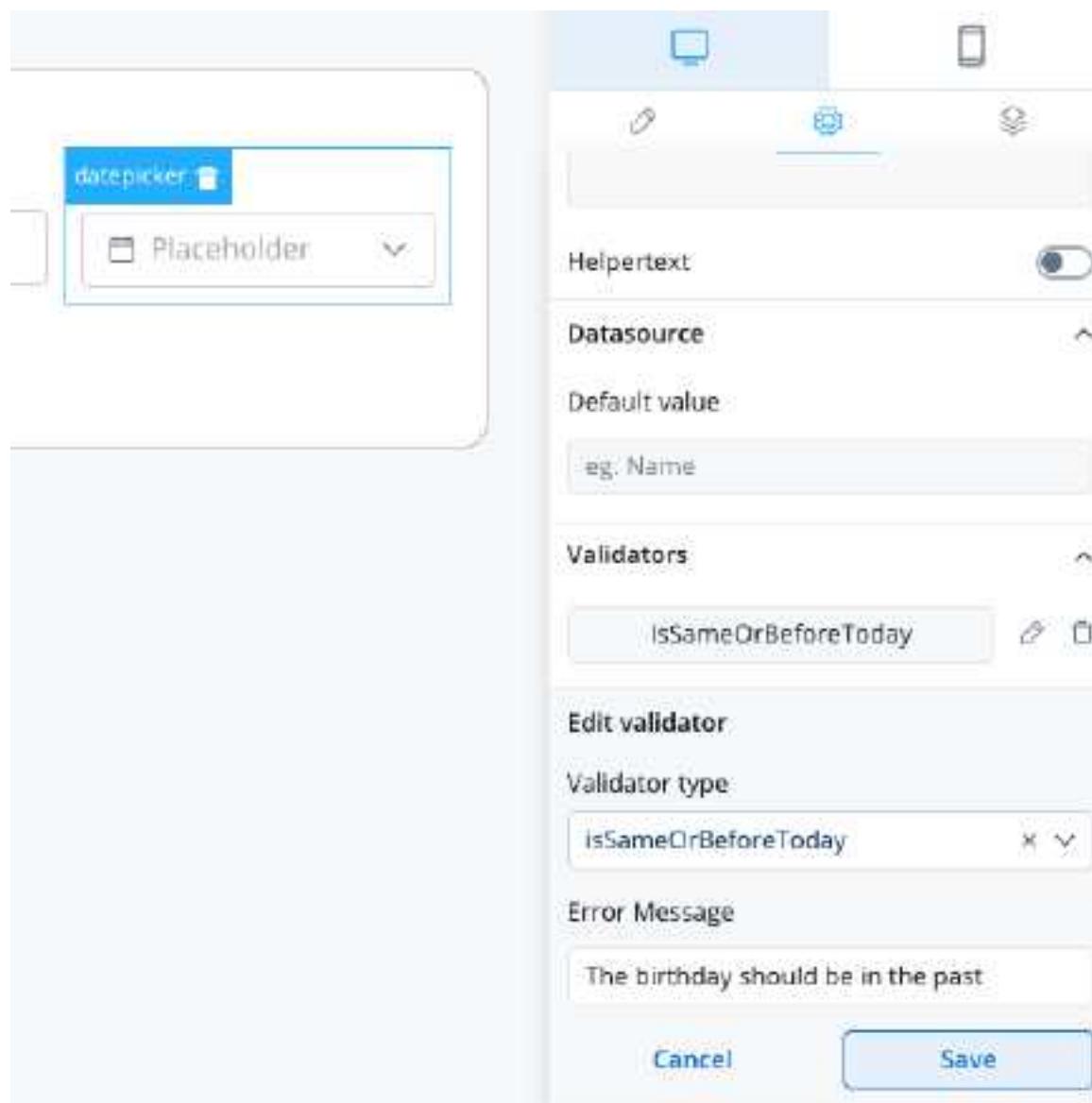
## pattern validator

This validator checks whether the input value matches the specified pattern (for example, a [regex expression](#)).



## datepicker - isSameOrBeforeToday

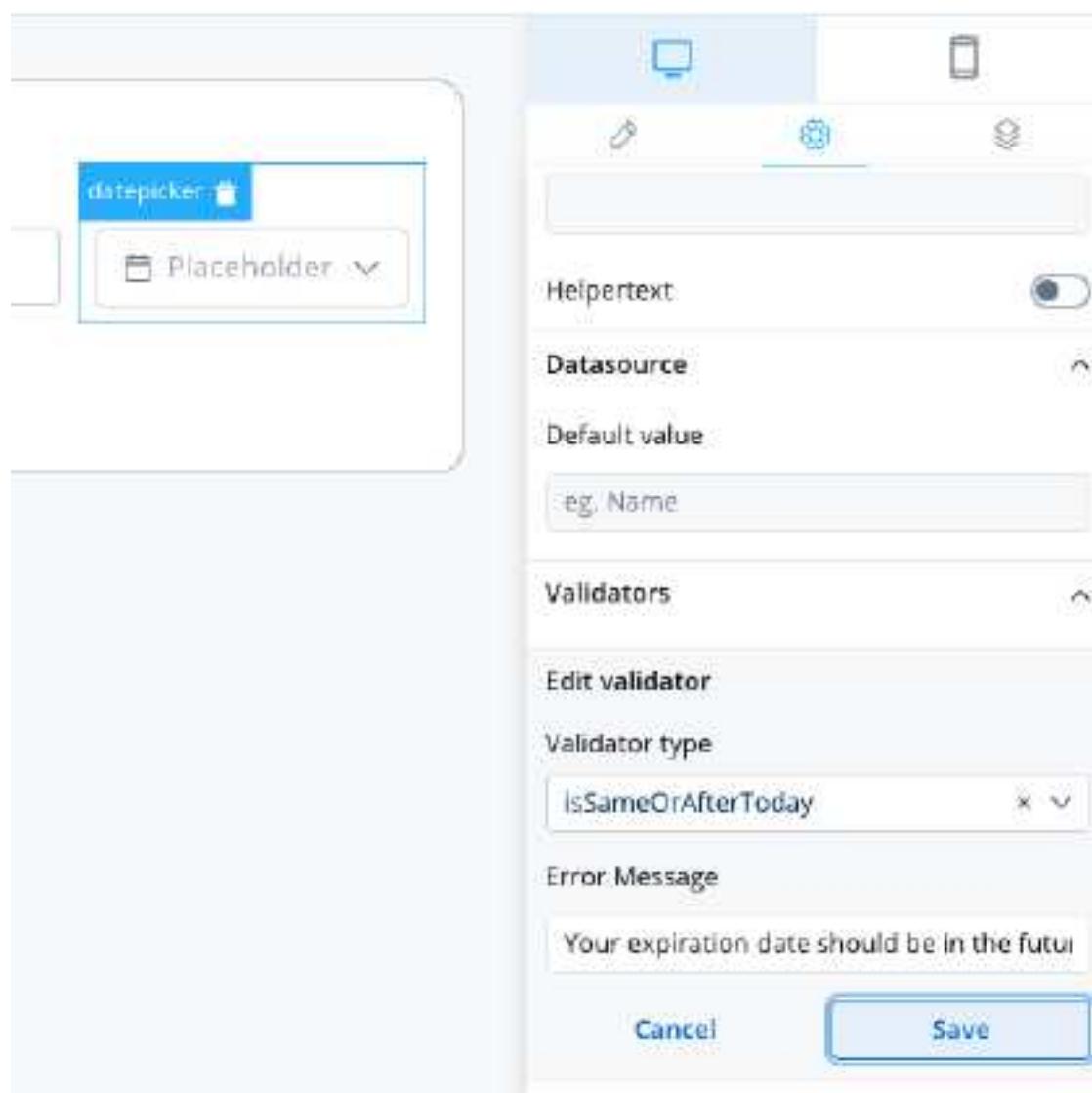
This validator can be used to validate **datepicker** inputs. It checks whether the selected date is today or in the past. If there are no characters at all, this validator will not trigger. It is advisable to use this validator with a **required** validator.



## 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.



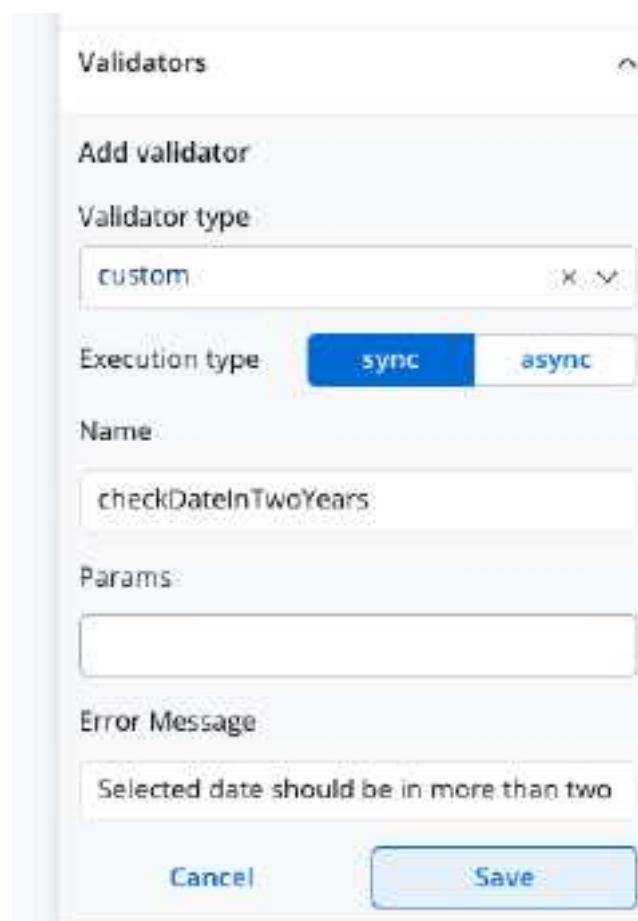
## Custom validators

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.



Validators

Add Validator

Validator type

custom

Execution type

sync

async

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

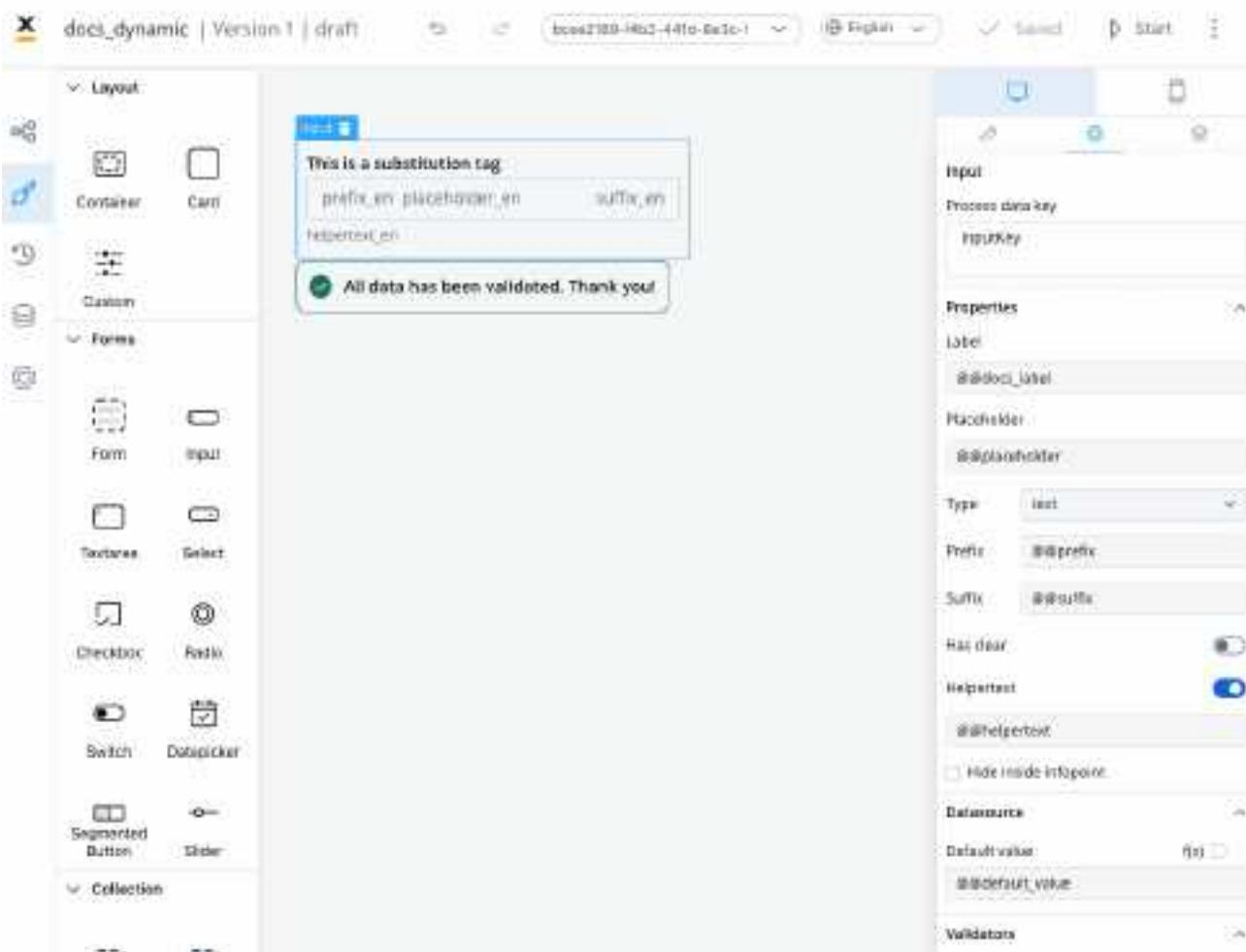
Dynamic values refer to the capability of dynamically populating element properties in the user interface based on process parameters or substitution tags. These values can be customized at runtime, allowing the application to adapt to specific scenarios or user input. With dynamic values, you can personalize 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:

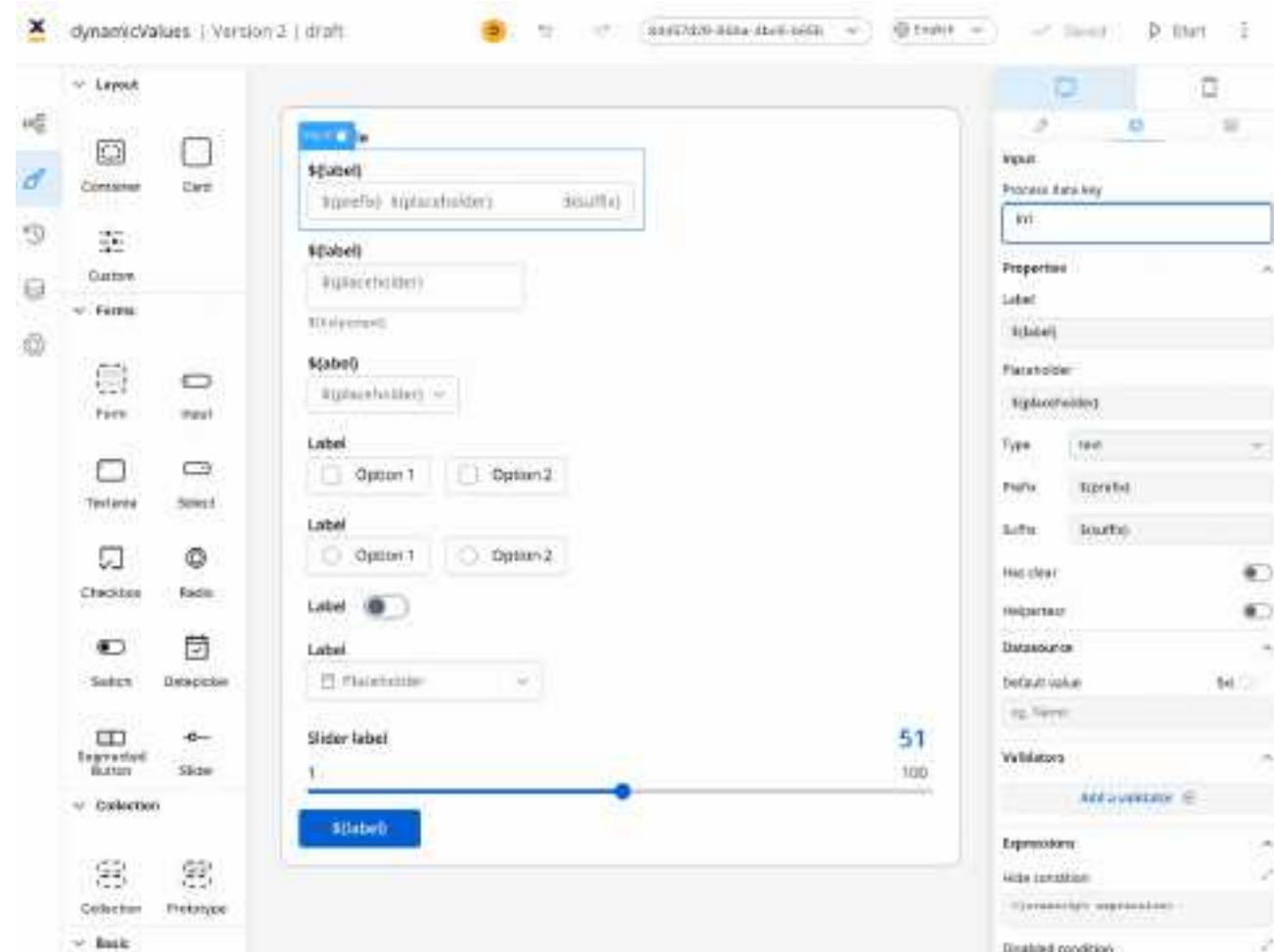
| Element          | Property                      | Accepts Params/Subst Tags |
|------------------|-------------------------------|---------------------------|
| Form Elements    | Default Value (except switch) | Yes                       |
|                  | Label, Placeholder            | Yes                       |
|                  | 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



## 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",
```

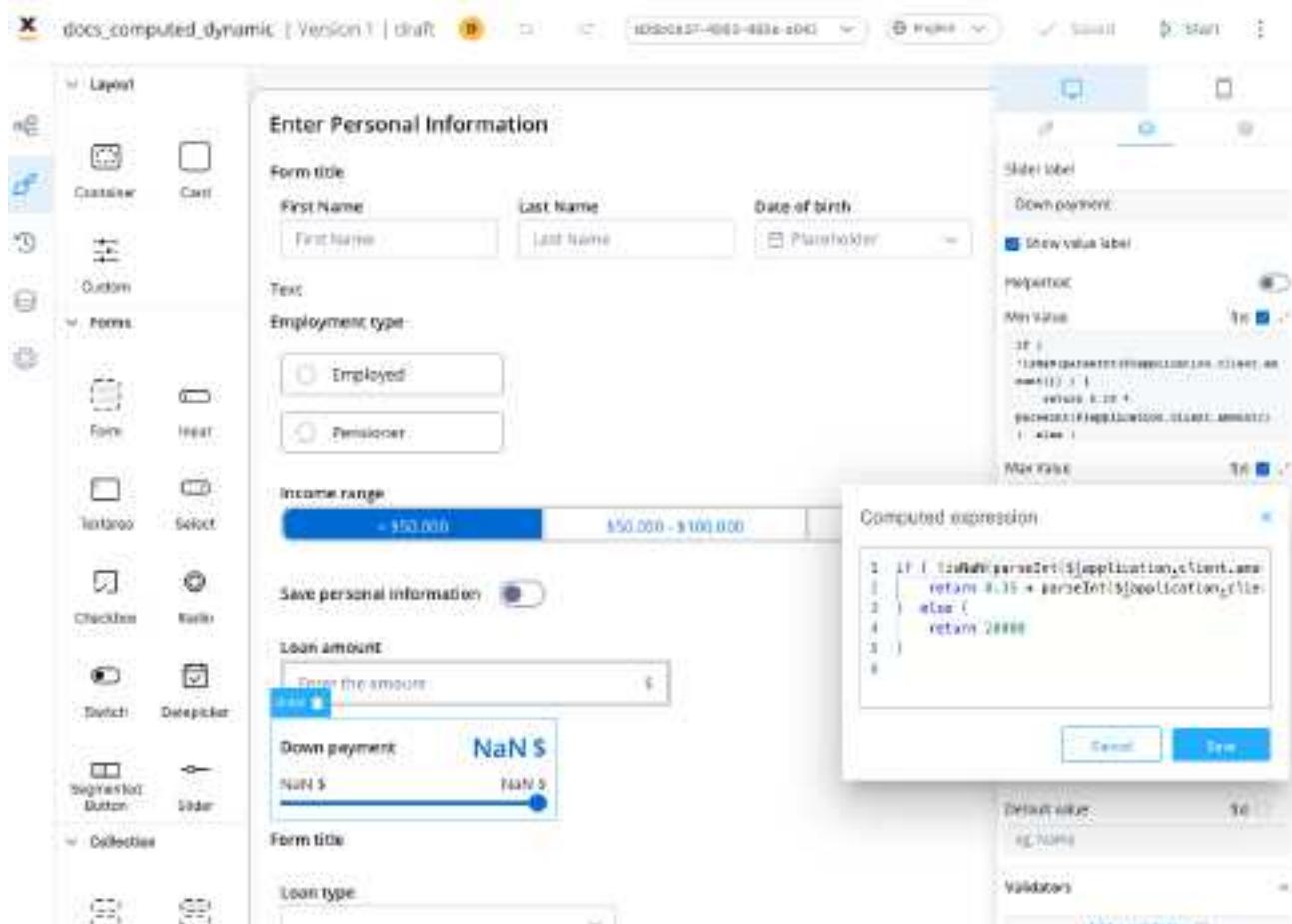
```
"errorM":"This is a error message",
"prefix":"prx",
"suffix":"sfx",
"subtitle":"This is a subtitle",
"message":"This is a message",
"defaultV":"defaultValue",
"value":"Value101",
"value":"Value101",
"confirmLabel":"This is a confirm
label",
"cancelLabel":"This is a cancel label",
"defaultValue":"dfs",
"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.



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.

The screenshot shows a 'Personal Information' form with various fields like First Name, Last Name, Date of Birth, Employment type (Employee or Freelancer), Income range (with a slider from 0 to 100,000), and a 'Save personal information' button. To the right, the 'Computed value' panel is open for a slider input. The formula is set to `if(this.getEmployeeType() == 'Employee', this.getSalary(), 0.0)`. The preview shows 'NaNs' because the 'Employee' condition is not met. Other properties like 'Min Value' (0) and 'Max Value' (100000) are also visible.

## 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**

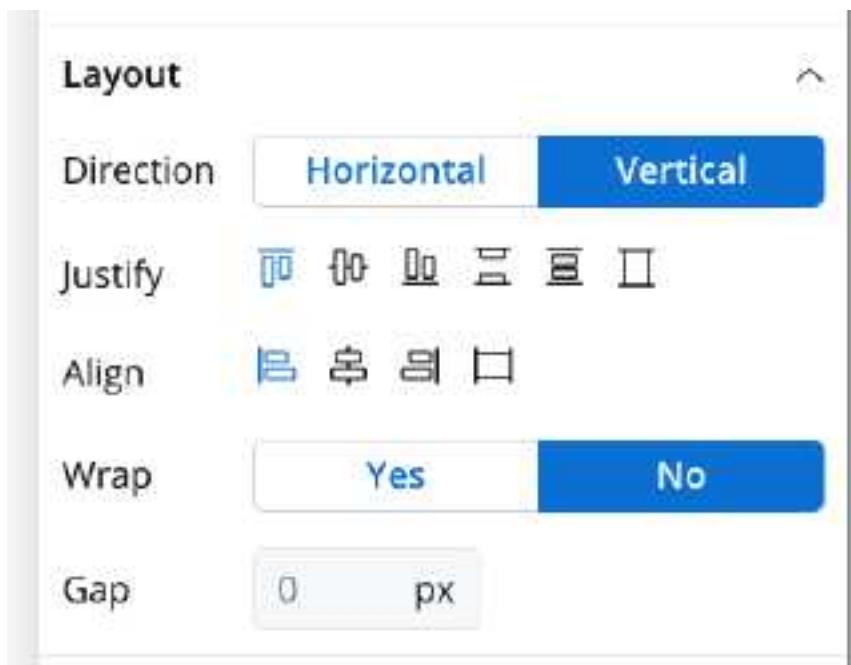
For input elements (e.g., text input), you may require a default value from a process variable, while a number input may need a computed value.

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).

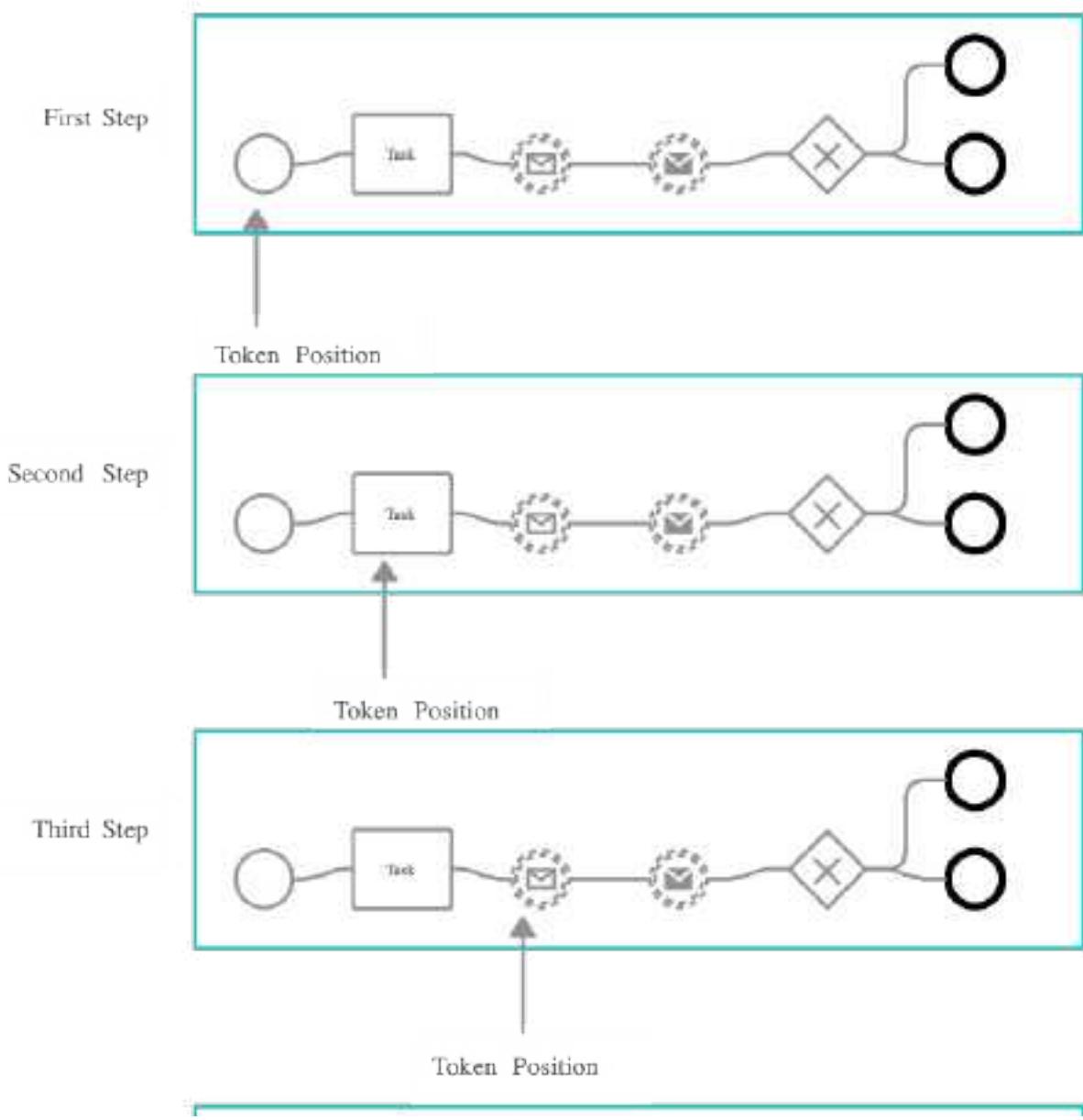
The token is a [BPMN](#) concept that represents a state within a process instance. It keeps track of the current position in the process flow and is used to store data related to the current process instance state.

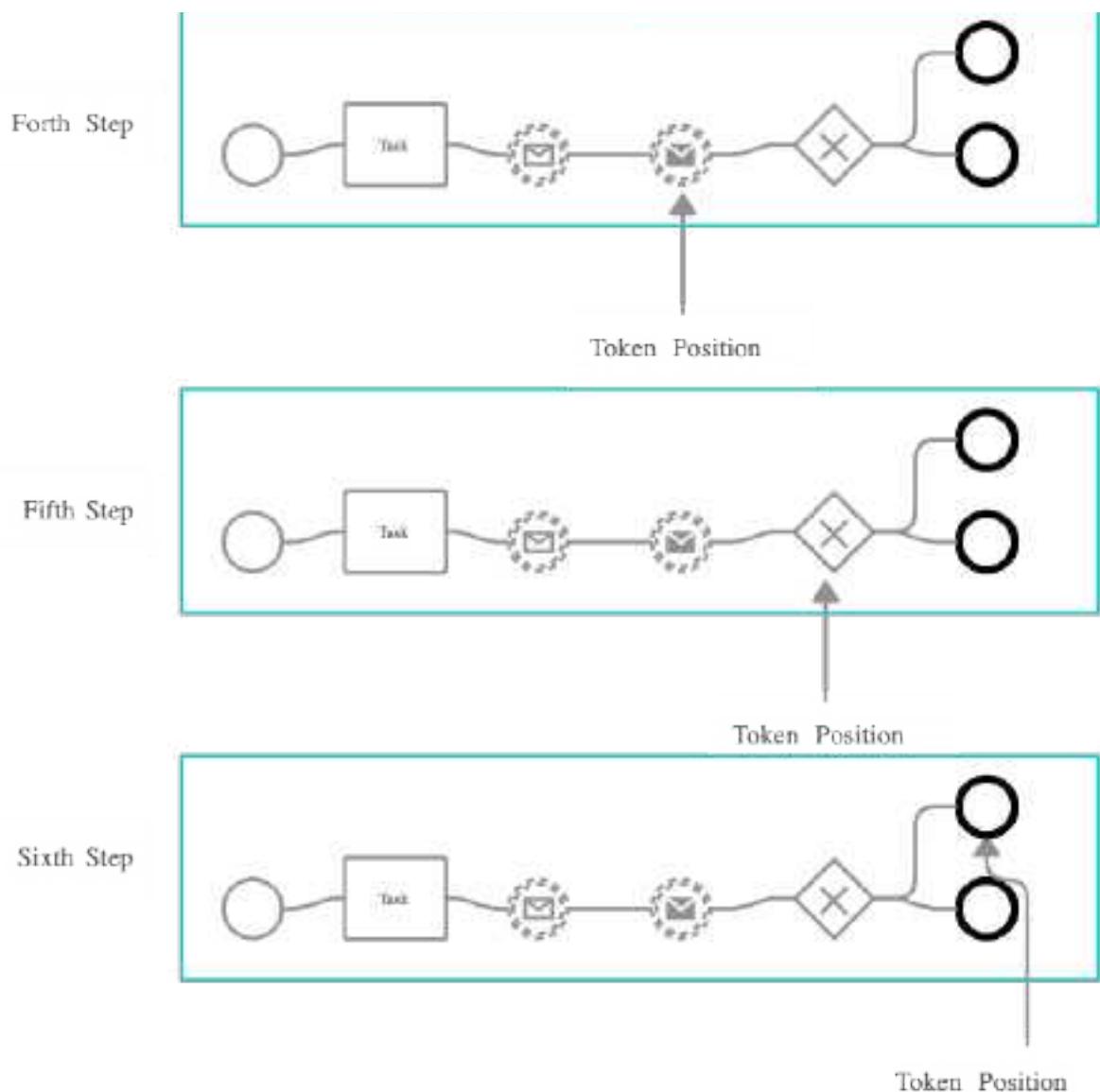
A token is created each time a new process instance is started. As the actions on the process instance are executed, the token advances from one node to the next.

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 'Active processes' table. The table has columns: Process ID, Instance name, Execution, Status, Current Node Name, and Start date. There are two rows of data:

| Process ID                           | Instance name    | Execution | Status | Current Node Name                    | Start date            |
|--------------------------------------|------------------|-----------|--------|--------------------------------------|-----------------------|
| 925e1194-1d6e-4f77-8e9f-54621f93aef0 | sd1_w_100_p_0010 | -         | INITED | a003a56e-5577-4a2c-921f-6561ca0020e0 | 10 Aug 2023, 21:11 PM |
| 206872a7-71e8-46dc-9f70-5d5a87ab6121 | sd1_w_9_0011     | -         | INITED | a042c031-24a0-0010-a05b-409246a04061 | 10 Aug 2023, 11:23 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:

| Name                                | Metric | Created at            | Author   |
|-------------------------------------|--------|-----------------------|----------|
| AutotestProcess_72695033            | Yellow | 09 Jun 2022, 1:49 PM  | QA FlowX |
| TA_BasicSteps_Process_1654862288218 | Yellow | 09 Jun 2022, 11:23 AM | QA FlowX |

| Name                 | Metric | Published at          | Author   |
|----------------------|--------|-----------------------|----------|
| TA_Subprocess_2      | Green  | 09 Jun 2022, 1:40 AM  | QA FlowX |
| TA_Subprocess_iter_2 | Green  | 09 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","actionType":1,"actionValue":null,"isFinal":false}
```

```

1: Object {"nodeId":491657,"name":"stepper","arrivedDate":"2023-07-26T09:57:58.085Z","lastExecutionTime":1685000000000,"state":"COMPLETED","actionStateData":null}
2: Object {"nodeId":491656,"name":"step1","arrivedDate":"2023-07-26T09:57:58.085Z","lastExecutionTime":1685000000000,"state":"COMPLETED","actionStateData":null}
3: Object {"nodeId":491662,"name":"Client Form","arrivedDate":1685000000000,"lastExecutionTime":1685000000000,"state":"COMPLETED","actionStateData":null}
{"492053": {"name": "saveClient", "state": "COMPLETED", "lastExecutionTime": 1685000000000, "actionStateData": null}}
4: Object {"nodeId":491664,"name":"end step1","arrivedDate":"2023-07-26T09:57:58.085Z","lastExecutionTime":1685000000000,"state":"COMPLETED","actionStateData":null}
5: Object {"nodeId":491661,"name":"step2","arrivedDate":"2023-07-26T09:57:58.085Z","lastExecutionTime":1685000000000,"state":"COMPLETED","actionStateData":null}
6: Object {"nodeId":491655,"name":"company form","arrivedDate":1685000000000,"lastExecutionTime":1685000000000,"state":"COMPLETED","actionStateData":null}
{"492052": {"name": "SaveCompany", "state": "COMPLETED", "lastExecutionTime": 1685000000000, "actionStateData": null}}
7: Object {"nodeId":491658,"name":"stop step2","arrivedDate":1685000000000,"lastExecutionTime":1685000000000,"state":"COMPLETED","actionStateData":null}
8: Object {"nodeId":491659,"name":"stop_stepper","arrivedDate":1685000000000,"lastExecutionTime":1685000000000,"state":"COMPLETED","actionStateData":null}
9: Object {"nodeId":492452,"name":"CreateDocument","arrivedDate":1685000000000,"lastExecutionTime":1685000000000,"state":"COMPLETED","actionStateData":null}
{"492102": {"name": "sendInformation", "state": "COMPLETED", "lastExecutionTime": 1685000000000, "actionStateData": null}}
10: Object {"nodeId":492453,"name":"ReceiveDocuments","arrivedDate":1685000000000,"lastExecutionTime":1685000000000,"state":"COMPLETED","actionStateData":null}
18T09:57:58.085271Z", "actionStateData": null}
11: Object {"nodeId":491660,"name":"end_process","arrivedDate":1685000000000,"lastExecutionTime":1685000000000,"state":"COMPLETED","actionStateData":null}
backSeq: Object {"nodes": [491663, 491657, 491656, 491662, 491664, 491655]}
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:

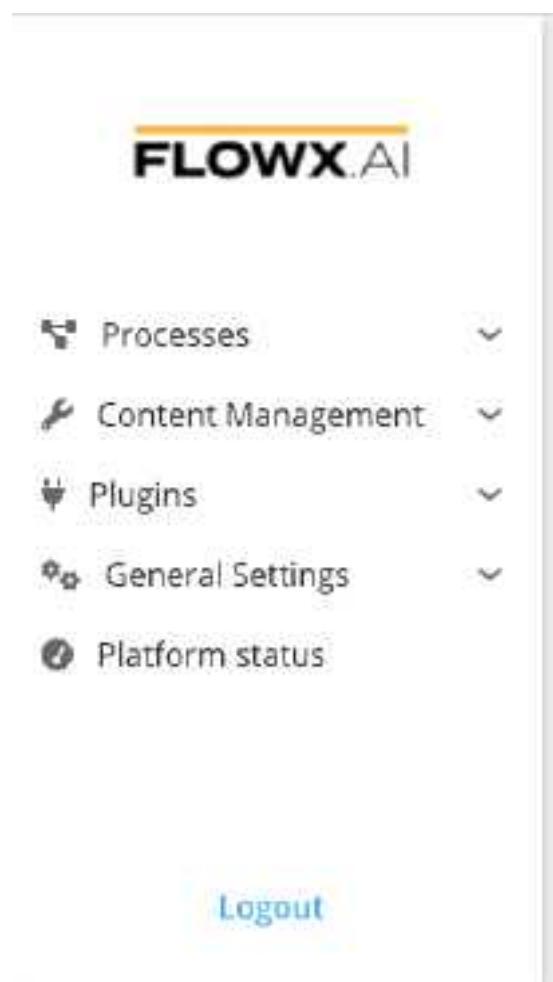
» [GitHub - Nashorn](#)

» [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

# FLOWX.AI

-  Processes ▾
-  Content Management ▾
-  Plugins ▾
-  General Settings ▾
-  Platform status



### ! 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 'Processes' tab selected in the sidebar. The main area is titled 'Process Definitions' and contains two sections: 'Drafts / In progress' and 'Published'. The 'Drafts / In progress' section lists two items: 'Test Process' (version 0.1) and 'Test Process 1' (version 0.1). The 'Published' section lists two items: 'Happy Process' (version 0.1) and 'Distro Process' (version 0.1). Each item has a preview thumbnail, a 'Version' column showing the current version, a 'Edited at' column showing the last edit date and time, and a 'Edited by' column showing the user who made the edit. There are also three action icons (more options, edit, delete) next to each item.

| Name           | Version | Edited at             | Edited by    |
|----------------|---------|-----------------------|--------------|
| Test Process   | 0.1     | 26 May 2023, 11:11 AM | QA FlowX     |
| Test Process 1 | 0.1     | 26 May 2023, 11:13 AM | QA FlowX     |
| Name           | Version | Published at          | Published by |
| Happy Process  | 0.1     | 26 May 2023, 11:09 PM | QA FlowX     |
| Distro Process | 0.1     | 23 May 2023, 8:03 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.

| Active processes |                                      |                       |                        |                   |                      |
|------------------|--------------------------------------|-----------------------|------------------------|-------------------|----------------------|
|                  | ID                                   | Activity name         | Status                 | Current task name | Last update          |
| 1                | 61e921ab-49d6-41a1-a58a-1aa8f1668793 | Client identification | <span>Green</span> End |                   | 18 May 2023, 2:56 PM |
| 2                | 97789c3a-4747-42a8-bc64-641a0562ff18 | Test Process          | <span>Green</span> End |                   | 18 May 2023, 2:57 PM |
| 3                | 3410005d-7224-4f9a-9188-27a3c8fb6f38 | Test Process 1        | <span>Green</span> End |                   | 19 May 2023, 1:11 PM |
| 4                | 637c7653-bc11-4f9e-be24-c3e39e888743 | Test Process 2        | <span>Green</span> End |                   | 19 May 2023, 1:11 PM |
| 5                | 98a590de-2774-46d2-b42e-18a1012d48ba | Test Process 3        | <span>Green</span> End |                   | 19 May 2023, 1:48 PM |
| 6                | 9a273420-bd83-48d1-90fc-71aa7f6cd894 | Test Process 4        | <span>Green</span> End |                   | 19 May 2023, 1:48 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, there is a sidebar with navigation items: 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 sections: 'Drafts / In progress' and 'Published'. The 'Drafts / In progress' section has one item listed:

| No. | Version | Edited at            | Edited by |
|-----|---------|----------------------|-----------|
| 1   | 1       | 25 May 2022, 4:40 PM | QA FlowX  |

The 'Published' section has one item listed:

| No. | Version | Published at          | Published by |
|-----|---------|-----------------------|--------------|
| 1   | 1       | 26 May 2022, 12:40 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.

| Activities |        |            |                 |         |          |                  |
|------------|--------|------------|-----------------|---------|----------|------------------|
|            | Title  | Stage      | Assigned        | Status  | Priority | Last updated     |
|            | Task 1 | Onboarding | silvia@flowx.ai | STARTED | 3        | 25.5.2021, 12:45 |
|            | Task 2 | Onboarding | front@flowx.ai  | EXPIRED | 3        | 8.3.2022, 16:21  |
|            | Task 3 | Onboarding | silvia@flowx.ai | EXPIRED | 3        | 27.1.2021, 17:50 |
|            | Task 4 | Onboarding | silvia@flowx.ai | FAILED  | 3        | 15.12.2021, 2:40 |
|            | Task 5 | Onboarding | silvia@flowx.ai | FAILED  | 3        | 15.12.2021, 2:40 |
|            | Task 6 | Onboarding | silvia@flowx.ai | FAILED  | 3        | 15.12.2021, 2:40 |
|            | Task 7 | Onboarding | silvia@flowx.ai | FAILED  | 3        | 15.12.2021, 2:40 |
|            | Task 8 | Onboarding | silvia@flowx.ai | FAILED  | 3        | 15.12.2021, 2:40 |
|            | Task 9 | Onboarding | silvia@flowx.ai | FAILED  | 3        | 15.12.2021, 2:40 |

» 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. The left sidebar contains navigation links for Processors, Content Management, Plugins, General settings, and Platform status. The main area is titled "Notifications Template - ExampleTemplate". It has tabs for "Body" and "Data model". Under "Body", there is a "Type" dropdown set to "Mail", a "From" field with "ExampleTemplate" selected, and a "To" field containing "Contract.B[[@instance]]". A date picker shows "2023-07-26". Below these are two text areas: one with "Hello #firstinput #secondinput" and another with "Object from details. To be used in a mail." At the bottom are "Test", "Save", and "Delete" 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

At the bottom of the sidebar, there is a "Logout" link.

The main content area is titled "Documents Templates - ExampleTemplate". It has tabs for "Body" and "Data model". The "Body" tab is selected, showing the following content:

- Language: Romanian (Romanian - ro-RO)
- Source button
- Rich text editor toolbar
- Text area containing:

Template generated for the FLOWX.AI Academy by

#firstInput

#secondInput

Leem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sollicitudin leo in sem tempus. ut aliquet eros poenitioe. Nullam condimentum dapibus massa in maximus. Vivamus nulla est, posuere eget metus quis, lacinia euismod magna. Integer non nisi non ipsum poenitior 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
- Text, Save, and Publish buttons

» 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.

| Generic Parameters |                           |             |         |
|--------------------|---------------------------|-------------|---------|
| key                | value                     | environment | actions |
| imagePlaceholder   | https://image.example.com | dev         |         |
| test1              | https://image.jpg         | qa          |         |
|                    | https://image.jpg         | test2       |         |
|                    | https://image.jpg         | test        |         |
|                    | https://image.jpg         | prod        |         |
| baseURL            | https://url1.com/         | qa          |         |
|                    | https://url2.com/         | test2       |         |
| baseURLFromAPI     | https://domain.com/       | qa          |         |
| test               | test_value                | qa          |         |

### 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.

| Username                  | Email                     | Name              |  |
|---------------------------|---------------------------|-------------------|--|
| 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         |  |
| boss.twishes@email.com    | boss.twishes@email.com    | Boss 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 various configuration options like Enumerations, Substitution tags, Content models, Languages, Source systems, Plugins, General Settings, Generic Parameters, Integration management (which is highlighted in blue), Licensing, and Access management. Below the sidebar is a user profile for 'John Doe'. The main content area is titled 'Integrations' and lists five entries:

|   | Name               | Identifier        | System name | Actions |
|---|--------------------|-------------------|-------------|---------|
| 1 | Integration 1      | Intgr             | 0t.         |         |
| 2 | Document PL...     | documentPl...     | FlowX       |         |
| 3 | Notification PL... | notificationPl... | FlowX       |         |
| 4 | Integration Test   | test              | System      |         |
| 5 | 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, a sidebar menu includes: Content Models, Languages, Source Systems, Plugins (Task Manager, All tasks, Hook), Stages, Notification template, Document template, General Settings (Generic Parameters, Licensing, Access management, Users, Roles, Groups), and Platform status. The main area is titled "Platform Status 2.7.0" and contains two tables: "Flow Components" and "Other Components".

| Component name | Status          | Installed version | Suggested version |
|----------------|-----------------|-------------------|-------------------|
| Component 1    | <span>OK</span> | 0.3.40            | 0.3.40            |
| Component 2    | <span>OK</span> | 0.2.23            | 0.2.23            |
| Component 3    | <span>OK</span> | 1.0.37            | 1.0.37            |

| Component name     | Status            | Installed version |
|--------------------|-------------------|-------------------|
| Custom Component   | <span>OK</span>   | 0.1.20            |
| Custom component 1 | <span>DOWN</span> | -                 |

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.

| Process Definitions                   |         |                       |                   |  |
|---------------------------------------|---------|-----------------------|-------------------|--|
| Search for process definition name    |         |                       |                   |  |
| <b>Drafts / In progress</b>           |         |                       |                   |  |
| Name                                  | Version | Last edit             | Created by        | Action   |
| item_3                                | 1.0     | 18 May 2022, 01:19 PM | Drago Calavasle   | <a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a> |
| ts_tbc_goo                            | 1.0     | 18 May 2022, 01:19 PM | QA FlowX          | <a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a> |
| tsi_InvMandatoryKey                   | 1.0     | 18 May 2022, 01:19 PM | QA FlowX          | <a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a> |
| ccc_main_process_principal_3862945... | 1.0     | 18 May 2022, 01:33 AM | user@chiricahu... | <a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a> |
| ccc_main_process_principal_3862978... | 1.0     | 18 May 2022, 01:32 PM | user@chiricahu... | <a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a> |
| <b>Published</b>                      |         |                       |                   |  |
| Name                                  | Version | Published at          | Published by      | Action   |
| ccc_main_process_principal            | 1.0     | 18 May 2022, 01:19 PM | user@chiricahu... | <a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a> |
| taskFlowE                             | 1.0     | 18 May 2022, 01:18 PM | QA FlowX          | <a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a> |
| Vast_Process                          | 1.0     | 18 May 2022, 12:03 PM | QA FlowX          | <a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a> |
| envi_citate_uver                      | 1.0     | 18 Apr 2022, 10:32 AM | stiu grigore      | <a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a> |
| envi_citate_idemtiton                 | 1.0     | 18 Apr 2022, 10:33 AM | stiu grigore      | <a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a> |

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

| Active processes |                                      |                    |                    |  |                      |
|------------------|--------------------------------------|--------------------|--------------------|--|----------------------|
|                  | Process ID                           | Definition name    | Status             | Created Node Name                                  |                      |
| 1                | 10935649-7567-4304-9494-8033546d43   | Libra              | <span>READY</span> | libra@Def-0001-4723-100<br>libra@Def-0001-4723-100 | 10 May 2023, 5:27 PM |
| 2                | 9042f6cc-21a9-42e5-9e73-8817f3478498 | Libra              | <span>READY</span> | libra@Def-0001-4723-100<br>libra@Def-0001-4723-100 | 10 May 2023, 5:23 PM |
| 3                | b612034-8139-4189-9e00-4ee4272a230   | Libra              | <span>READY</span> | libra@Def-0001-4723-100<br>libra@Def-0001-4723-100 | 10 May 2023, 5:23 PM |
| 4                | 10935621-8999-4407-9342-217092099a1  | alice_main_process | <span>READY</span> | alice_1  | 10 May 2023, 4:00 PM |
| 5                | 80aa2f33-80a4-4125-9f13-7ed46c3910ce | alice_main_process | <span>READY</span> | alice_1  | 10 May 2023, 4:00 PM |
| 6                | 144900d-8028-413a-8529-4603-61208329 | alice_main_process | <span>READY</span> | alice_1  | 10 May 2023, 4:00 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

### Process Definitions

Search by process definition name

| Drafts / In progress           |         |                       |                  |
|--------------------------------|---------|-----------------------|------------------|
| Name                           | Version | Created at            | Updated by       |
| memo_5                         | 1       | 18 May 2022, 5:18 PM  | Dragos Catanescu |
| deleteActionsPatenBug          | 1       | 19 May 2022, 4:31 PM  | Dorian Florescu  |
| lbi_test_gao                   | 1       | 19 May 2022, 3:39 PM  | QA FlowX         |
| lbi_testMandatoryKey           | 1       | 19 May 2022, 12:19 PM | QA FlowX         |
| FCI_MSI_Proces_202101...       | 1       | 19 May 2022, 10:12 AM | serban.catanescu |
| Published                      |         |                       |                  |
| Name                           | Version | Published at          | Published by     |
| client_identification_refactor | 1       | 17 Jun 2022, 10:0 PM  | silviu.groze     |
| FCI_MSI_Proces_202101...       | 1       | 18 May 2022, 2:38 PM  | serban.catanescu |
| testVlad6                      | 1       | 09 May 2022, 5:00 PM  | QA FlowX         |
| Vlad_Process                   | 1       | 02 May 2022, 12:20 PM | QA FlowX         |
| lbi_update_user                | 1       | 08 Apr 2022, 10:49 AM | silviu.groze     |

#### 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.

The screenshot shows the FLOWX.AI platform interface. On the left is a sidebar with navigation links: Processes (selected), Content Management, Plugins, General Settings, Platform status, and Layout. The main area is titled "Process Definitions" and contains two tables: "Drafts / In progress" and "Published".

**Drafts / In progress:**

| Name                   | Version | Editor          | Last edit             | Action |
|------------------------|---------|-----------------|-----------------------|--------|
| gateway_decisions_docs | 1.0     | Dragos-Corvinus | 18 May 2022, 11:11 AM | [Edit] |
| item_E                 | 1.0     | Dragos-Corvinus | 18 May 2022, 5:19 PM  | [Edit] |
| dataAccessPlanning     | 1.0     | Oscar-Franco    | 18 May 2022, 11:11 PM | [Edit] |
| SLC-test_gap           | 1.0     | QA_FlowX        | 18 May 2022, 3:59 PM  | [Edit] |
| SLC-testMandatoryKey   | 1.0     | QA_FlowX        | 19 May 2022, 12:18 PM | [Edit] |

**Published:**

| Name                           | Version | Editor         | Published at          | Action |
|--------------------------------|---------|----------------|-----------------------|--------|
| client_identification_refactor | 1.0     | slivengregore  | 17 Jun 2022, 1:08 PM  | [Edit] |
| ccc_mx_process_principal       | 1.0     | seban-chitescu | 18 May 2022, 2:35 PM  | [Edit] |
| testVlad6                      | 1.0     | QA_FlowX       | 18 May 2022, 5:19 PM  | [Edit] |
| Vlad_Process                   | 1.0     | QA_FlowX       | 18 May 2022, 10:20 PM | [Edit] |
| wave_update_user               | 1.0     | slivengregore  | 06 Apr 2022, 11:11 AM | [Edit] |

» Exclusive Gateway Node

Was this page helpful?

# FIOWX.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 interface. On the left, there is a sidebar with navigation links: Processes (Definitions, Active process), Content Management (Documents, Submission tags, Content models, Languages, Source systems), Plugins (Task Manager, Activities, Hooks, Stages, Notification templates, Document templates), General Settings (Generic Parameters), Platform status, and Logout.

The main content area is titled "Process Definitions". It has two sections: "Draft / In progress" and "Published".

**Draft / In progress:**

| Name                      | Version | Created at            | Edited by |
|---------------------------|---------|-----------------------|-----------|
| AutoTestProcess834504323  | 0.1     | 23 May 2023, 1:41 PM  | QA FlowX  |
| TA_AccessRights           | 0.1     | 23 May 2023, 1:15 PM  | QA FlowX  |
| AutoTestProcess144887353  | 0.1     | 23 May 2023, 1:33 PM  | QA FlowX  |
| AutoTestProcess-109600716 | 0.1     | 23 May 2023, 12:32 PM | QA FlowX  |
| AutoTestProcess1277519091 | 0.1     | 23 May 2023, 12:51 PM | QA FlowX  |

**Published:**

| Name                           | Version | Published at          | Published by      |
|--------------------------------|---------|-----------------------|-------------------|
| AutoTestProcess-857887723      | 0.1     | 23 May 2023, 11:10 PM | QA FlowX          |
| AutoTestProcess85487841        | 0.1     | 23 May 2023, 12:48 PM | QA FlowX          |
| shiva_update_user              | 0.1     | 23 May 2023, 11:08 AM | shivagrigore      |
| client_identification_refactor | 0.1     | 17 Jan 2023, 1:08 PM  | shivagrigore      |
| ctrl_main_process_principal    | 0.1     | 18 May 2023, 2:35 PM  | serben.chiriceanu |

» 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 navigation links: Processes, Definitions (which is currently selected), Active process, Content Management, Plugins, and General Settings. The main area is titled "Process Definitions" and contains two tables: "Drafts / In progress" and "Published".  
**Drafts / In progress:**

| Name                | Version | Created at            | Created by       |
|---------------------|---------|-----------------------|------------------|
| Stepper Structure   | 1.0     | 24 May 2022, 11:41 AM | Dragos Cernavade |
| IA_ActionProcess_v1 | 1.0     | 24 May 2022, 8:08 AM  | QA FlowX         |
| APIGateway          | 1.0     | 23 May 2022, 11:32 AM | Andrea Popescu   |

  
**Published:**

| Name                   | Version | Published at         | Published by |
|------------------------|---------|----------------------|--------------|
| AutoTestProcess-147... | 0.1     | 23 May 2022, 8:52 PM | QA FlowX     |
| AutoTestProcess147...  | 0.1     | 23 May 2022, 8:18 PM | QA FlowX     |
| AutoTestProcess212...  | 0.1     | 23 May 2022, 8:10 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 interface with the title "Process Definitions". On the left, there is a sidebar with navigation links for "Processes", "Central Management", "Plugins", and "General Settings".

The main content area is divided into two sections:

- Drafts / Incomplete:** This section lists four process definitions:
  - Folder Structure (Status: Incomplete, Last updated: 13 May 2023, 12:31 PM, Created by: Dragos Cioroianu)
  - numbers\_drag\_drop (Status: Incomplete, Last updated: 14 May 2023, 10:04 AM, Created by: Dragos Cioroianu)
  - AutoTestProcess-801144700 (Status: Incomplete, Last updated: 13 May 2023, 9:00 PM, Created by: QA FlowX)
  - AutoTestProcess-TEST004214 (Status: Incomplete, Last updated: 13 May 2023, 8:18 PM, Created by: QA FlowX)
- Published:** This section lists four process definitions:
  - AutoTestProcess-179967906 (Status: Published, Last updated: 13 May 2023, 12:27 PM, Published by: QA FlowX)
  - AutoTestProcess-106620483 (Status: Published, Last updated: 13 May 2023, 8:18 PM, Published by: QA FlowX)
  - AutoTestProcess-143367164 (Status: Published, Last updated: 13 May 2023, 9:18 PM, Published by: QA FlowX)
  - AutoTestProcess-2522103317 (Status: Published, Last updated: 13 May 2023, 7:38 PM, Published by: 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 process management interface. On the left, there's a sidebar with navigation links for Processes (Drafts, Active, In Progress), Customer Management (Franchises, Substitution tags, Content review), Languages, System settings, Plugins (Task Manager, API keys, Hosts, Logs), and Notifications/templates.

The main area is titled "Process Definitions". It has two sections:

- Drafts/in progress:** Shows two entries: "Test Process" (version 1, created 26 May 2022, 14:03 PM, submitted by John Doe) and "Test Process 1" (version 1, created 26 May 2022, 14:03 PM, submitted by QA FlowX).
- Published:** Shows two entries: "Test Process 3" (version 1, published 26 May 2022, 12:45 PM, published by John Doe) and "Shipping Situation" (version 1, published 26 May 2022, 12:12 PM, published by John Doe).

A search bar at the top right says "Search by process definition name".

## 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, a sidebar navigation includes 'Processes' (selected), 'Definitions', 'Active process', 'Content Management' (selected), 'Enumeration', 'Substitution maps', 'Current models', 'Languages', 'Source systems', 'Plugins', 'Task Manager', and 'All results'. The main area is titled 'Process Definitions' with a search bar. It displays two sections: 'Drafts / In progress' and 'Published'.  
**Drafts / In progress:**

| Name            | Version | Edited at             | Edited by |
|-----------------|---------|-----------------------|-----------|
| AutsTestProc... | 0.1     | 26 May 2022, 12:26 PM | QA-FinxK  |
| AutsTestProc... | 0.1     | 26 May 2022, 12:36 PM | QA-FinxX  |

  
**Published:**

| Name              | Version | Published at          | Published by     |
|-------------------|---------|-----------------------|------------------|
| stopper struct... | 0.1     | 26 May 2022, 12:13 PM | Dragon Corwasile |
| AutsTestProc...   | 0.1     | 26 May 2022, 12:17 PM | QA-FinxK         |

## 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 interface with a sidebar containing navigation links like Processes, Dictionaries, Active process, Content Management, Plugins, Task Manager, etc. The main area is titled 'Process Definitions' and shows two published processes:

| Name                    | Version | Published at          | Published by      |
|-------------------------|---------|-----------------------|-------------------|
| stopper structure       | 0.1     | 24 May 2022, 12:12 PM | Dragos Caranfilie |
| AutoTestProcess22300756 | 0.1     | 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).

FLOWX.AI

Process Definitions

Drafts / In progress

| Name                      | Version | Submitted             | Edited by |
|---------------------------|---------|-----------------------|-----------|
| AutoTestProcess-1E5163... | 0.1     | 24 May 2022, 10:35 PM | QA FlowX  |
| AutoTestProcess-430180... | 0.1     | 24 May 2022, 10:10 PM | QA FlowX  |

Published

| Name                      | Version | Published at          | Published by     |
|---------------------------|---------|-----------------------|------------------|
| stepper structure         | 0.1     | 24 May 2022, 10:18 PM | Dragon Gearwheel |
| AutoTestProcess2736671... | 0.1     | 24 May 2022, 10:07 PM | QA FlowX         |

Search by process definition name

Processes

- Definitions
- Active process

Content Management

- Enumerations
- Subscription tags
- Content models
- Languages
- Source systems

Plugins

- Task Manager
- All tasks

Issues

FLOWX.AI

Process Definitions

Drafts / In progress

| Name                      | Version | Submitted at          | Edited by |
|---------------------------|---------|-----------------------|-----------|
| AutoTestProcess-1E5163... | 0.1     | 24 May 2022, 10:30 PM | QA FlowX  |
| AutoTestProcess-430180... | 0.1     | 24 May 2022, 10:10 PM | QA FlowX  |

Published

| Name                      | Version | Published at          | Published by     |
|---------------------------|---------|-----------------------|------------------|
| stepper structure         | 0.1     | 24 May 2022, 10:18 PM | Dragon Gearwheel |
| AutoTestProcess2736671... | 0.1     | 24 May 2022, 10:07 PM | QA FlowX         |

Search by process definition name

Processes

- Definitions
- Active process

Content Management

- Enumerations
- Subscription tags
- Content models
- Languages
- Source systems

Plugins

- Task Manager
- All tasks

Issues

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 containing navigation links like Processes, Content Management, and Plugins. The main area is titled 'Process Definitions' and displays two entries under 'Drafts / In progress'. Both entries are named 'AutoTest...' and have a version of '1.0'. The first entry was edited at '23 May 2022, 8:57 PM' by 'QA FlowX'. The second entry was published at '23 May 2022, 8:59 PM' by 'QA FlowX'. A search bar at the top right says 'Search by process definition'.

| Name        | Version | Edited at            | Edited by |
|-------------|---------|----------------------|-----------|
| AutoTest... | 1.0     | 23 May 2022, 8:57 PM | QA FlowX  |
| AutoTest... | 1.0     | 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 with the sidebar collapsed. The main area is titled "Process Definitions". It contains two sections: "Drafts / In progress" and "Published".

**Drafts / In progress:**

| Name           | Version | Edited at             | Created by |
|----------------|---------|-----------------------|------------|
| AutoTestPro... | 0.1     | 27 May 2022, 11:42 AM | QA FlowX   |
| TA_ActionP...  | 0.1     | 27 May 2022, 9:29 AM  | QA FlowX   |

**Published:**

| Name           | Version | Published at         | Published by |
|----------------|---------|----------------------|--------------|
| AutoTestPro... | 0.1     | 24 May 2022, 3:51 PM | QA FlowX     |
| AutoTestPro... | 0.1     | 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.

| Name   | Version | Edited at             | Edited by |
|--------|---------|-----------------------|-----------|
| Test 1 | 1       | 27 May 2022, 12:01 PM | John Doe  |
| Test 2 | 1       | 26 May 2022, 4:49 PM  | Jane Doe  |
| Test 3 | 1       | 26 May 2022, 12:43 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:

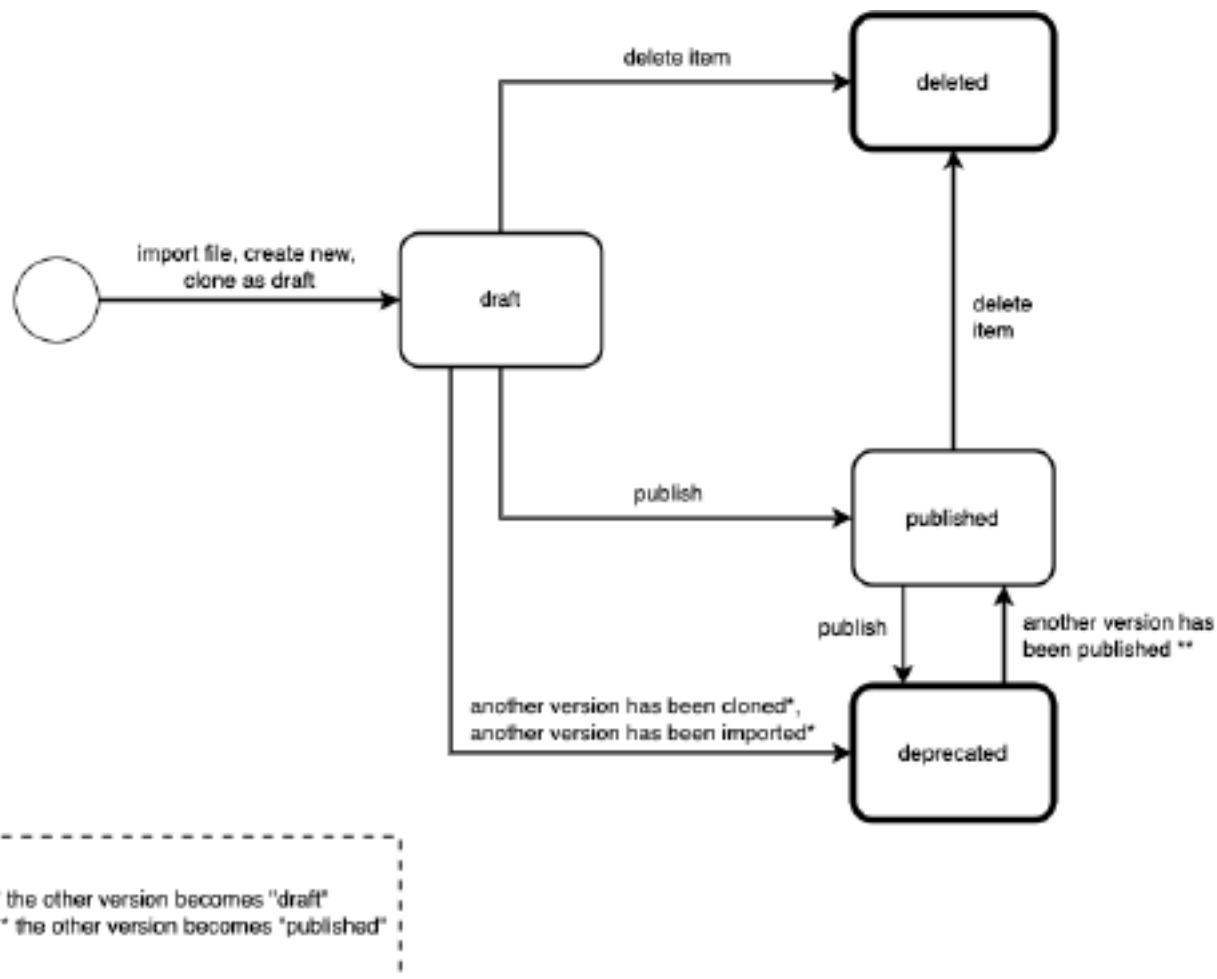
- **draft** - in progress and can't be used to create  
The fallback content to display on prerendering  
, can only be started in test mode
- **published** - only one version can be published at a time, and it will be used to  
instantiate new processes; published versions can be edited
- **deleted** - 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:

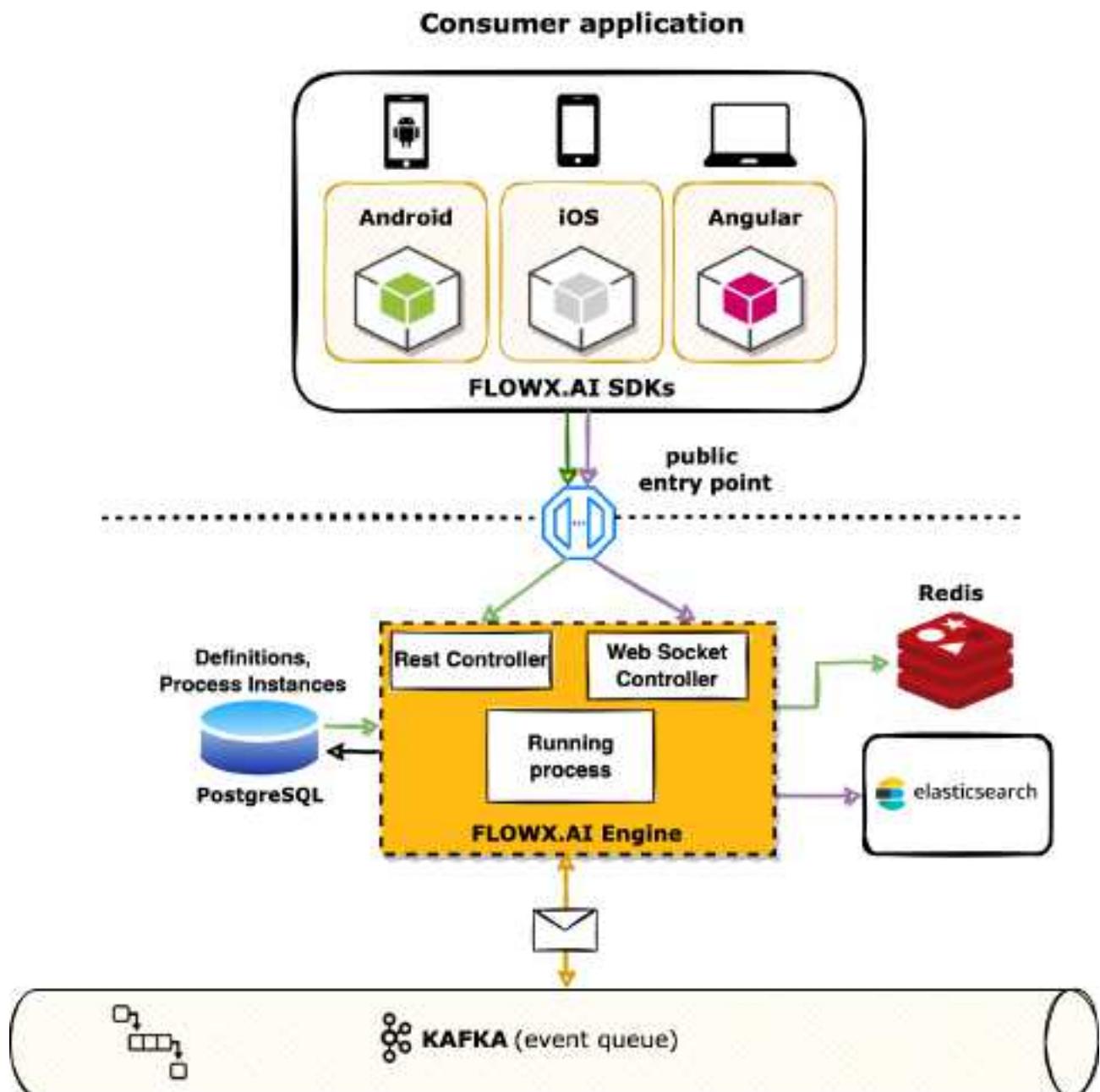
SECURITY\_ACCESSAUTHORIZATIONS\_MANAGEPROCESSES\_SCOPES\_READ\_ROLES  
ROLE\_NAME\_TEST

**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:

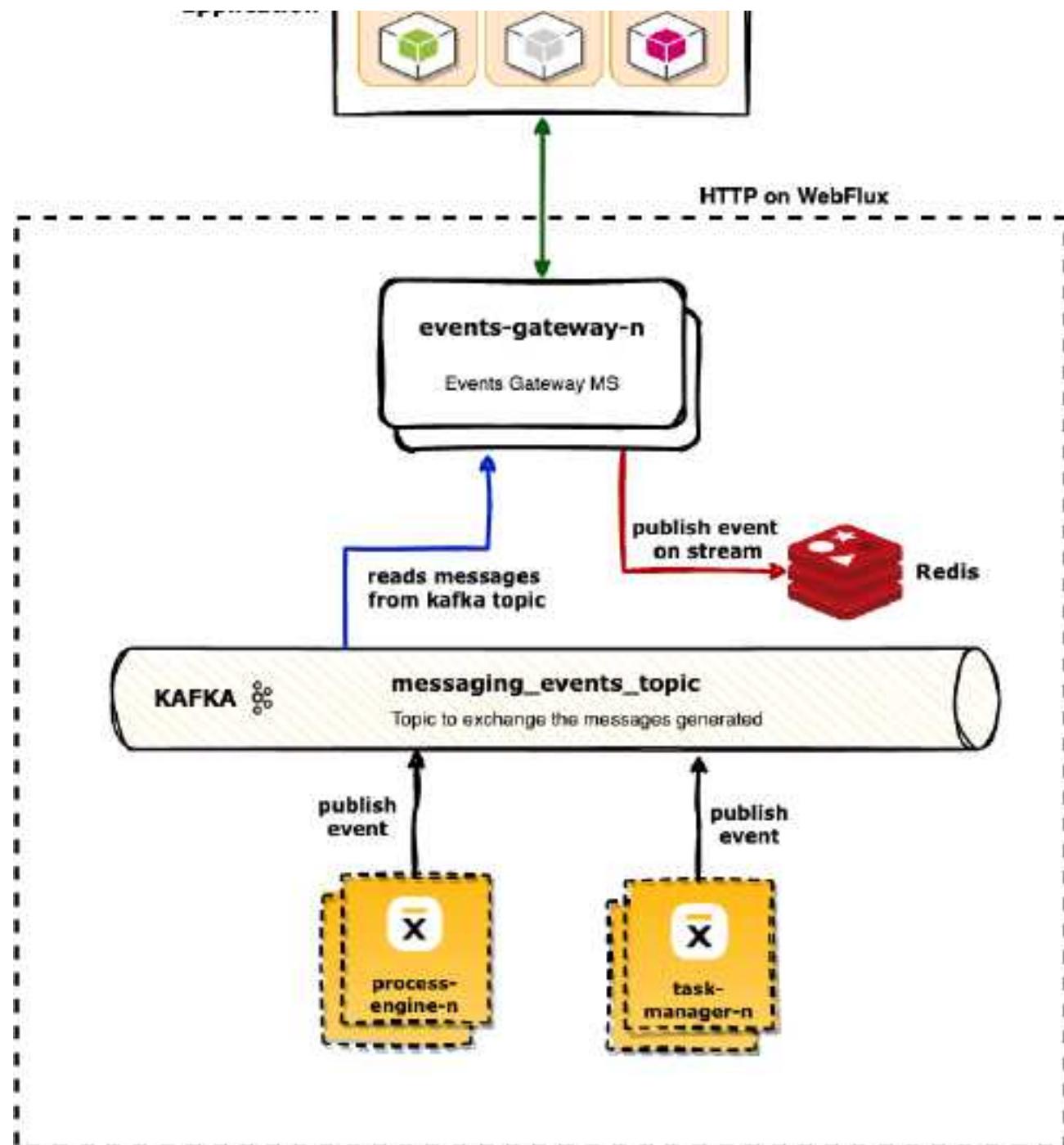
» [Advancing controller setup guide](#)

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**.

| Name              | Version | Draft                               | Last updated         |   |
|-------------------|---------|-------------------------------------|----------------------|---|
| ActivityDomain... | 1.0     | <input checked="" type="checkbox"/> | 24 Jun 2022, 5:47 PM | <a href="#">Open</a> <a href="#">Delete</a> |
| ActivityDomain... | 1.0     | <input checked="" type="checkbox"/> | 23 Jun 2022, 4:21 PM | <a href="#">Open</a> <a href="#">Delete</a> |
| ActivityDomain... | 1.0     | <input checked="" type="checkbox"/> | 24 Jun 2022, 5:47 PM | <a href="#">Open</a> <a href="#">Delete</a> |
| County            | 1.0     | <input checked="" type="checkbox"/> | 24 Jun 2022, 5:47 PM | <a href="#">Open</a> <a href="#">Delete</a> |
| Enumeration_...   | 1.0     | <input checked="" type="checkbox"/> | 27 Jun 2022, 1:09 PM | <a href="#">Open</a> <a href="#">Delete</a> |

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:

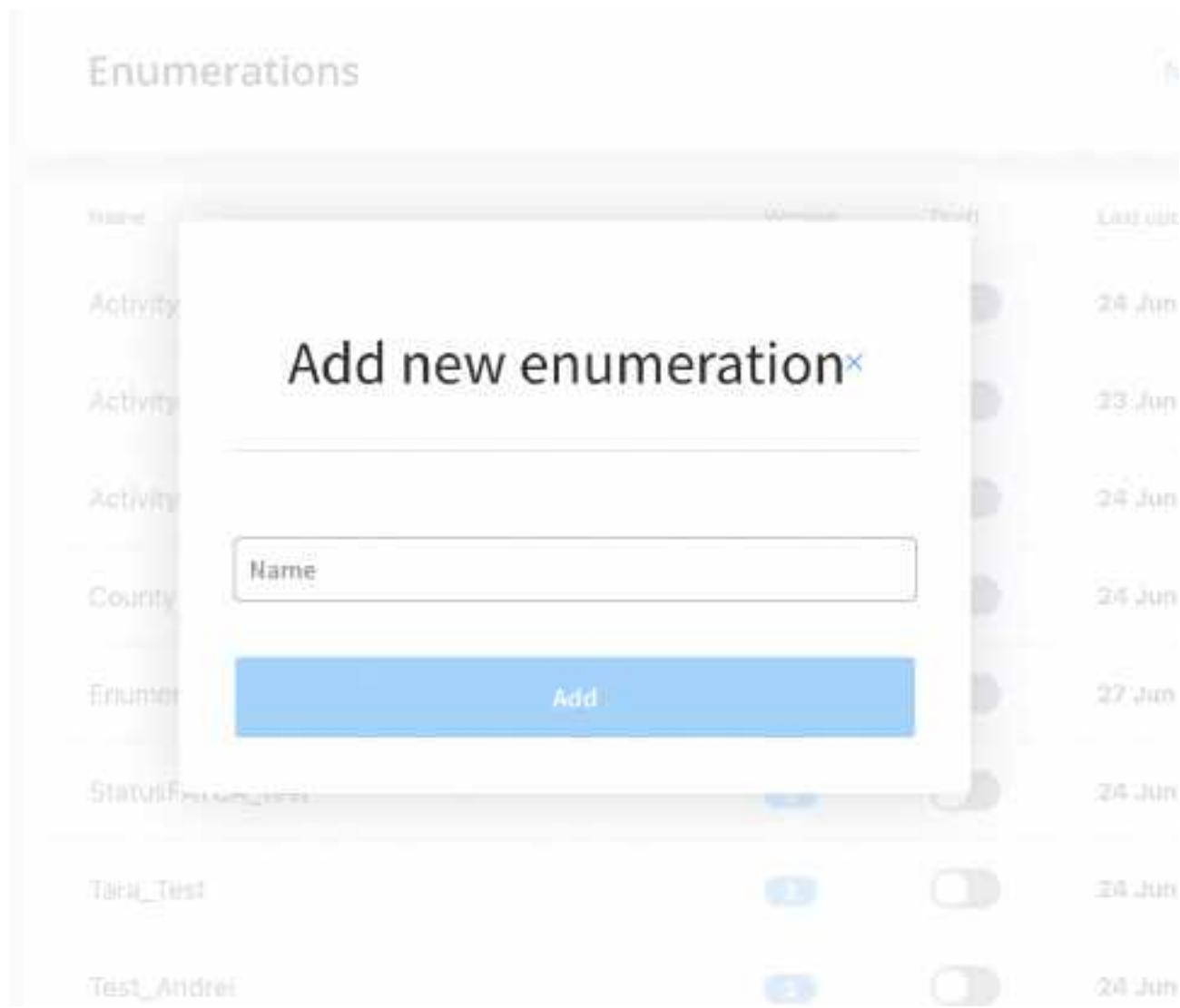
- **Code** - not displayed in the end-user interface, but used to assure value uniqueness
- **Labels** - strings that are displayed in the end-user interface, according to the language set for the generated solution
- **External source systems codes** - values that are set for each external system that 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 |               |               |    |      |       | <a href="#">New value</a>                                    |
|-------------------------------------|---------------|---------------|----|------|-------|--|
| Code                                | en            | fr            | de | FLEX | FLOWX |  |
| A                                   | Agricultur... | Agricultur... | -  | -    | -     | <a href="#">»</a> <a href="#">edit</a> <a href="#">trash</a> |
| B                                   | Extractiv...  | Extractiv...  | -  | -    | -     | <a href="#">»</a> <a href="#">edit</a> <a href="#">trash</a> |
| C                                   | Industria...  | Manufact...   | -  | -    | -     | <a href="#">»</a> <a href="#">edit</a> <a href="#">trash</a> |
| D                                   | Productio...  | Productio...  | -  | -    | -     | <a href="#">»</a> <a href="#">edit</a> <a href="#">trash</a> |

## 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**.



## 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

| ID                 | Name               | Description | Type | FlowX |
|--------------------|--------------------|-------------|------|-------|
| test_enumeration_1 | test_enumeration_1 |             |      |       |

## 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).

The screenshot shows the FLOWX.AI interface with a sidebar on the left containing navigation links: 'Processes' (with 'Definitions' and 'Active process' sub-links), 'Content Management' (with 'Enumerations' and 'Substitution tags' sub-links), and a 'New value...' button. The main content area is titled 'Values for testEnumerationDocs'. It features a table with columns: Code, TO, FF, OO, FLEX, and FLOWX. A single row is present with the code 'test\_test', value 'Test', and the FLOWX column highlighted with a red border. To the right of the table are edit and delete icons.

| Code      | TO   | FF | OO | 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

| Name                      | Version | Draft | Last used            | Actions  |
|---------------------------|---------|-------|----------------------|--|
| Activity Domain Companies | 1.0     | OFF   | 24 Jun               | <span>Import</span><br><span>from JSON</span><br><span>from CSV</span><br><span>New enumeration</span> |
| Test Enumerations         | 1.0     | OFF   | 23 Jun               | <span>Export</span><br><span>to JSON</span><br><span>to CSV</span>                                     |
| Example                   | 1.0     | OFF   | 24 Jun               | <span>Import</span><br><span>from JSON</span><br><span>from CSV</span>                                 |
| County                    | 1.0     | OFF   | 24 Jun 2022, 5:47 PM | <span>Import</span><br><span>from JSON</span><br><span>from CSV</span>                                 |
| Test                      | 1.0     | OFF   | 24-Jun 2022, 5:47 PM | <span>Import</span><br><span>from JSON</span><br><span>from CSV</span>                                 |
| Test Monthly Income       | 1.0     | OFF   | 24-Jun 2022, 5:47 PM | <span>Import</span><br><span>from JSON</span><br><span>from CSV</span>                                 |
| Test Activity Domains     | 1.0     | OFF   | 24-Jun 2022, 5:47 PM | <span>Import</span><br><span>from JSON</span><br><span>from CSV</span>                                 |
| Monthly Income            | 1.0     | OFF   | 24-Jun 2022, 5:47 PM | <span>Import</span><br><span>from JSON</span><br><span>from CSV</span>                                 |

## 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                      |         |       |                      |
|-----------------------------------|---------|-------|----------------------|
| Name                              | Version | Draft | Last updated         |
| ActivityDomain_Companies          | 1.0     | OFF   | 27 Jun 2022, 6:00 PM |
| ActivityDomain_Companies-A-01-01  | 1.0     | OFF   | 23 Jun 2022, 4:21 PM |
| ActivityDomain_Companies-B-05-002 | 1.0     | OFF   | 27 Jun 2022, 6:00 PM |
| County                            | 1.0     | OFF   | 27 Jun 2022, 6:00 PM |
| Enumeration_Tesy                  | 1.0     | OFF   | 27 Jun 2022, 1:09 PM |
| StatusFATCA_test                  | 1.0     | OFF   | 27 Jun 2022, 6:00 PM |
| Terr_Test                         | 1.0     | OFF   | 27 Jun 2022, 6:00 PM |
| Test_Andrei                       | 1.0     | OFF   | 27 Jun 2022, 6:00 PM |
| VenitLuna_test                    | 1.0     | OFF   | 27 Jun 2022, 6:00 PM |

Was this page helpful?

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

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

The fallback content to display on prerendering

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

| Key          | en-GB | en-US | de     | fr     | it | es | Action |
|--------------|-------|-------|--------|--------|----|----|--------|
| emptyState   | -     | -     | Per... | Th...  | -  | -  |        |
| firstname    | -     | -     | Pre... | Fir... | -  | -  |        |
| openAccount  | -     | -     | Op...  | Op...  | -  | -  |        |
| openPersonal | -     | -     | De...  | Op...  | -  | -  |        |
| searchAfter  | -     | -     | Ca...  | Se...  | -  | -  |        |
| openPF       | -     | -     | De...  | Op...  | -  | -  |        |
| file         | -     | -     | 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.

Sul

## Add new substitution tag<sup>x</sup>

Key

empty

firs

open

open

sear

open

rezo

sear

Key

Languages

ro:

en:

de:

**Add**

The screenshot shows a user interface for managing substitution tags. On the left, there's a vertical sidebar with navigation items like 'Sul', 'empty', 'firs', 'open', 'open', 'sear', 'open', 'rezo', and 'sear'. The main area has a title 'Add new substitution tag' with a close button. Below it is a 'Key' input field. Under 'Languages', there are three dropdown menus: 'ro:' (selected), 'en:', and 'de:'. At the bottom is a large blue 'Add' button.

### ⚠ 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 containing navigation links: Processes (Definitions, Active process), Content Management (Enumerations, Substitution tags, Content models, Languages, Source systems). The main area is titled 'Content Models' and contains a table with three rows. The columns are 'Model name', 'Edited at', and 'Edited by'. The rows are: 'Links' (Edited at: 2023-07-26, Edited by: [user icon]), 'article' (Edited at: 2023-07-26, Edited by: [user icon]), and 'test\_doi' (Edited at: 2023-07-26, Edited by: [user icon]). A 'New content model' button is located in the top right corner of the main area.

| Model name | Edited at  | Edited by   |
|------------|------------|-------------|
| Links      | 2023-07-26 | [user icon] |
| article    | 2023-07-26 | [user icon] |
| test_doi   | 2023-07-26 | [user icon] |

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 titled "General settings". In the "Name" field, the value "Property name" is entered. The "Type" dropdown menu is open, showing options: "String" (selected), "Number", and "Boolean". The "Mandatory" checkbox is unchecked. There are "Save" and "Close" buttons at the bottom right.

## 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 with a sidebar containing navigation links such as Processes, Content Management (with sub-links like Enumeration, Substitution tags, Content models, Languages, and Source systems), and Plugins. The main content area is titled "Content Models" and displays a table with three rows:

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

A "New content model" button is located in the top right corner of the content area.

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, Definitions, Active process, Content Management, Enumerations, Substitution tags, Content models, Languages (which is highlighted in blue), and Source systems. The main area is titled "Languages" and displays a table with three rows. The columns are labeled "Code", "Name", and "Default". The rows show "ro" (Romanian) as the default language, "en" (English) which is not marked as default, and "de" (German) which is also not marked as default. There are edit and delete icons next to each row.

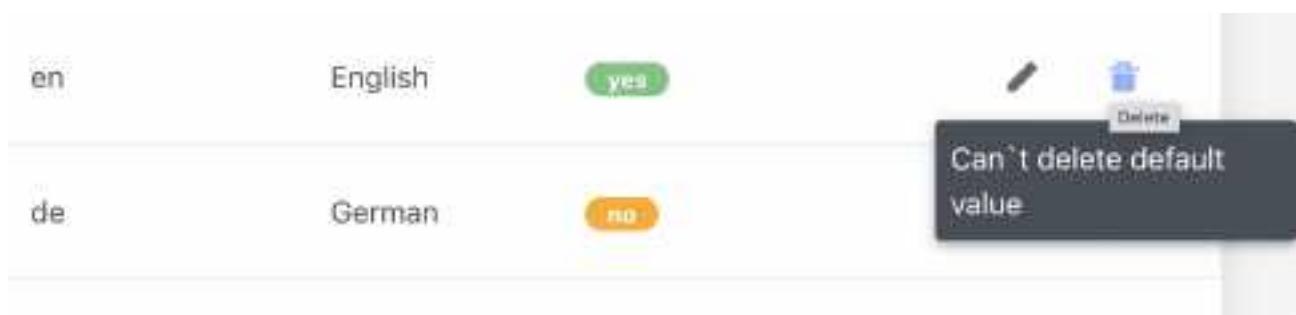
| Code | Name     | Default |
|------|----------|---------|
| ro   | Romanian | ✓       |
| en   | English  | ✗       |
| de   | German   | ✗       |

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

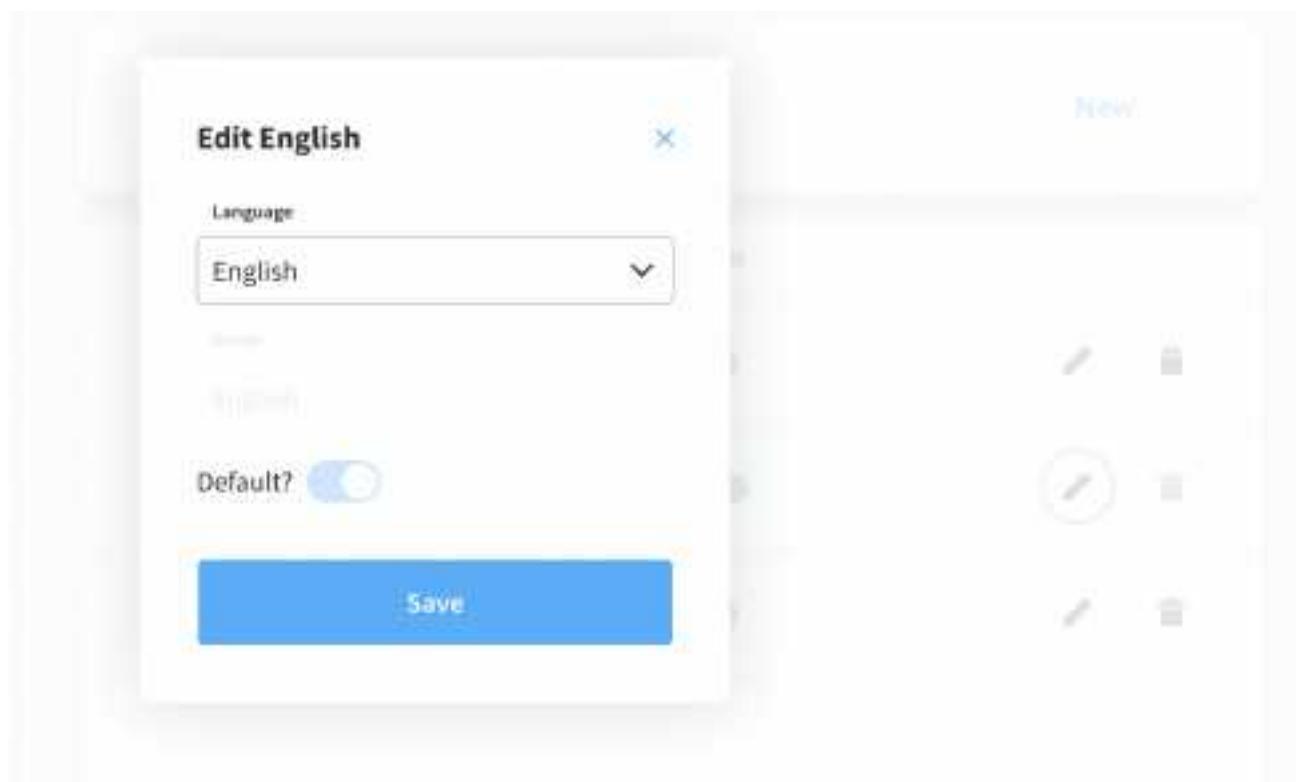
- **Code** - not displayed in the end-user interface, but used to assure value uniqueness
- **Name** - the name of the language
- **Default** - you can set a language as **Default** (default values can't be deleted)

### ⚠ CAUTION

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



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



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

### 🔥 DANGER

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

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

## Adding a new language

To add a new language, follow the next steps:

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

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

- Content Management
  - Enumerations
  - Substitution tags
  - Content models
  - Languages** (highlighted)
  - Source systems
- Plugins
  - Task Manager
  - All tasks
  - Hooks
  - Stages
  - Notification templates
  - Document templates
- General Settings

The main area is titled "Languages". It displays a table with three rows of data:

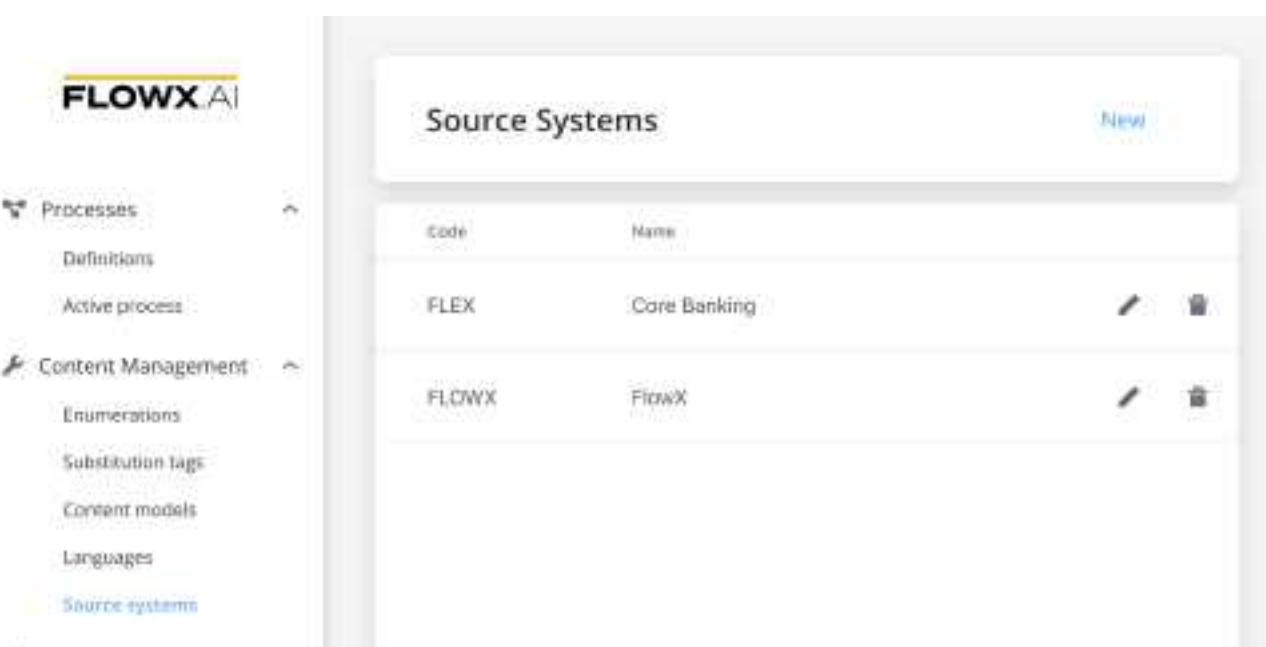
| Code | Name     | Default  | Action  |
|------|----------|--|---|
| ro   | Romanian | <span style="background-color: orange;">Default</span> | <span style="color: orange;">Edit</span> <span style="color: orange;">Delete</span> |
| en   | English  | <span style="background-color: green;">Primary</span>  | <span style="color: green;">Edit</span> <span style="color: green;">Delete</span>   |
| de   | German   | <span style="background-color: orange;">Default</span> | <span style="color: orange;">Edit</span> <span style="color: orange;">Delete</span> |

Was this page helpful?

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

If multiple

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



The screenshot shows the FlowX.AI interface with a sidebar on the left containing navigation links such as Processes, Content Management (with sub-links like Definitions, Active process, Enumerations, Substitution tags, Content models, Languages), and Source systems. The main area is titled 'Source Systems' and contains a table with two rows:

| Code  | Name         | Action      |
|-------|--------------|-------------|
| FLEX  | Core Banking | Edit Delete |
| FLOWX | FlowX        | Edit Delete |

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

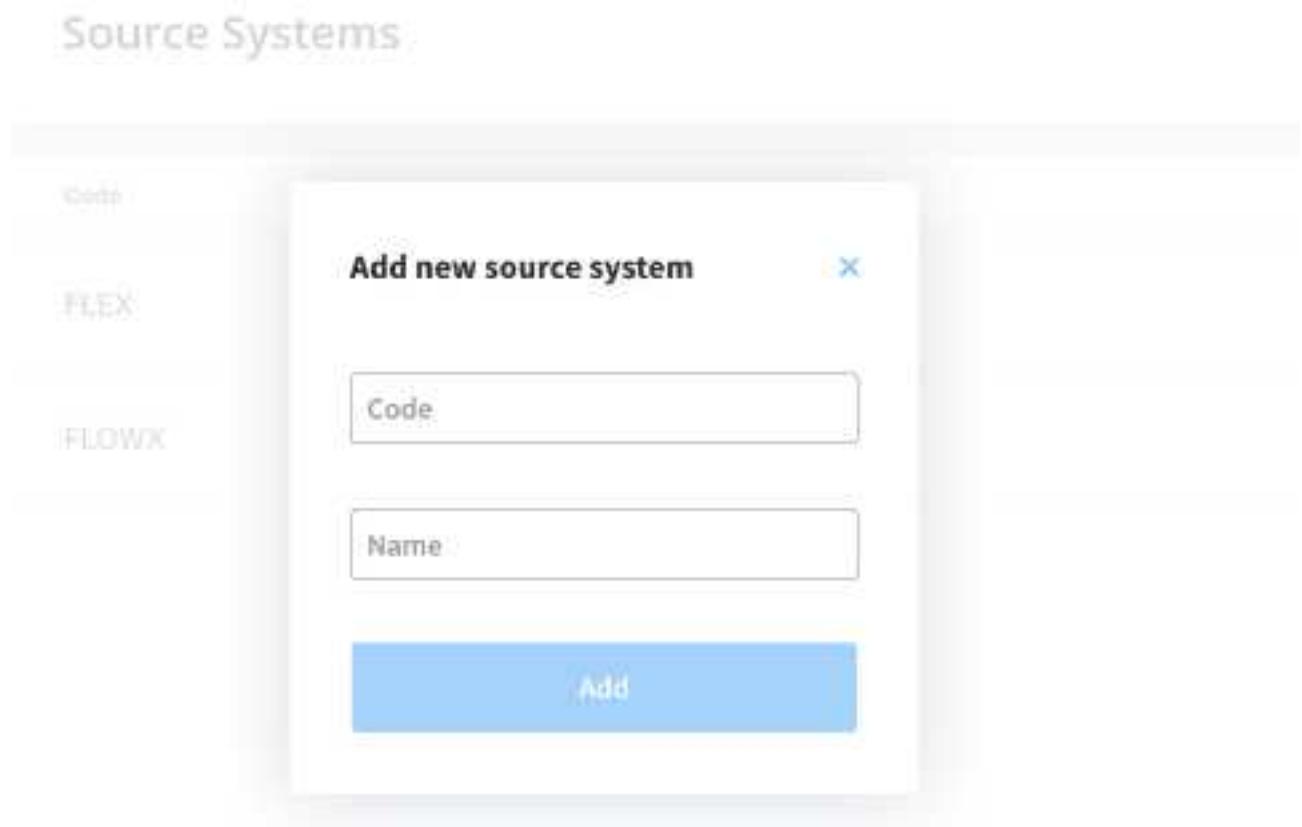
The fallback content to display on prerendering and select the **Content Management** tab.

2. Select **Source systems** from the list.

3. Fill in the necessary details:

- Code
- Name

4. Click **Add** after you finish.



Was this page helpful?

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

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

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

The screenshot shows the FLOWX.AI platform's Media library. On the left, there is a sidebar with navigation links for 'Priorities', 'Definitions', 'Active process', 'Previous instances', 'Failed process start', 'Content Management', 'Translations', 'Publication tags', 'Content results', 'Languages', 'Source systems', 'Media library' (which is currently selected), and 'Plugins'. At the bottom of the sidebar, there is a user profile for 'John Doe' with a yellow status icon. The main area is titled 'Media library' and contains a table with the following data:

| Preview | Name             | Type | Size    | Created At           | Added By | Action |
|---------|------------------|------|---------|----------------------|----------|--------|
|         | adidas           | webp | 0.21 MB | 04 Oct 2022, 9:34 AM | John Doe |        |
|         | bmw              | png  | 0.1 MB  | 04 Oct 2022, 9:34 AM | John Doe |        |
|         | dodge-challenger | jpg  | 0.18 MB | 04 Oct 2022, 9:34 AM | John Doe |        |
|         | viper_2          | png  | 0.02 MB | 04 Oct 2022, 9:34 AM | John Doe |        |
|         | dodge viper      | jpg  | 0.05 MB | 04 Oct 2022, 9:34 AM | John Doe |        |
|         | repais           | gif  | 0.74 MB | 04 Oct 2022, 9:34 AM | John Doe |        |
|         | lamborghini      | png  | 0.63 MB | 04 Oct 2022, 9:34 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:

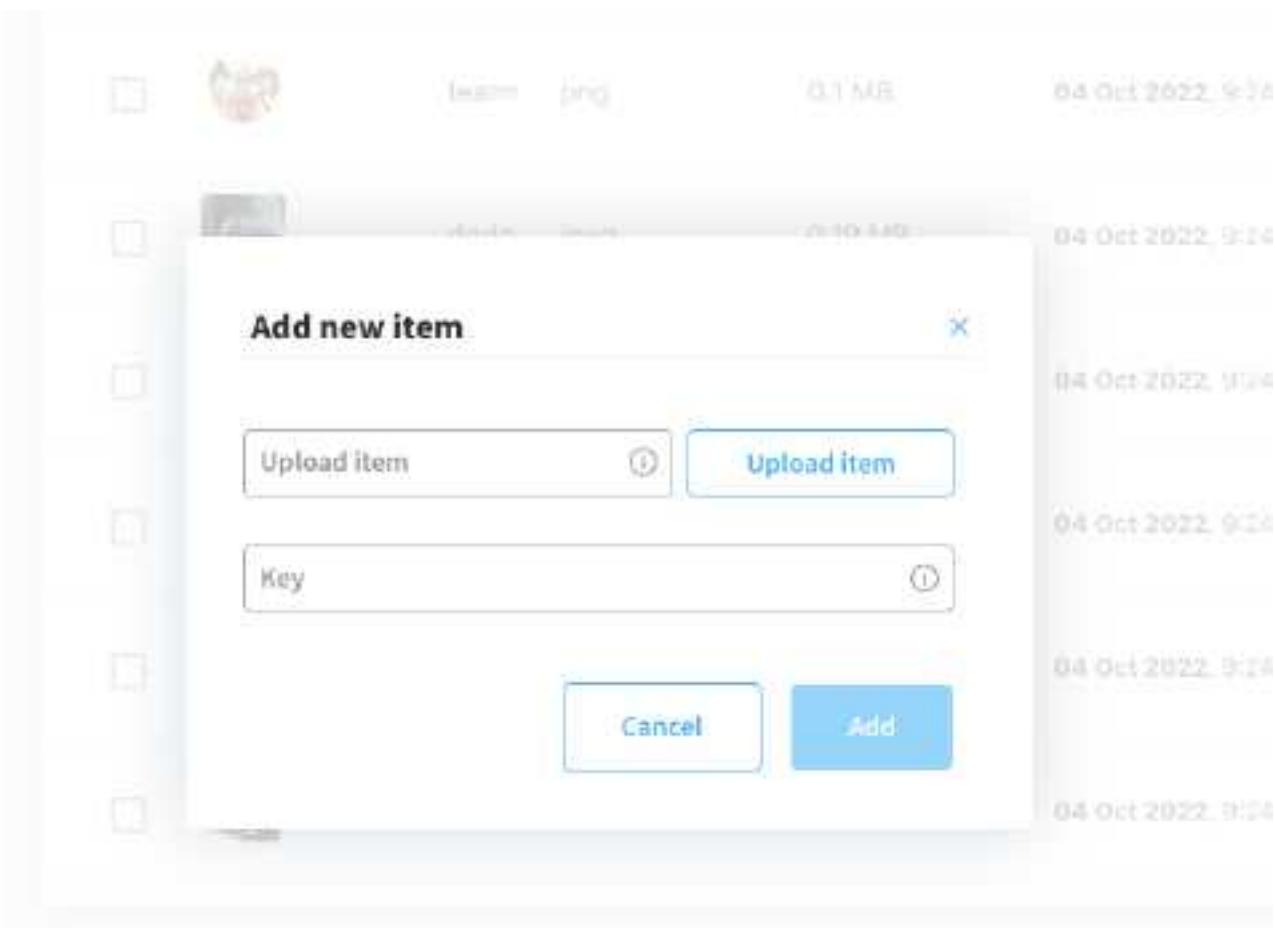
1. Open

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            |     |                  |      |           |                       |          |  |
|--------------------------|-----|------------------|------|-----------|-----------------------|----------|--|
| Search item by key:      |     |                  |      |           |                       | New Item |  |
| Preview                  | Key | Format           | Size | Edited at | Edited by             |          |  |
| <input type="checkbox"/> |     | sirus            | 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 | jpg  | 0.19 MB   | 04 Oct 2022, 9:24 AM  | John Doe |  |
| <input type="checkbox"/> |     | violet_2         | jpg  | 0.02 MB   | 04 Oct 2022, 9:24 AM  | John Doe |  |
| <input type="checkbox"/> |     | dodge viper      | jpg  | 0.05 MB   | 04 Oct 2022, 9:24 AM  | John Doe |  |
| <input type="checkbox"/> |     | implants         | gif  | 0.74 MB   | 04 Oct 2022, 9:24 AM  | John Doe |  |
| <input type="checkbox"/> |     | landrover        | png  | 0.005 MB  | 04 Oct 2022, 9:24 AM  | John Doe |  |

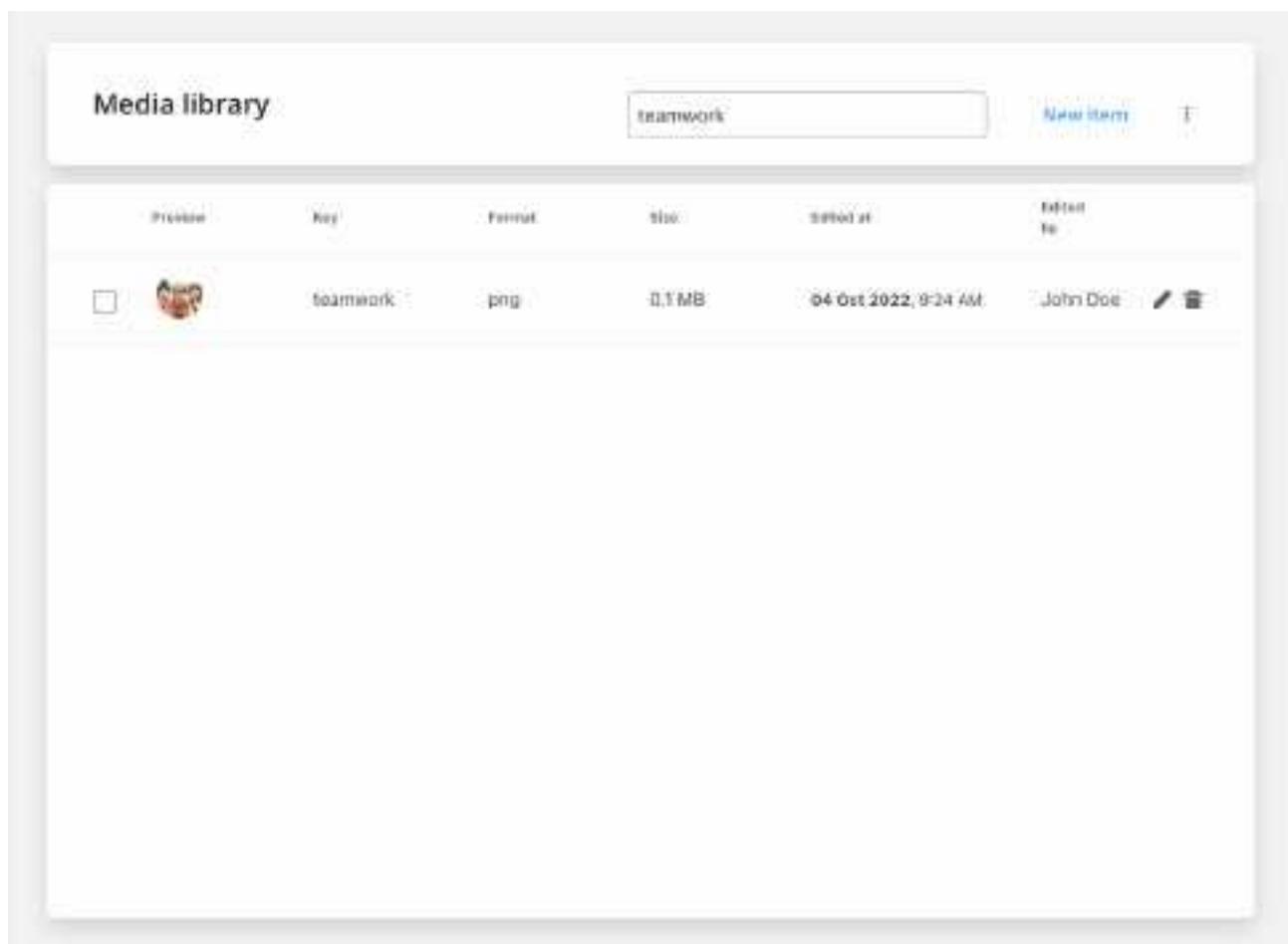
## Searching assets

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

| Media library            |     |          |      |           |                      |          |  |
|--------------------------|-----|----------|------|-----------|----------------------|----------|--|
| Search item by key:      |     |          |      |           |                      | 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).

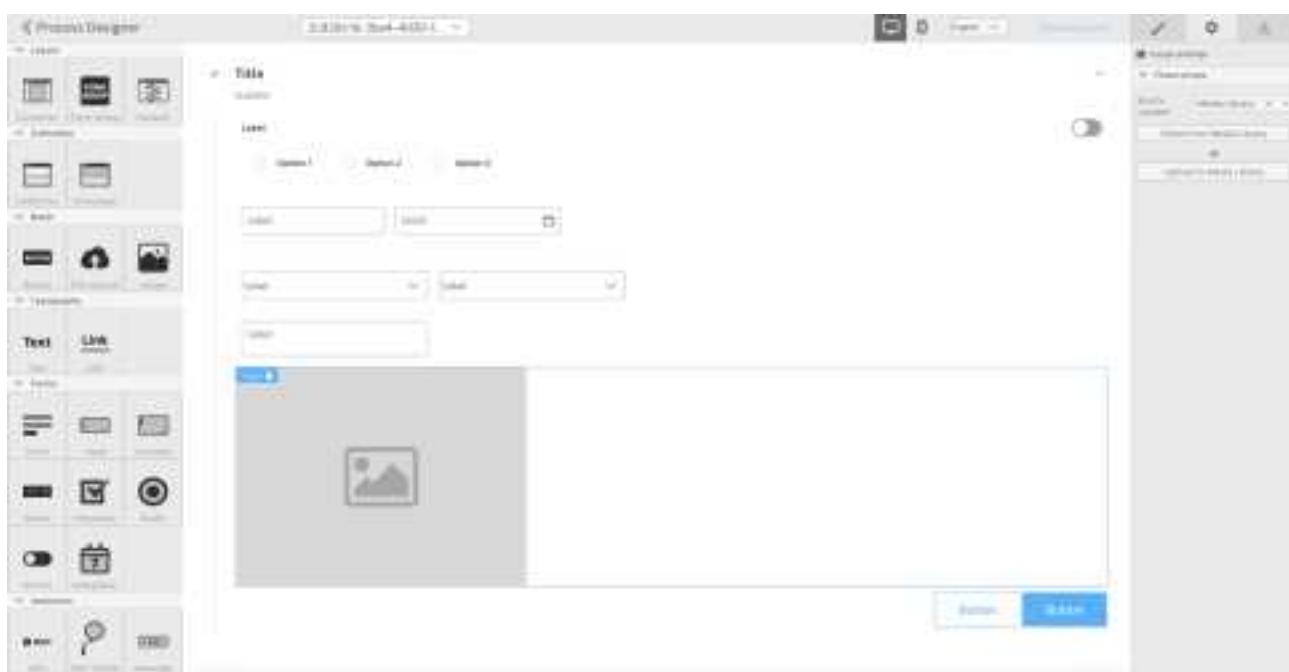


## Referencing assets in UI Designer

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

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

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

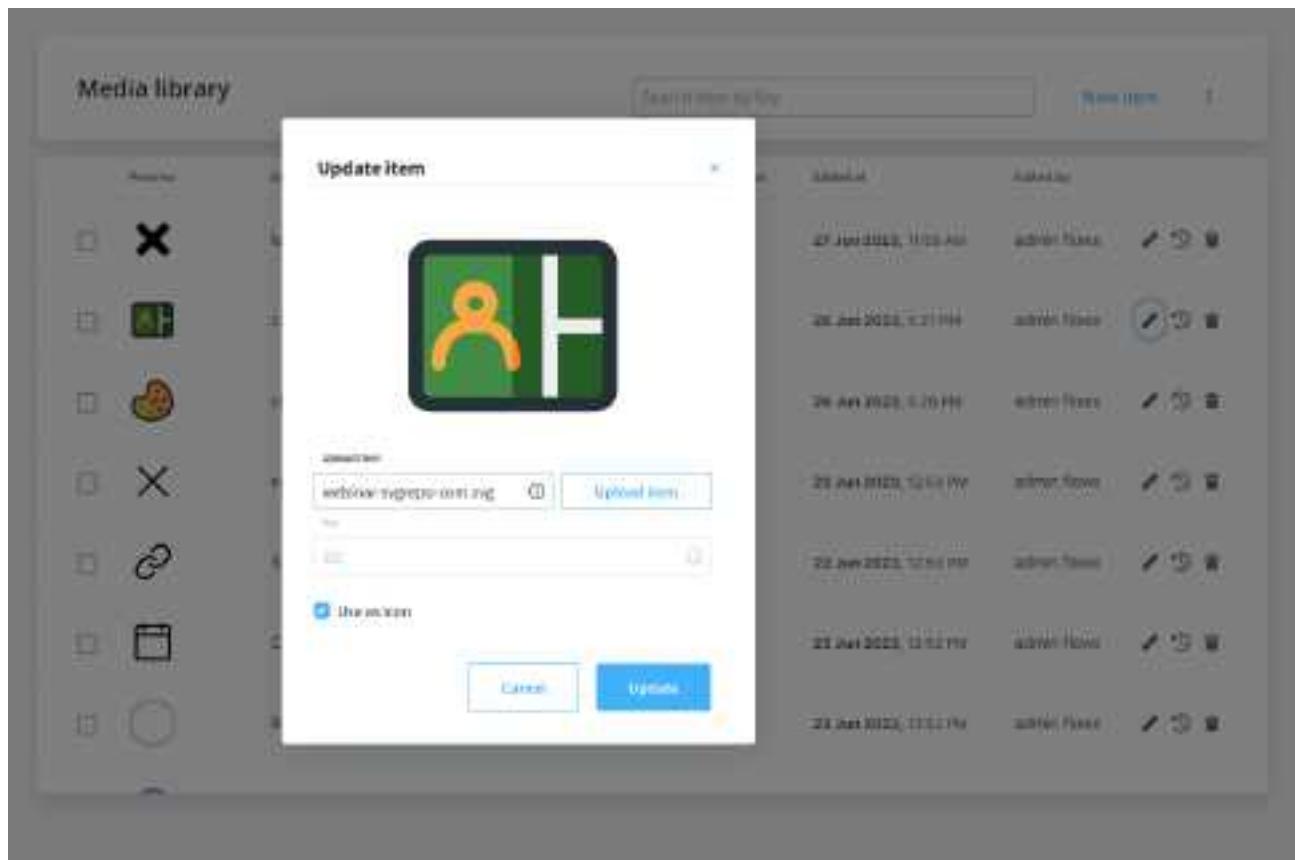


### !(INFO)

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

## Icons

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



### !(info)

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

### !(info)

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

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

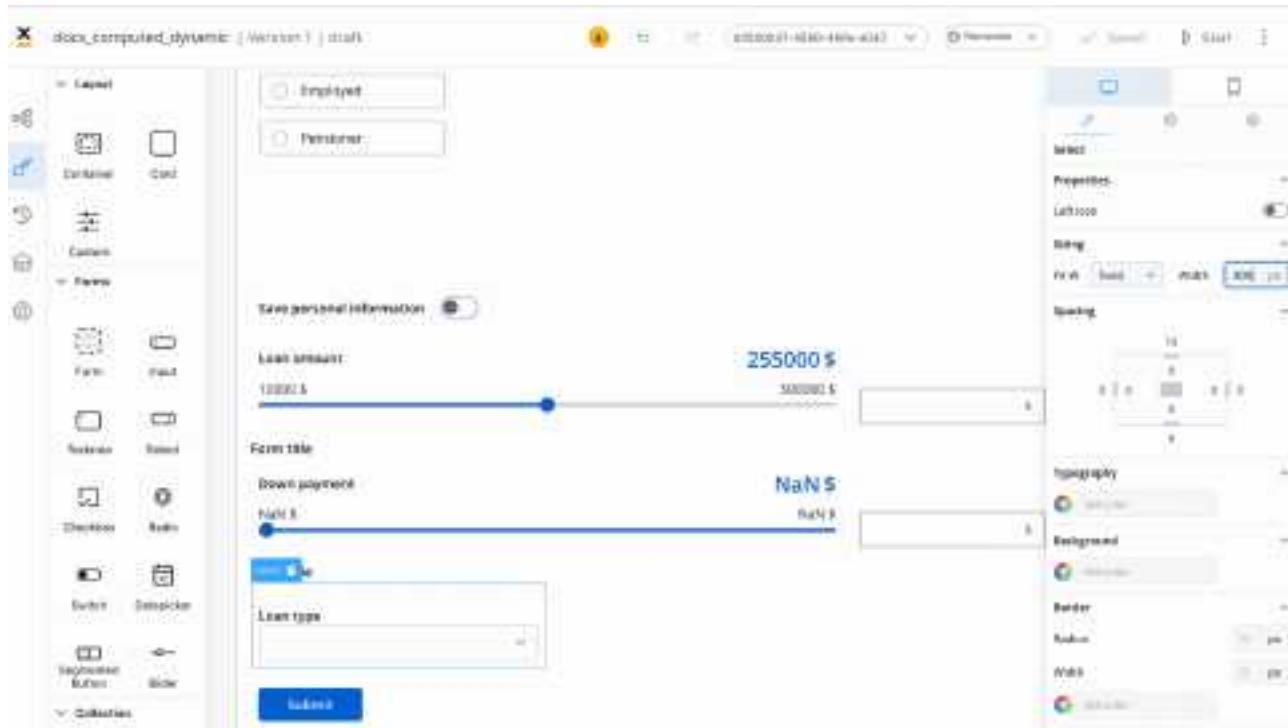
from icon sets that provide these size options or to resize icons proportionally to fit within these dimensions.

### ⚠ CAUTION

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

## Customization

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



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

» Input element

» Select element

» Buttons

## Export/import media assets

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

|                          | Preview | Key   | Format | Size    | Date                  |
|--------------------------|---------|-------|--------|---------|-----------------------|
| <input type="checkbox"/> |         | file1 | svg    | 0.76 KB | 31 May 2023, 11:39 AM |
| <input type="checkbox"/> |         | file2 | svg    | 0.67 KB | 31 May 2023, 11:39 AM |
| <input type="checkbox"/> |         | arr1  | svg    | 0.89 KB | 31 May 2023, 11:37 AM |
| <input type="checkbox"/> |         | arr2  | svg    | 0.97 KB | 31 May 2023, 11:37 AM |
| <input type="checkbox"/> |         | h1    | svg    | 0.41 KB | 30 May 2023, 3:06 PM  |
| <input type="checkbox"/> |         | h2    | svg    | 0.52 KB | 30 May 2023, 3:06 PM  |
| <input type="checkbox"/> |         | cf    | svg    | 0.2 KB  | 30 May 2023, 3:06 PM  |

## Import media assets

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

## Export all

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

The exported JSON structure will resemble the following example:

```
{  
  "images": [  
    {  
      "key": "cart",  
      "application": "flowx",  
      "filename": "maxresdefault.jpg",  
      "format": "jpeg",  
      "contentType": "image/jpeg",  
      "size": 39593,  
      "storagePath":  
        "https://d22tnnndi9lo60.cloudfront.net/devmain/flowx/cart/1681951544_maxresdefault.jpg",  
      "thumbnailStoragePath":  
        "https://d22tnnndi9lo60.cloudfront.net/devmain/flowx/cart/1681951544_maxresdefault.jpg",  
    },  
    {  
      "key": "pizza",  
      "application": "flowx",  
      "filename": "pizza.jpeg",  
      "format": "jpeg",  
      "contentType": "image/jpeg",  
      "size": 22845,  
      "storagePath":  
        "https://d22tnnndi9lo60.cloudfront.net/devmain/flowx/pizza/1681951544_pizza.jpeg",  
      "thumbnailStoragePath":  
        "https://d22tnnndi9lo60.cloudfront.net/devmain/flowx/pizza/1681951544_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**.

| Name                       | Status  | Created               | Edited by |
|----------------------------|---------|-----------------------|-----------|
| AutoTestProcess123E102864  | Running | 28 Apr 2023, 10:55 PM | QA-Felix  |
| AutoTestProcess123E770047  | Running | 28 Apr 2023, 10:55 PM | QA-Felix  |
| AutoTestProcess123E8811295 | Running | 28 Apr 2023, 10:55 PM | QA-Felix  |
| AutoTestProcess123E6401B3  | Running | 28 Apr 2023, 10:55 PM | QA-Felix  |

| Name         | Status    | Published             | Published by    |
|--------------|-----------|-----------------------|-----------------|
| test_123E_1  | Published | 28 Apr 2023, 10:55 PM | Silvia Digruber |
| test_123E_2  | Published | 28 Apr 2023, 10:55 PM | Silvia Digruber |
| test_123E_3  | Published | 28 Apr 2023, 10:55 PM | Silvia Digruber |
| test_123E_3% | Published | 28 Apr 2023, 10:55 PM | Silvia Digruber |

## 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 (with sub-links: Generic Parameters, Integration management, Licensing, Access management), Users, Roles, Groups, Audit Log, Platform status, and Admin Flows. The main area displays a table of generic parameters:

| Key              | Value   | Environment             | Action |
|------------------|---|-------------------------|--------|
| imagePlaceholder | https://autoartmodels.de/wp-content/uploads/2020/04/71721w-scaled.jpg   | qa                      |        |
| baseUri          | https://designer.qa.flowxai.dev/<br>https://designer.dev3.flowxai.dev/<br>https://designer.staging.flowxai.dev<br>https://designer.demo.flowxai.dev | dev3<br>staging<br>demo |        |
| envfilePath      | https://admin.devmain.flowxai.dev/document/<br>https://admin.qa.flowxai.dev/document/<br>https://admin.demo.flowxai.dev/document/                   | devmain<br>qa<br>demo   |        |
| test_key         | test_DEV<br>test_QA   | dev<br>qa               |        |

## Using generic parameters

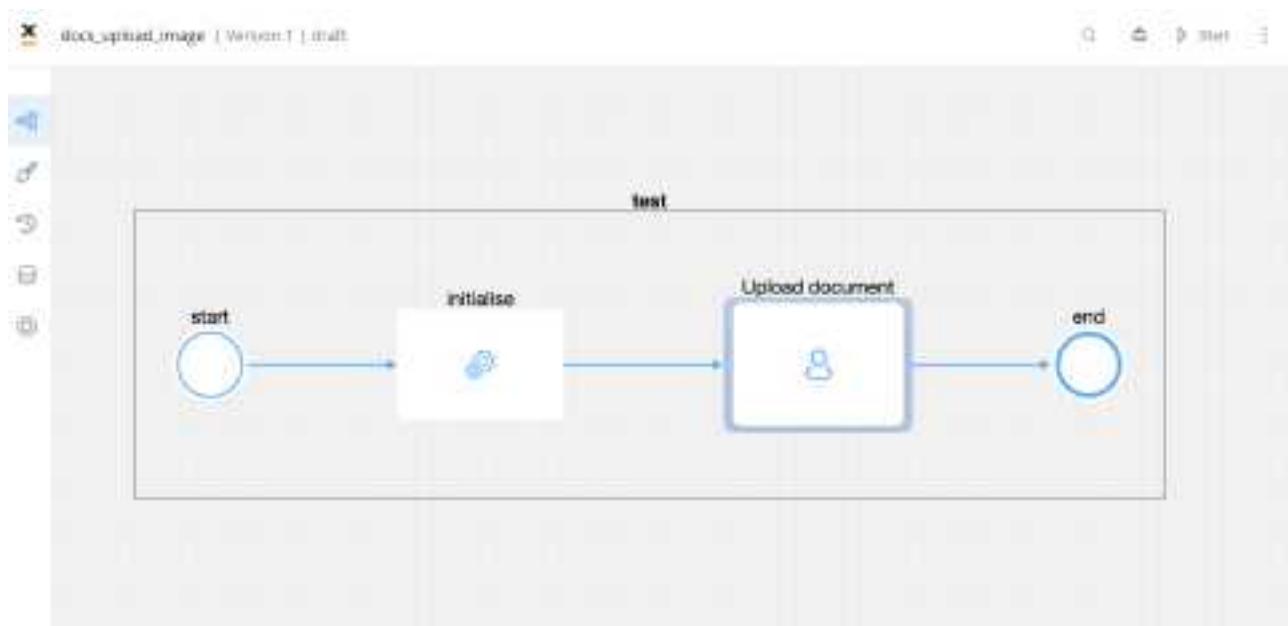
### Use case

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



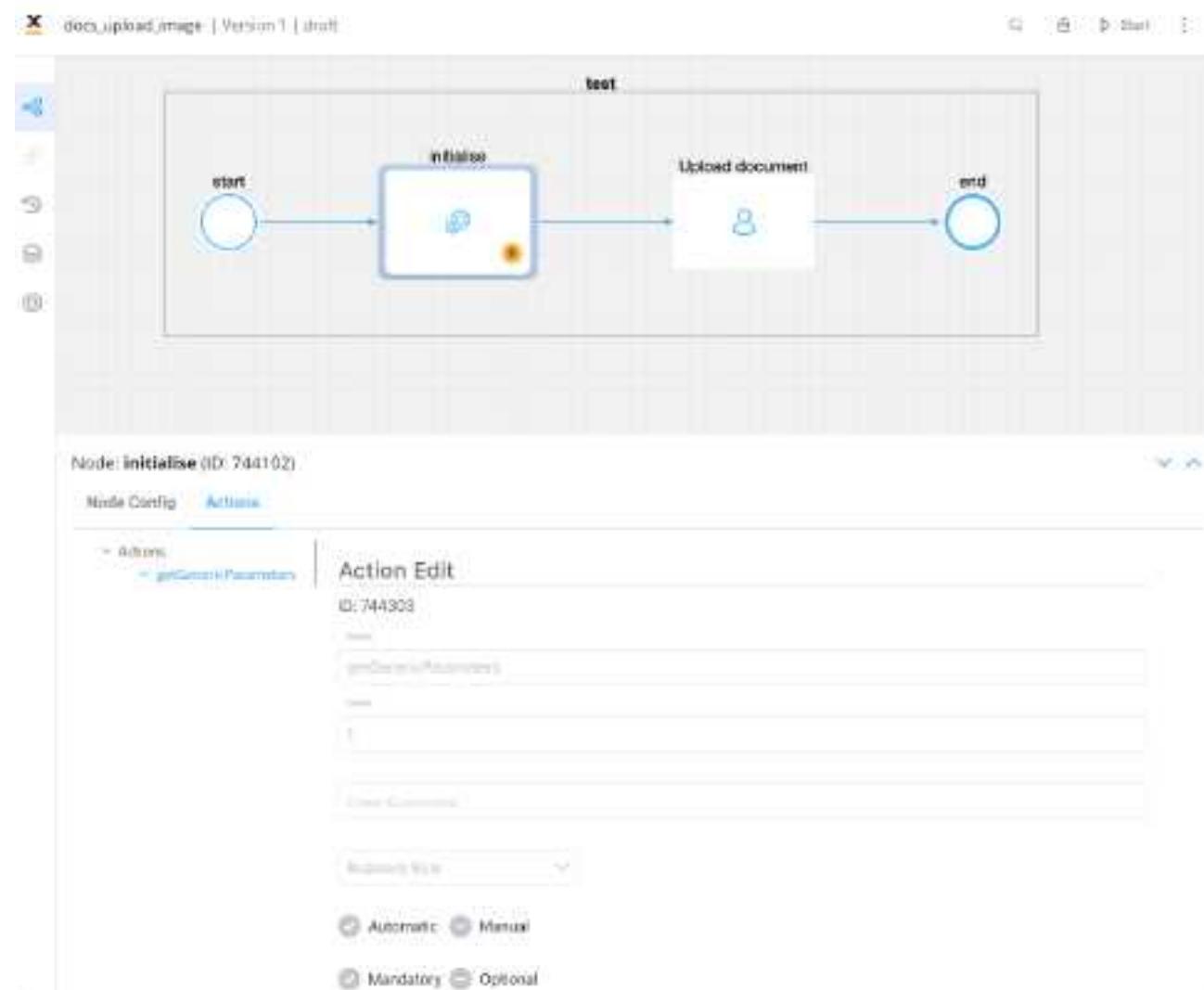
## Configuring the task node

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



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".

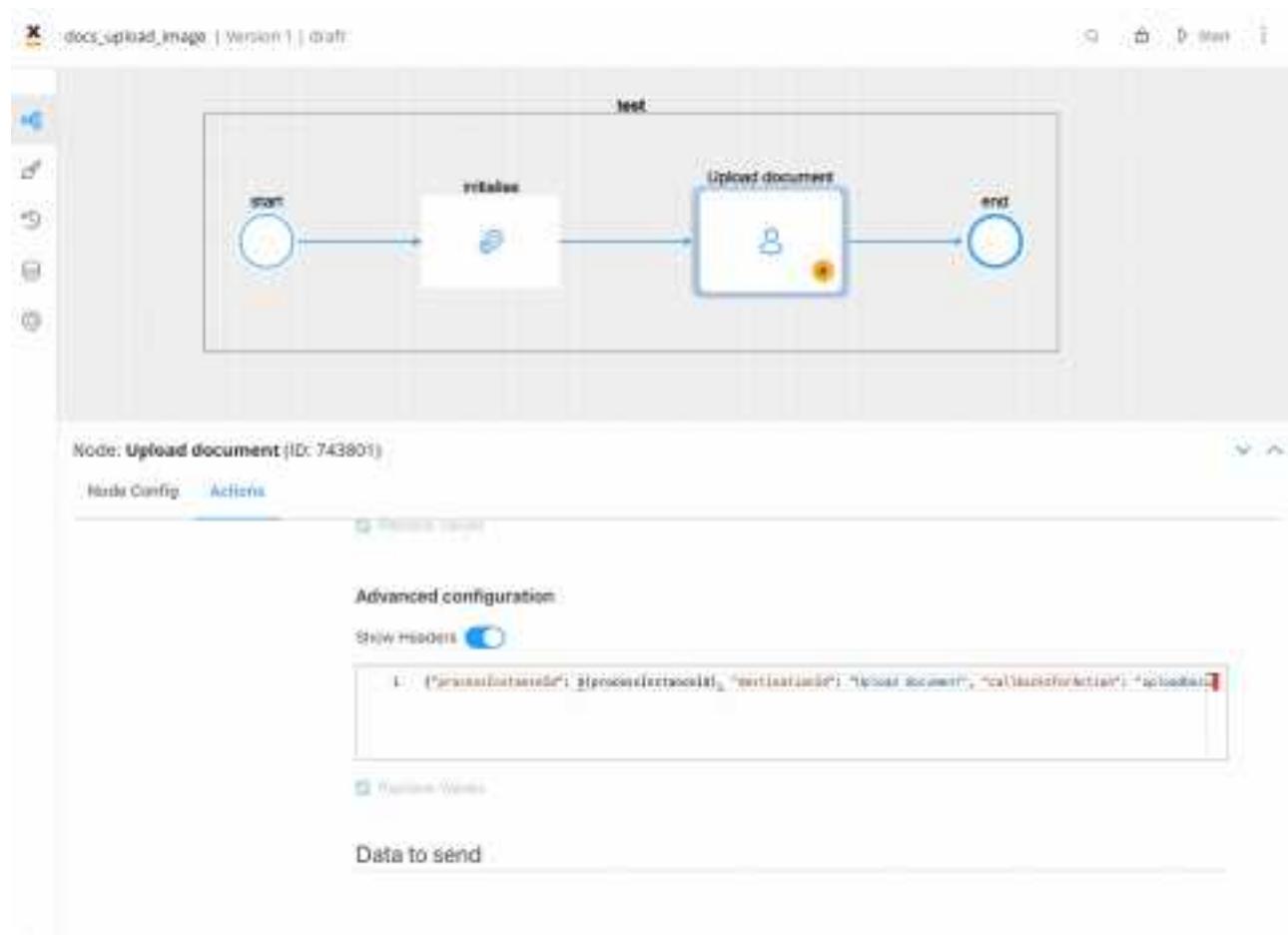
In summary, the code seems to be processing an uploaded document by checking its download path, constructing a full file path, and updating the document object with the new file path. Finally, it outputs the updated document object.

### 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               | Action             | Event              | Event Type | Subject Identifier                     | Process | Details                  |
| 20 Jun 2023, 12:07 PM | john.doe@email.com | Process Initiation | Process Initiation | View       | 6a2d9e50fa-0001-42f1-a10f-1a1b2f8800ff |         | <a href="#">View Log</a> |
| 20 Jun 2023, 12:07 PM | john.doe@email.com | Process Initiation | Process Initiation | Start      | 6a2d9e50fa-0001-42f1-a10f-1a1b2f8800ff |         | <a href="#">View Log</a> |
| 20 Jun 2023, 11:59 AM | john.doe@email.com | Process Monitoring | Process Definition | Created    | name_of_the_process                    |         | <a href="#">View Log</a> |
| 20 Jun 2023, 11:57 AM | john.doe@email.com | BPNDI Diagram      | Read               | 134MB      | model_name                             |         | <a href="#">View Log</a> |
| 20 Jun 2023, 11:57 AM | john.doe@email.com | BPNDI Diagram      | Read               | Updated    | test.bpmn                              |         | <a href="#">View Log</a> |
| 20 Jun 2023, 11:55 AM | john.doe@email.com | BPNDI Diagram      | Active             | Created    | name_of_the_action                     |         | <a href="#">View Log</a> |
| 20 Jun 2023, 11:55 AM | john.doe@email.com | BPNDI Diagram      | Connector          | Deleted    | 6a2d9e50fa-0001-42f1-a10f-1a1b2f8800ff |         | <a href="#">View Log</a> |
| 20 Jun 2023, 11:55 AM | john.doe@email.com | BPNDI Diagram      | Connector          | Created    | 6a2d9e50fa-0001-42f1-a10f-1a1b2f8800ff |         | <a href="#">View Log</a> |

## 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 |          |                    |                      |                  |                                      |   |   |      |      |
|------------|----------|--------------------|----------------------|------------------|--------------------------------------|---|---|------|------|
| User       | Subject  | Event              | Timestamp            | Action           | Details                              | Event ID  | URL   | Body | More |
| John Doe   | John Doe | View               | 2023-01-20T10:00:00Z | Process Instance | Start                                | 00000000-0000-0000-0000-000000000000            | /processes/00000000-0000-0000-0000-000000000000                           |      |      |
| John Doe   | John Doe | Process Instance   | 2023-01-20T10:00:00Z | Start            | 00000000-0000-0000-0000-000000000000 | /processes/00000000-0000-0000-0000-000000000000 |   |      |      |
| John Doe   | John Doe | Process Definition | 2023-01-20T10:00:00Z | Created          | Normal                               | Normal_Process                                  | /processes/Normal_Process   |      |      |
| John Doe   | John Doe | Node               | 2023-01-20T10:00:00Z | Update           | node_name                            | node_name                                       | /processes/Normal_Process/nodes/node_name                                 |      |      |
| John Doe   | John Doe | Node               | 2023-01-20T10:00:00Z | Update           | node_id                              | node_id   | /processes/Normal_Process/nodes/node_id                                   |      |      |
| John Doe   | John Doe | Action             | 2023-01-20T10:00:00Z | Created          | www_my_fav_action                    | www_my_fav_action                               | /processes/Normal_Process/actions/www_my_fav_action                       |      |      |
| John Doe   | John Doe | Connector          | 2023-01-20T10:00:00Z | Delete           | 00000000-0000-0000-0000-000000000000 | 00000000-0000-0000-0000-000000000000            | /processes/Normal_Process/connectors/00000000-0000-0000-0000-000000000000 |      |      |
| John Doe   | John Doe | Connector          | 2023-01-20T10:00:00Z | Created          | 00000000-0000-0000-0000-000000000000 | 00000000-0000-0000-0000-000000000000            | /processes/Normal_Process/connectors/00000000-0000-0000-0000-000000000000 |      |      |

- 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

The screenshot shows a modal window titled "Audit log details" over a background of audit log entries. The modal contains the following information:

Event: ui designer, created, 06 Oct 2022 at 6:01 PM  
Url: <https://admin.qa.flowxai.dev/api/nodes/716516/templates>

Body:

```
{ "uiTemplateParentId":6292714, "componentIdentifier": "FORM_GROUP", "type": "FL06X", "order": 0, "key": "", "platformsDisplayOptions": [{}], "style": null, "flexLayout": { "fxLayout": "column", "fxLayoutAlign": "start stretch", "fxLayoutGap": "0px" }, "className": null, "platform": "DEFAULT", "templateConfigId": null, "formFieldId": null } ] }
```

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:

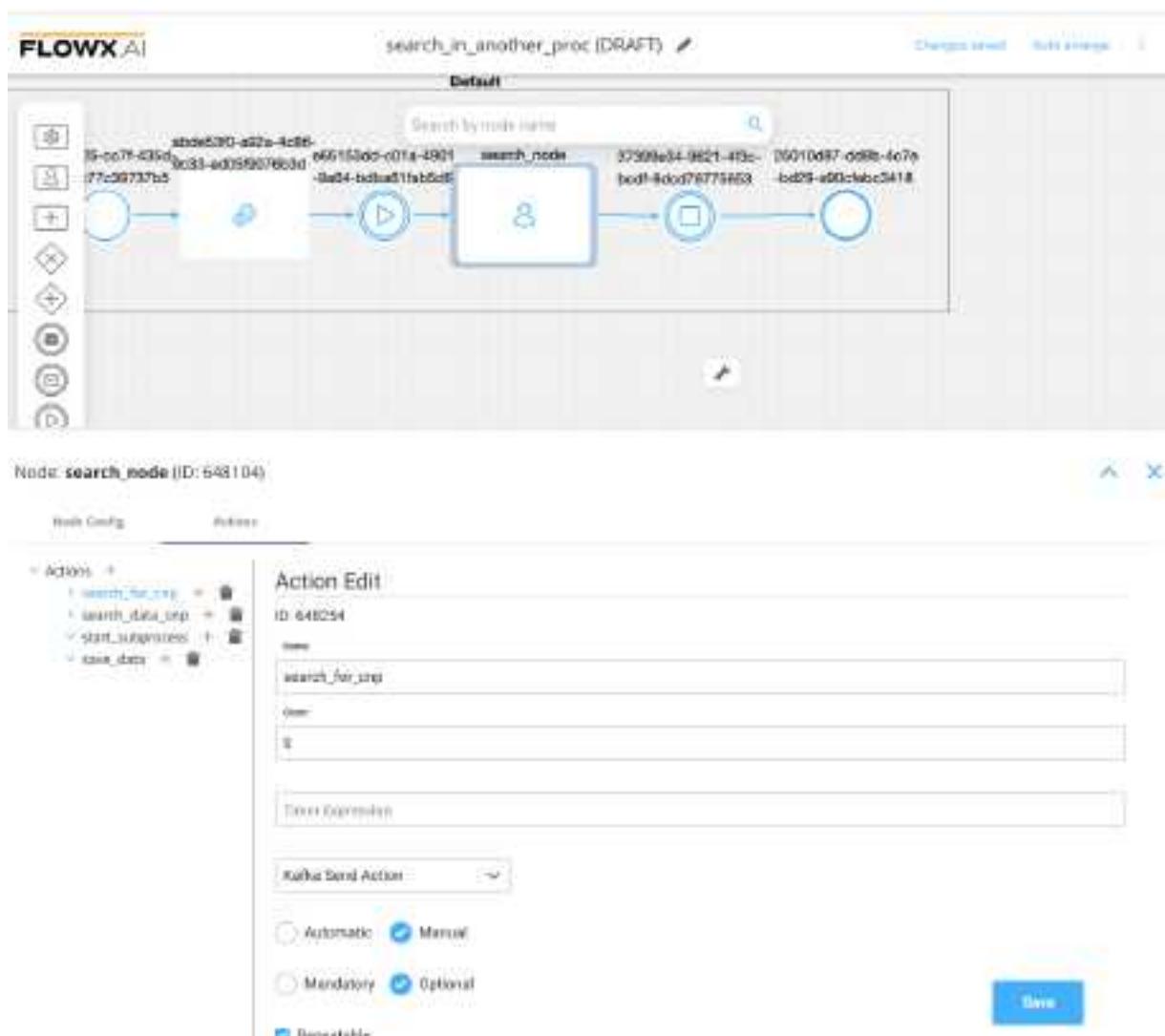
- search for data in other processes
- display results about other processes where the search key was found

## 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,qs,data,search,v1

**Message**

```
1  {
2      "searchKey": "application.clientIdentificationData.lastName",
3      "value": "SearchValue2",
4      "processDefinitionName": ["elvco_add_date_process"],
5      "processStartDateAfter": "2022-08-24T13:31:47.912524Z"
6  }
```

**Advanced configuration**Show Headers: 

```
1  {"processInstanceId": ${processInstanceId}, "destinationId": "search_node", "callbackForAction": "search_for_step"}
```

**Data to send**

searchValue2

**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.

```
processDefinitionSearchInInstance.json Active process count: 448169 - 0
```

```
records 0 items 0 suppressions 0

processDefinitionId: 641918
+ searchKey: "application.client.name"
  result: []
  totalCountAvailable: false
  pageNumber: 1
  tokenID: 641918
  smartValue: "Next best"
  sortByUUID: "2023-07-27T00:00:00Z"
  webSocketPath: "/ws/updated/process"
  processDefinitionURL: "http://localhost:8080/api/processes/641918"
  webSocketAddress: "ws://localhost:8080/api/processes/641918"
```

! 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):

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

## » Search data service setup guide

## Was this page helpful?

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

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

# Angular project requirements

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

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

### INFO

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

### CAUTION

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

### CAUTION

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

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

## Installing the library

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

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

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

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

## Using the library

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

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

## Using Paperflow web components

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

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

### !(INFO)

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

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

## Theming

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

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

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

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

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

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

```
  ),  
));
```

## Authorization

### ⓘ INFO

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

`Authorization: Bearer ...`).

### ⓘ NOTE

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

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

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

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

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

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

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

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

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

```
// example of a custom validator that restricts data
selection to
// the current or the previous year

currentOrLastYear: function currentOrLastYear(AC:
AbstractControl): { [key: string]: any } {
  if (!AC) {
    return null;
  }

  const yearDate = moment(AC.value, YEAR_FORMAT, true);
  const currentDateYear = moment(new
Date()).startOf('year');
  const lastYear = moment(new Date()).subtract(1,
'year').startOf('year');

  if (!yearDate.isSame(currentDateYear) &&
!yearDate.isSame(lastYear)) {
    return { currentOrLastYear: true };
  }

  return null;
}
```

### ⚠ CAUTION

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

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

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

### Parameters:

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

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

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

| Name    | Description  | Type    | Mandatory | Default value |   |
|---------|--|---------|-----------|---------------|---|
| isDraft | <p>When true allows starting a process in draft state.</p> <p>*Note that isDraft = true requires that processName be the <b>id</b> (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 table titled "Activities" with the following columns: Title, Stage, Assignee, Status, Priority, and Last updated. There are 10 rows of data, each representing a task. The tasks are categorized into two stages: Onboarding and PF. Most tasks are assigned to "shru@flowx.ai" and are in the "STARTED" status. The last updated time for all tasks is 14.12.2021, 11:02.

| Title             | Stage      | Assignee      | Status  | Priority | Last updated      |
|-------------------|------------|---------------|---------|----------|-------------------|
| Sivu Seminars New | Onboarding | shru@flowx.ai | STARTED | 3        | 14.12.2021, 11:02 |
| Sivu Seminars     | PF         | shru@flowx.ai | STARTED | 1        | 14.12.2021, 11:02 |
| Sivu Seminars New | Onboarding | shru@flowx.ai | STARTED | 3        | 14.12.2021, 11:01 |
| Sivu Seminars     | PF         | shru@flowx.ai | STARTED | 1        | 14.12.2021, 11:01 |
| Sivu Seminars     | PF         | shru@flowx.ai | STARTED | 1        | 14.12.2021, 11:01 |
| Sivu Seminars New | Onboarding | shru@flowx.ai | STARTED | 3        | 14.12.2021, 11:01 |
| Sivu Seminars New | PF         | shru@flowx.ai | STARTED | 3        | 14.12.2021, 11:00 |
| Sivu Seminars     | PF         | shru@flowx.ai | STARTED | 1        | 14.12.2021, 11:00 |
| Sivu Seminars New | Onboarding | shru@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      | <a href="https://yc">https://yc</a> |
| 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.<br>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 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 tables: "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 | [Edit] [Delete] |
| d... | 1       | 30 Sep 2022, 2:32 PM  | andrei anali   | [Edit] [Delete] |
| t... | 1       | 30 Sep 2022, 11:15 AM | QA FlowX       | [Edit] [Delete] |
| T... | 1       | 30 Sep 2022, 10:29 AM | QA FlowX       | [Edit] [Delete] |

**Published:**

| Name | Version | Published at          | Published by   | Actions         |
|------|---------|-----------------------|----------------|-----------------|
| A... | 1       | 03 Oct 2022, 8:12 AM  | QA FlowX       | [Edit] [Delete] |
| c... | 10      | 30 Sep 2022, 3:08 PM  | Silviu Grigore | [Edit] [Delete] |
| d... | 1       | 30 Sep 2022, 10:34 AM | Bogdan Ionescu | [Edit] [Delete] |
| T... | 1       | 30 Sep 2022, 10:18 AM | QA FlowX       | [Edit] [Delete] |

## 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 4 "clientType": "FF",
2 5 "documentsList": [
3 6   {
4 7     "customId": "123456",
5 8     "templateName": "test_doc",
6 9     "language": "en",
7 10    "data": {
8 11      "key": "value"
9 12    },
10 13    "includeBarcode": true
11 14  }
12 15 ]
```

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

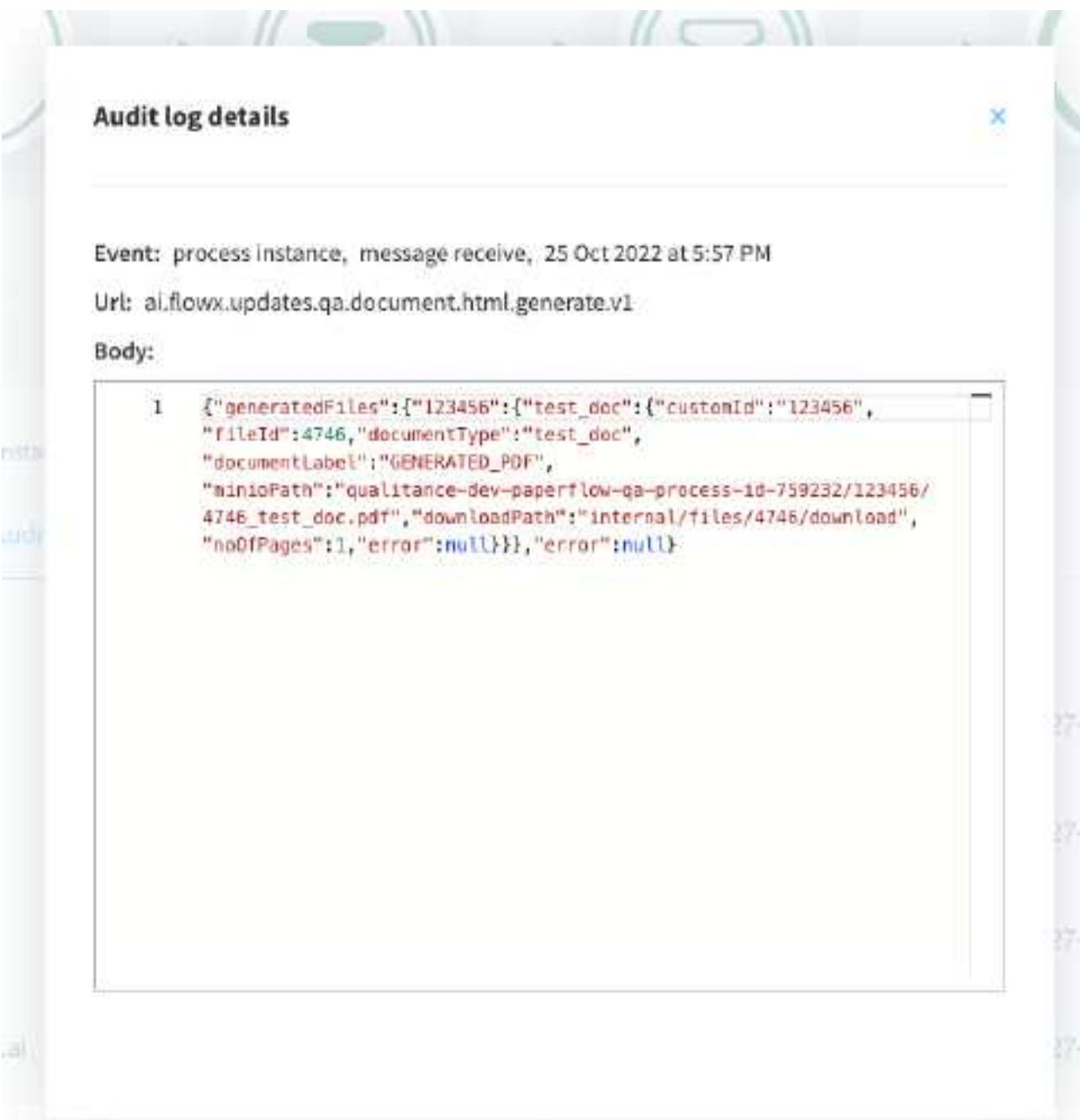
The engine listens for messages on topics with specific naming patterns. Make sure to use an outgoing topic name that matches the pattern configured in the engine.

## 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 web browser window titled "Audit log details". The window contains event logs. The first log entry shows the following details:

Event: process instance, message receive, 25 Oct 2022 at 5:57 PM  
Url: ai.flowx.updates.qa.document.html.generate.v1  
Body:

```
1 {"generatedFiles": {"123456": {"test_doc": {"customId": "123456",  
"fileId": 4746, "documentType": "test_doc",  
"documentLabel": "GENERATED_PDF",  
"minioPath": "qualitance-dev-paperflow-qa-process-id-759232/123456/  
4746_test_doc.pdf", "downloadPath": "internal/files/4746/download",  
"noOfPages": 1, "error": null}}}, "error": null}
```

Values expected in the event body:

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

Example of generated file response received on

KAFKA\_TOPIC\_DOCUMENT\_GENERATE\_HTML\_IN topic :

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

```
        "noOfPages": 1,  
        "error": null  
    },  
},  
"error": null  
}
```

Was this page helpful?

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

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

## Configuring HTML templates

### Text parameters

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

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

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

Data specifications:

```
{
  "data": {
    "companyName": "Test Company SRL",
    "cui": "R01234567"
  }
}
```

## Dynamic tables - repeatable rows

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

**FlowX AI** provides a detailed description of the offer. 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 1 | 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 2 | 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 |
|--|---------|
| <b>General Config</b><br>Node name<br><input type="text" value="upload_file"/><br>Can go back? <input checked="" type="checkbox"/>   |         |
| <b>Flow Names</b><br>Leave empty if this node is to be included in all flows<br><input type="text"/>   |         |
| <b>Swimlane</b><br><input type="text" value="Default"/>  |         |
| <input type="text" value="Stage"/>   |         |
| <b>Response Timeout</b><br><input type="text" value="Response Timeout (PT30S)"/>   |         |
| <b>Data stream topics</b><br><input checked="" type="radio"/> Custom <input type="radio"/> From integration<br><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

Home  
upload\_file

User  
T

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.flows.in.qa.document.persist.v1
```

 Replace Values

### Document Type

```
BULK
```

 Replace Values

### Folder

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

 Replace Values

### Advanced configuration

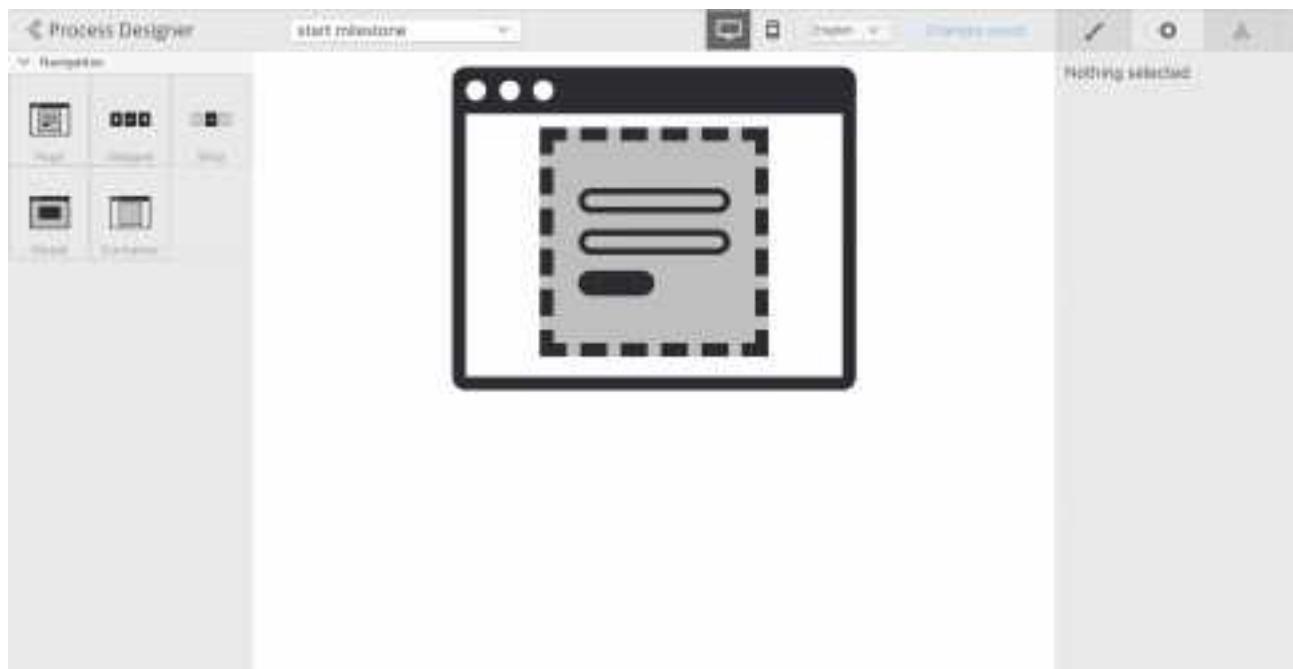
Show Headers 

## 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.

The screenshot shows a web-based audit log interface. At the top, it says "Audit log details". Below that, it displays an event: "Event: process instance, message receive, 13 Oct 2022 at 2:46 PM". It also shows the URL: "Url: ai.flowx.updates.qa.document.persist.v1". Under "Body:", there is a JSON object:

```
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 interface. At the top, there's a navigation bar with the FLOWX.AI logo, a project name "pdf\_to\_jpeg (DRAFT)", and buttons for "Change owner" and "Edit workspace". Below the navigation is a toolbar with various icons for creating nodes like start, end, parallel, and sequence. The main workspace displays a workflow diagram titled "Default". The diagram consists of four nodes: "start", "convert\_request", "convert\_reply", and "end", connected sequentially. The "convert\_request" node is highlighted with a yellow border. Below the workspace is a modal dialog titled "Node: convert\_request (ID: 725262)". The dialog has two tabs: "Node Config" (selected) and "Actions". Under "Actions", there's a section for "Action Edit" with the ID "725652". It includes fields for "Name" (set to "3261a153b54749c8b21d5cc7a00070d0") and "Index" (set to "1"). There's also a "Time Expression" field which is empty. A dropdown menu shows "Kafka Send Action" is selected. Below the dropdown are two radio buttons: "Automate" (checked) and "Manual", and two checkboxes: "Mandatory" (checked) and "Optional". A "Save" button is located at the bottom right of the dialog.

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

Parameters

Custom From integration

Topics

```
ai.flowx.images.document.convert.v1
```

4. Fill in the body of the message request:

Message

```
t {"fileId": 1154, "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 in

5

b40-S

b40-S

b40-S

7077.01

process instance

2023

13000000-0000-4800-0b40-8

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, there's a toolbar with icons for file operations like 'New', 'Open', 'Save', etc., and a search bar labeled 'Search by node name'. Below the toolbar is a breadcrumb navigation bar showing 'split\_docs (DRAFT)'. On the left, a vertical sidebar contains icons for creating various node types: start, end, decision, parallel, sequence, kafka send, kafka receive, http, and email. The main workspace displays a workflow diagram titled 'Default' with four nodes: 'start', 'split\_request' (highlighted in orange), 'split\_reply', and 'end'. Arrows connect them in sequence. Below the workspace is a node details panel for 'Node: split\_request (ID: 727240)'. It has tabs for 'Node Config' and 'Actions'. The 'Actions' tab is active, showing a sub-tree under 'Actions' with 'split\_request' selected. A 'Action Edit' dialog is open for the 'split\_request' action. The dialog fields include:

- ID: 726253
- name: split\_request
- order: 1
- Timer Expression: (empty)
- Action Type: Kafka Send Action
- Activation: Automatic (checked)
- Requirement: Mandatory (checked)

A blue 'Save' button is at the bottom right of the dialog.

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": 14747,  
3   "parts": [  
4     {  
5       "documentType": "BULK",  
6       "customId": "1234_758764",  
7       "pageNo": [1,2]  
8     }  
9   ]  
10 }
```

### Advanced configuration

Show Headers

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

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

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

### INFO

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

`KAFKA_TOPIC_DOCUMENT_SPLIT_IN` - default value:

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

`KAFKA_TOPIC_DOCUMENT_SPLIT_OUT` - default value:

`ai.flowx.updates.qa.document.split.v1` - this is the topic on which the engine expects the reply

The above examples of topics are extracted from an internal testing environment. When setting topics for other environments, follow this pattern:

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

### CAUTION

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

## Receiving the reply

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

### Node: **split\_reply** (ID: 728352)

#### Node Config

##### General Config

Node name

Can go back?

Swimlane



##### Response Timeout

##### Data stream topics

Topic Name

Key Name



The response body will contain the following values:

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

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) 

Default



Start → request\_update → receive\_reply → End

Node: **request\_update** (ID: 763601)

Node Config Actions

+ Action: + update\_document + 

Action Edit

ID: 762018

Name: update\_document

Order: 1

Time Expression:

Kafka Send Action: 

Automatic  Manual

Mandatory  Optional

Repeatable

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

## Parameters

Custom From integration

### Topics

all\_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

The screenshot shows a modal window titled "Audit log details". Inside the modal, there is event information and a code block.

**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)

Action Edit

ID: 765241

Name: deleteDocument

Version: 1

Timer Expression:

Kafka Send Action

Automatic  Manual

Mandatory  Optional

Repeatable

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

Parameters

Custom From integration

Topics

```
at: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

The screenshot shows a modal window titled "Audit log details". Inside, there is one event entry. The event details are as follows:

**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_763411","fileId":4747,"documentType":"BULK",
  "error":null}
```

The background of the application shows a list of audit logs on the left side, with some entries partially visible.

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 Milestone 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

Autonrun Children?

Allow BACK on this action?

**Save**

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

## Parameters

### Topics

```
all,flowx,in,qn,document,persist,v1
```

 Replace Values

### Document Type

```
#(processInstanceId)
```

 Replace Values

### Folder

```
1234_5 #(processInstanceId)
```

 Replace Values

### Advanced configuration

 Show Headers

```
1 C#processInstanceId# 2#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.intg.document.url,v1
```

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

### Message

```
1 var {
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 More

Default

Search by node name:

Node: **receive\_reply** (ID: 766487) Edit X

Node Config

Substage:

Stage:

Response Timeout:

Data stream topics

Custom  From integration

Topic Name:  Key Name:

**!** 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_
  request&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 'Notification Templates' section within the FLOWX.AI application. The sidebar on the left is collapsed, showing the 'Plugins' section is currently active. The main area displays a table titled 'Notification Templates' with two sections: 'Drafts / In progress' and 'Published'. The 'Drafts / In progress' section contains three entries: 'Styling Template' (written by John Doe on 18 Oct 2022, 3:26 PM), 'Amazing Template' (written by Jane Doe on 18 Oct 2022, 3:21 PM), and 'Exquisite Template' (written by Bass Twinkles on 18 Oct 2022, 3:18 PM). The 'Published' section contains three entries: 'Cool Template' (written by John Doe on 10 Sep 2022, 4:30 PM), 'Awesome Template' (written by Jane Doe on 09 Aug 2022, 11:48 AM), and 'Creative Template' (written by Bass Twinkles on 08 Jun 2022, 11:11 AM). Each template entry includes a preview icon, a timestamp, the author's name, and edit and delete icons.

| Name               | Written       | Published at         | Maintained by |
|--------------------|---------------|----------------------|---------------|
| Styling Template   | John Doe      | 18 Oct 2022, 3:26 PM | John Doe      |
| Amazing Template   | Jane Doe      | 18 Oct 2022, 3:21 PM | Jane Doe      |
| Exquisite Template | Bass Twinkles | 18 Oct 2022, 3:18 PM | Bass Twinkles |

  
| Name | Written | Published at | Maintained by |
| --- | --- | --- | --- |
| Cool Template | John Doe | 10 Sep 2022, 4:30 PM | John Doe |
| Awesome Template | Jane Doe | 09 Aug 2022, 11:48 AM | Jane Doe |
| Creative Template | Bass Twinkles | 08 Jun 2022, 11:11 AM | Bass Twinkles |

## Configuring a template

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

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

The screenshot shows the FLOWX.AI platform interface for managing notifications. On the left, a sidebar navigation includes 'Processes', 'Content Management' (with 'Notification templates' selected), 'Plugins', 'General settings', and 'Platform status'. The main workspace is titled 'Notifications Template - ExampleTemplate'. It contains tabs for 'Body' (selected) and 'Data model'. Under 'Body', there are fields for 'Type' (set to 'Mail'), 'Forward on Kafka' (unchecked), 'Language' (set to 'Portuguese (Brazilian) (pt\_BR)'), and 'Subject' (set to 'Contract Billed'). Below these are 'Text' and 'HTML' editors. A preview window shows a message with a placeholder 'Hello {{input1}} {{input2}}' and a note: 'Objetivo: Detalhar o bônus para cada usuário.' At the bottom are 'Save' and 'Pusher' 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.

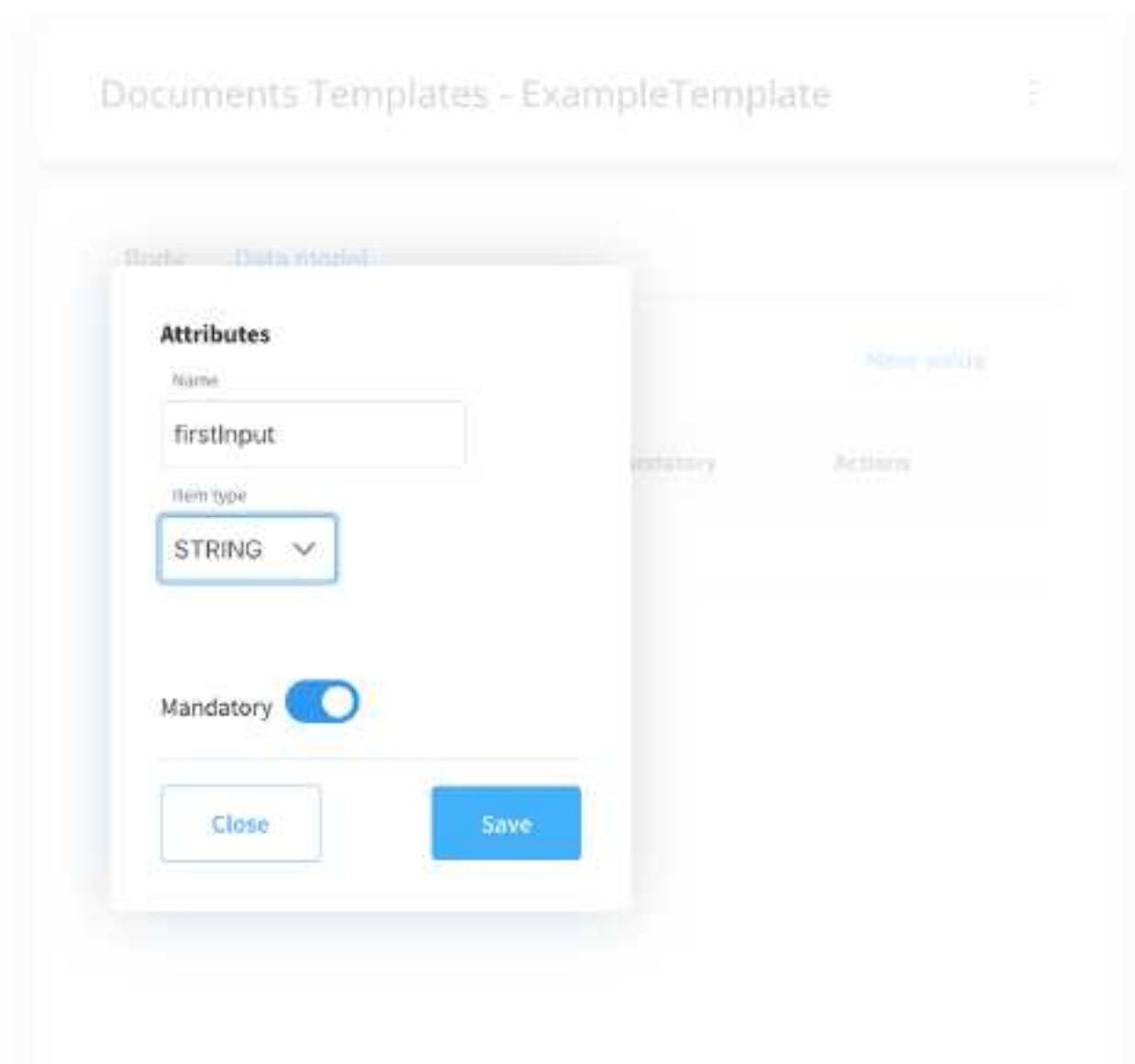
## Configuring the data model

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)



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

The screenshot shows the 'Notifications Templates - Test\_notification\_template' configuration page. On the left, a sidebar lists various system components: enumerations, substitution tags, content models, languages, source systems, and several sections under 'Plugins' (Task Manager, All tasks, Hooks, Stages), 'Notification templates' (selected), 'Document templates', and 'General Settings' (generic parameters, licensing, access management, users, roles). The main panel displays the template configuration for 'Test\_notification\_template'. It includes tabs for 'Body' (selected) and 'Data model'. Under 'Body', settings include 'Type' (set to 'MAIL'), a checked 'Forward on Kafka' option, 'Language' (set to 'Romanian-ro'), and a subject line ('Romanian-ru subject'). A 'Source' button is available to view the raw XML code. Below these settings is a WYSIWYG editor toolbar with various icons for text styling.

» [WYSIWYG Editor](#)

## Testing the template

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

The screenshot shows a configuration interface for a notification template. At the top, it says "Notifications Templates - Test\_notification\_template". Below that, there are two tabs: "Body" (which is selected) and "Data model".  
The "Body" tab contains the following fields:

- Type: A dropdown menu set to "MAIL".
- Forward on Kafka: An unchecked checkbox.
- Language: A dropdown menu set to "Romanian-rom".
- Romanian-rom subject: A text input field containing "#test\_data".

At the bottom of the "Body" tab, there is a "Source" button and a rich text editor toolbar with various icons like bold, italic, underline, etc. The text area of the rich text editor also contains "#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)

A screenshot of a user interface showing a list of notification templates. The list is organized into columns: 'Edited at', 'Edited by', and 'Published at'. A context menu is open over the second row, which corresponds to the entry '18 Oct 2022, 3:26 PM' and 'John Doe'. The menu options are: 'Publish template', 'Export template', 'Show history', and 'Delete'. The 'Delete' option is highlighted in red.

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

Was this page helpful?

## PLATFORM DEEP DIVE / Plugins / Custom Plugins / Notifications

# plugin / Using the plugin / Sending a notification

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

## Configuring the process

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

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

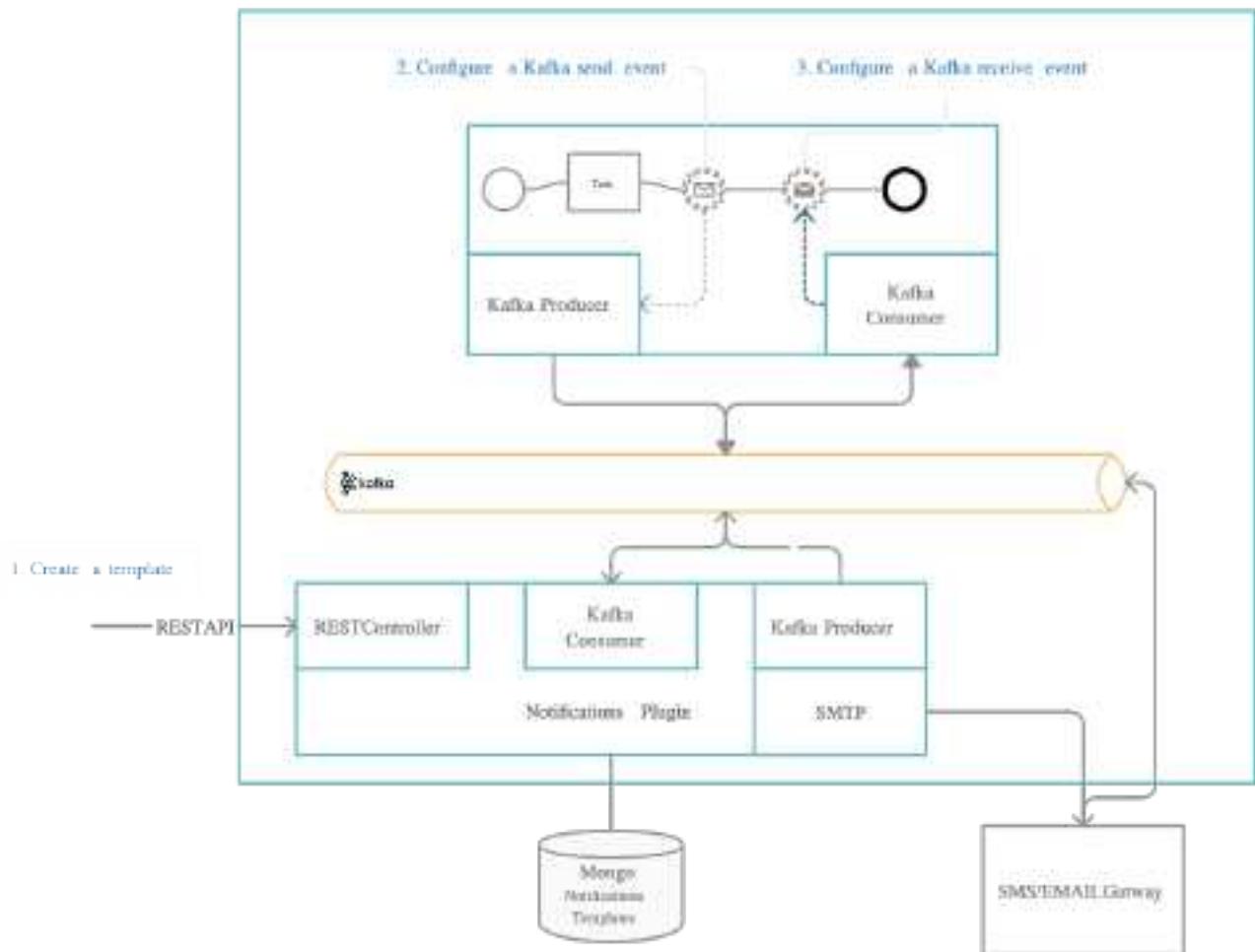
The following values are expected in the request body:

| Key      | Definition                       |           |
|----------|----------------------------------|-----------|
| language | The language that should be used | Mandatory |

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

**(!) INFO**

Check the detailed example below.

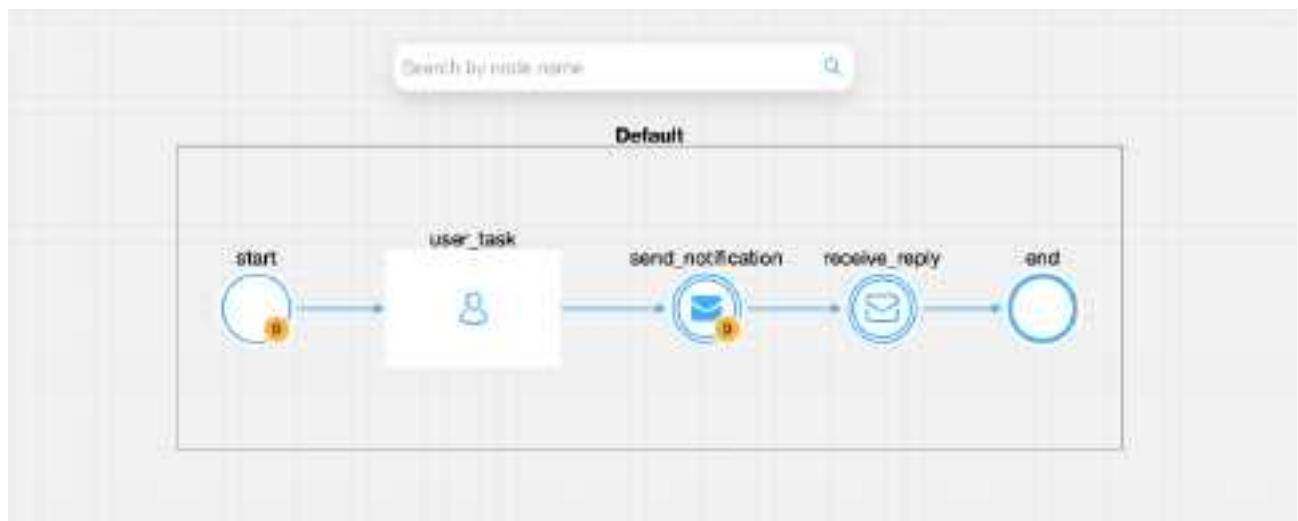


## Define needed Kafka topics

Kafka topic names can be set by using environment variables:

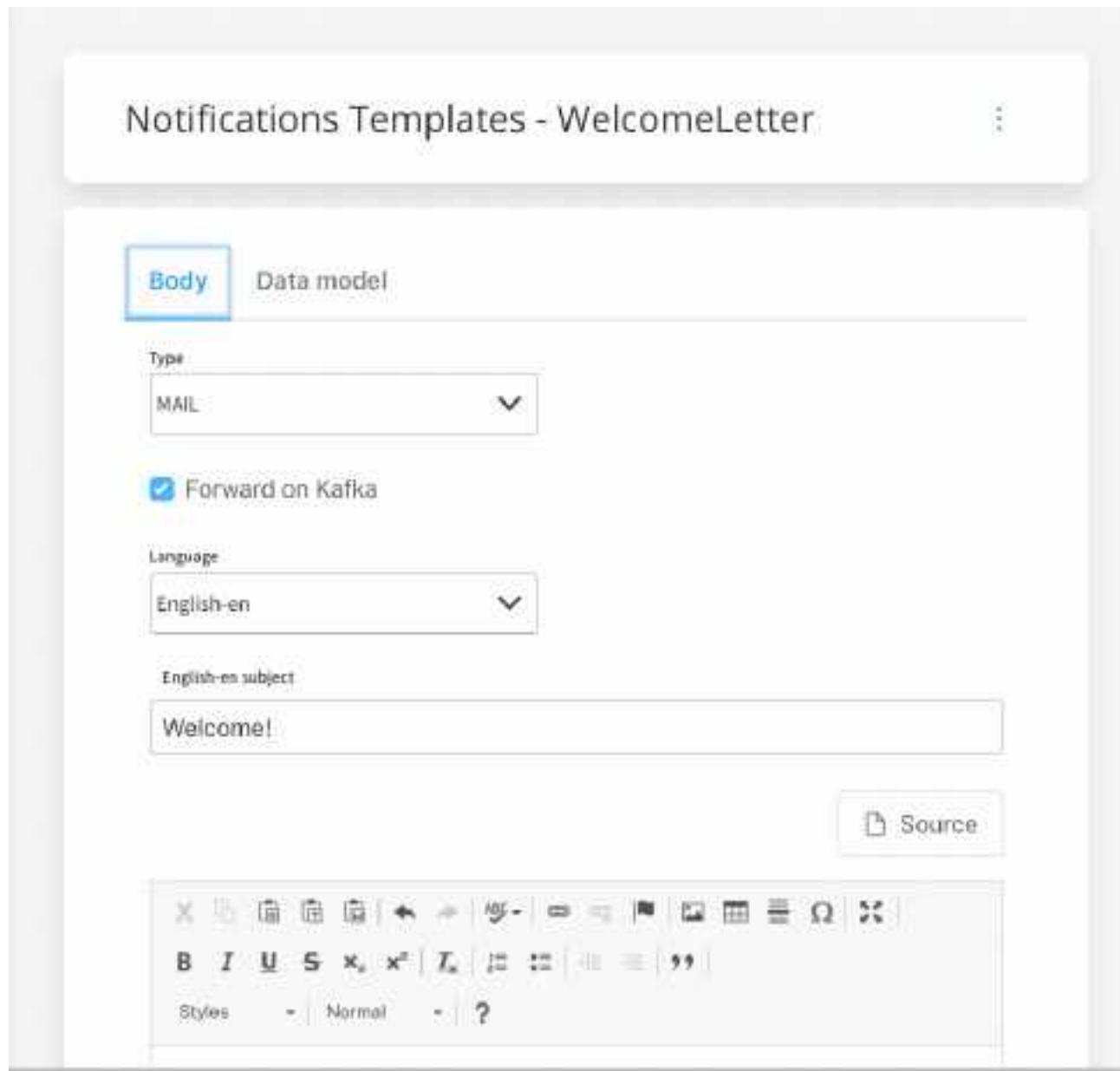
- **KAFKA\_TOPIC\_NOTIFICATION\_INTERNAL\_IN** - topic used to trigger the request to send a notification
- **KAFKA\_TOPIC\_NOTIFICATION\_INTERNAL\_OUT** - topic used for sending replies after sending the notification

## Example: send a notification from a business flow



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

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



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

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

The screenshot shows the configuration for a Kafka Send Action task. At the top, there are four checkboxes: 'Automatic' (checked), 'Manual', 'Mandatory' (checked), 'Optional', and 'Repeatable'. Below these is a switch labeled 'Allows Children?' which is turned off. A section titled 'Parameters' follows, with tabs for 'Custom' (selected) and 'From integration'. Under 'Topics', the value 'Flowx-notification-change' is listed. In the 'Message' section, a JSON message is defined:

```
1: {  
2:   "template": "welcome",  
3:   "channel": "MAIL",  
4:   "language": "en",  
5:   "receiver": "jane_doe@mail.com"  
6: }  
7:  
8:
```

Below the message, under 'Advanced configuration', there is a 'Show Headers' switch (on) and a code editor containing the header 'processInstanceID': \${processInstanceId}'. A blue 'Save' button is located at the bottom right of the configuration panel.

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

Name:

Can go back?

Variables:

Default ▼

Page ▼

Response Timeout:

Response Timeout [PT30S]

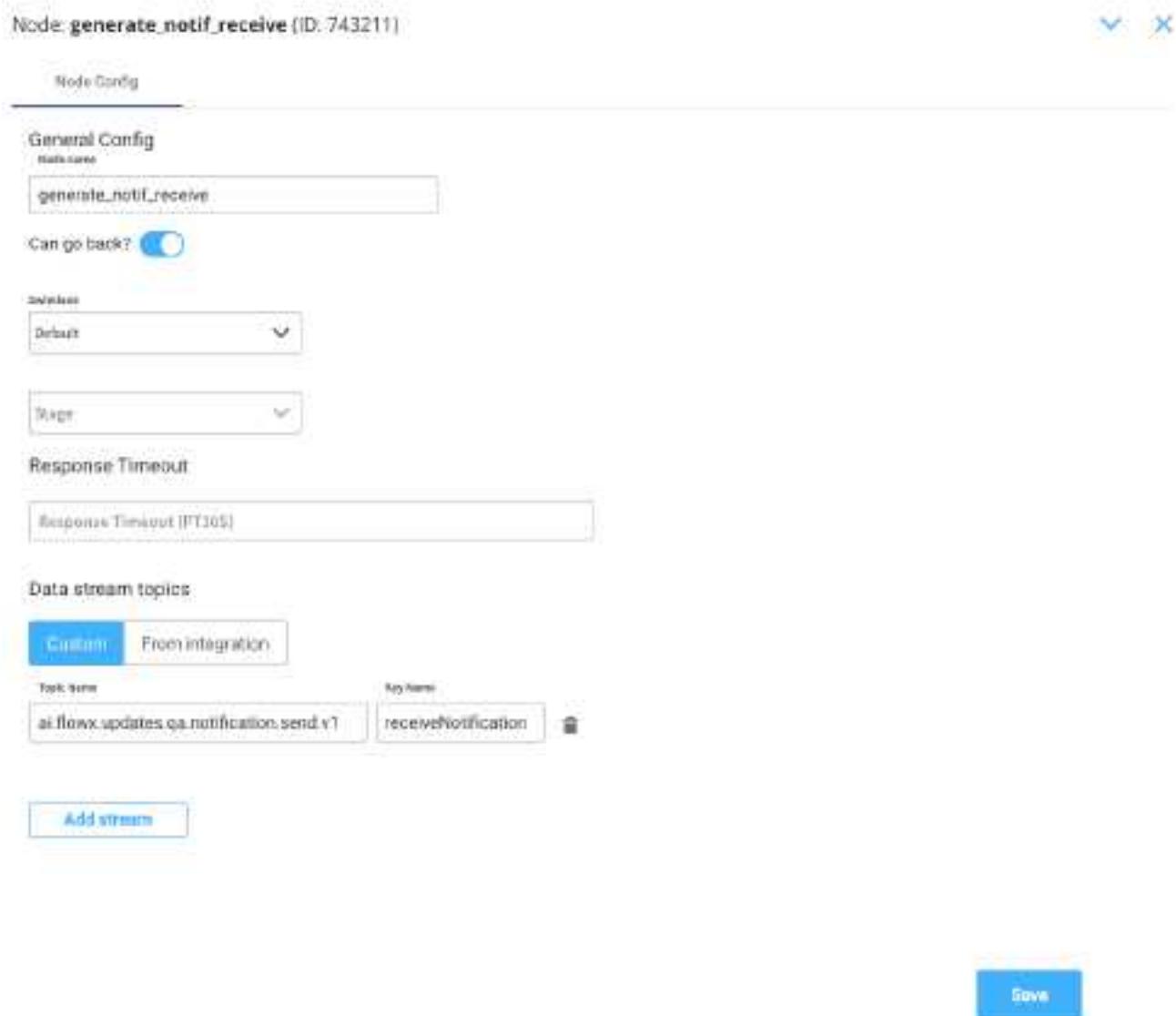
Data stream topics:

Custom From integration

Topic name:  Key name:  ✖

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.

The screenshot shows a modal window titled "Audit log details". Inside, it displays an event log entry for a "process instance, message receive" at 19 Oct 2022 at 5:24 PM. The URL is ai.flowx.updates.qa.notification.request.v1. The body of the message is a JSON object:

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

Below the modal, there are navigation links: "Process instance", "Message send", and a long ID "5fb8070-e0fc-4e10-8700-3114422611".

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

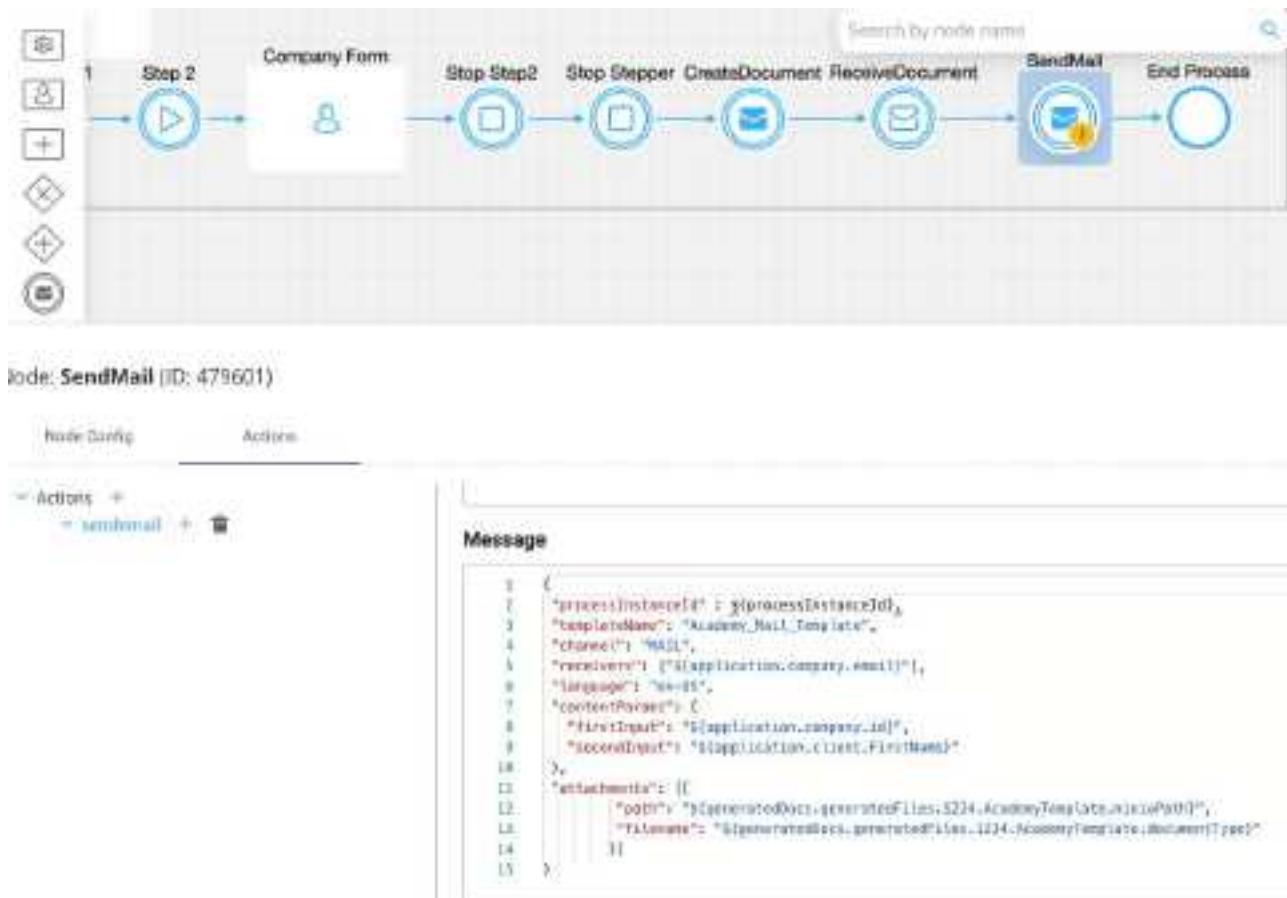
To use the notification plugin for sending emails with attachments, you must define the same topic configuration as for sending regular notifications. A notification template must be created, and the corresponding Kafka topics must be defined.

» [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.



### » 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 is null

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.

The screenshot shows the configuration for a 'Kafka Send Action' component. At the top, there are several configuration options:

- Kafka Send Action:** A dropdown menu.
- Automatic:** A checked radio button.
- Manual:** An unchecked radio button.
- Mandatory:** A checked radio button.
- Optional:** An unchecked radio button.
- Repeatable:** An unchecked radio button.
- Allows Children?**: A toggle switch that is off.

Below these settings is a section titled **Parameters**. It contains two tabs: **Custom** (selected) and **From integration**.

**Topics:** A text input field containing the value `flowx-notification-change`.

**Message:** A code editor showing a JSON message structure:

```
1: {  
2:   "template": "welcome",  
3:   "channel": "MAIL",  
4:   "language": "en",  
5:   "receiver": "jane_doe@mail.com"  
6: }  
7:  
8:
```

**Advanced configuration:** A section with a **Show Headers** toggle switch (on) and a code editor containing the header `processinstanceid`. A blue **Save** button is located at the bottom right of this section.

### (!) 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". Below it, the "Url" is listed as "ai.flowx.updates.qa.notification.request.v1". Under the "Body:" heading, a JSON object is displayed:

```
{ "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 "5f629070-8d1c-4e1e-870-3174422651".

Response example at `KAFKA_TOPIC_NOTIFICATION_INTERNAL_OUT`:

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

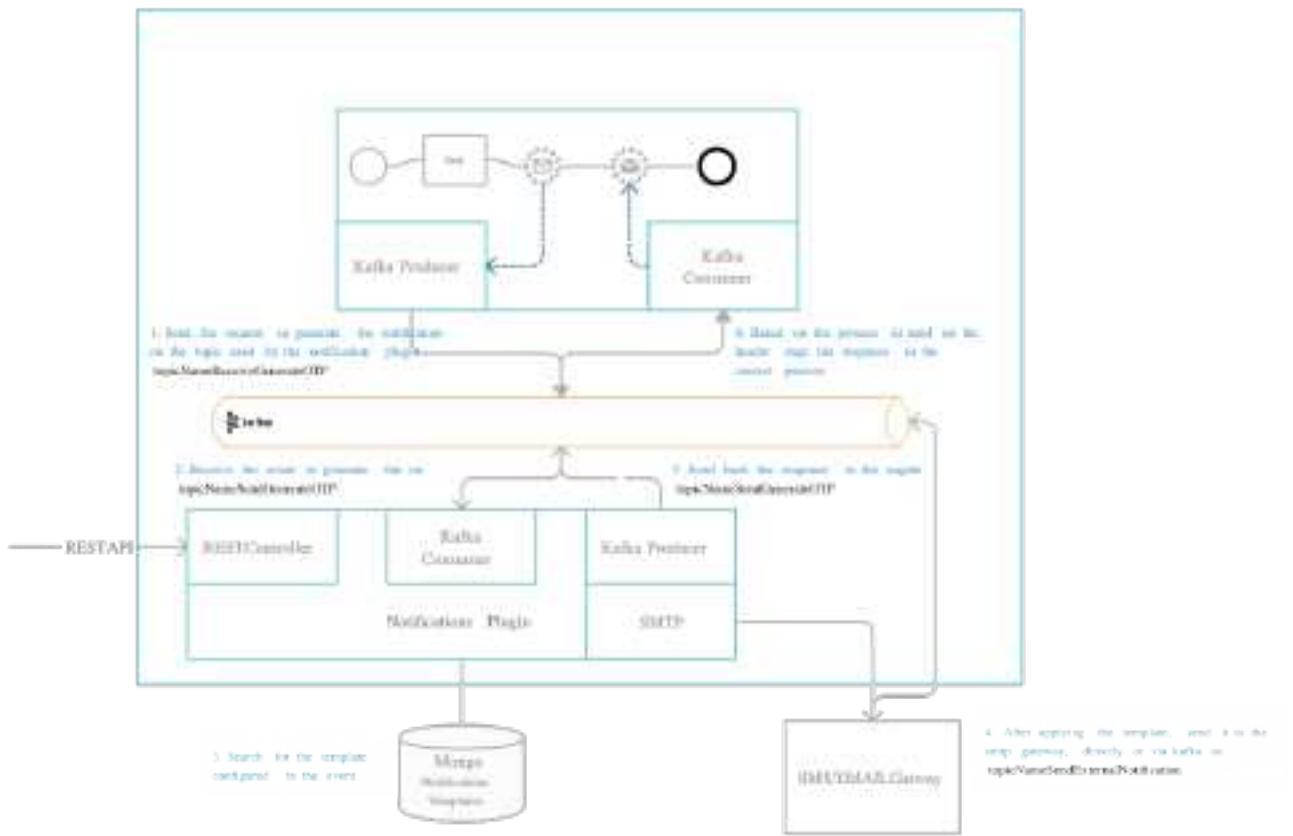
```
    "john.doe@mail.com"  
],  
"channel": "MAIL",  
"language": "en"  
}
```

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

From integration

Topics

so, files, go, notification, etc., in, generate, v3

## Messaggi

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

## 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 is a table with one row. The first column contains the number "1" and the second column contains a JSON object representing an audit log entry. The JSON is as follows:

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

The modal has a close button in the top right corner. The background of the application shows a sidebar with "ACTIVE" and "Audit" tabs, and a main area with "Flowx.ai", "Process instance", "Execute action", and "actions25".

## 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

Repeatable

Automate Children?

Allow BACK on this action?

### Parameters

Custom  From Integration

#### Topics

```
as_email_in_qa.notification.otp.generate,v1
```

#### Message

```

1   {
2     "templateName": "otpSMS",
3     "language": "en-US",
4     "channel": "SMS",
5     "clientID": "application-client-tester@caridata-personalClient@caridata",
6     "recipient": "application-client-contactDataMobilePhone-channelNumber"
7   }

```

#### Advanced configuration

Show Headers

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.



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:

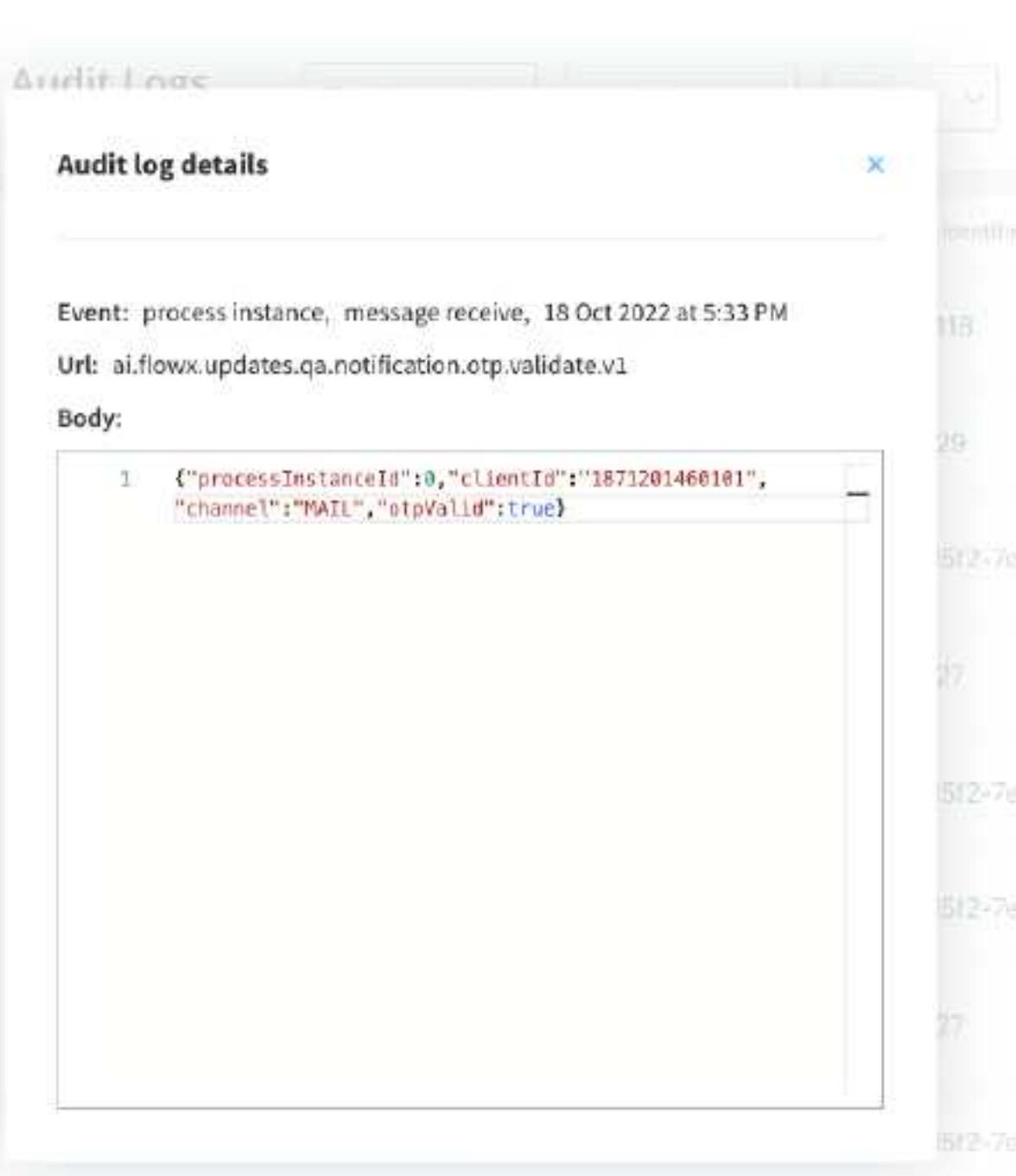
```
{  
    "processInstanceId": 12345,  
    "clientId": "1871201460101",  
    "channel": "MAIL",  
    "otp": "1111"  
}
```

## 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:



## 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: ai.flowx.updates.qa.notification.otp.valid  
Key name: 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          |                          |                       |             |
|---------------------------|--------------------------|-----------------------|-------------|
| Process                   |                          |                       |             |
| Swimlanes Process         |                          |                       |             |
| Swimlane                  | 周恩来                      | Edited at             | Assigned by |
| BACKOFFICE                | backoffice@email.com     | 09 Aug 2022, 10:30 AM | John Doe    |
| OFFICEADMIN               | admin@email.com          | 09 Aug 2022, 7:00 PM  | John Doe    |
| SUPERVISOR                | same as execution rights | 09 Aug 2022, 9:18 AM  | John Doe    |
| FRONTOFFICE               | frontoffice@email.com    | 09 Aug 2022, 9:18 AM  | John Doe    |
| Parallel Gateways Process |                          |                       |             |

### !(INFO)

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

## Accessing allocation rules

To access the allocation rules, follow the next steps:

1. Open [The fallback content to display on prerendering](#).
2. From the side menu, under **Task Management**, select the **Allocation rules** entry.

| Swimlane    | Allocation | Date/Time             | Allocated by |
|-------------|------------|-----------------------|--------------|
| OFFICEADMIN | John Doe   | 03-Aug-2022, 11:16 AM | John Doe     |
| SUPERVISOR  | John Doe   | 03-Aug-2022, 11:16 AM | John Doe     |
| BACKOFFICE  | John Doe   | 03-Aug-2022, 11:16 AM | John Doe     |
| FRONTOFFICE | John Doe   | 03-Aug-2022, 11:16 AM | John Doe     |

## Adding process and allocation rules

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

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

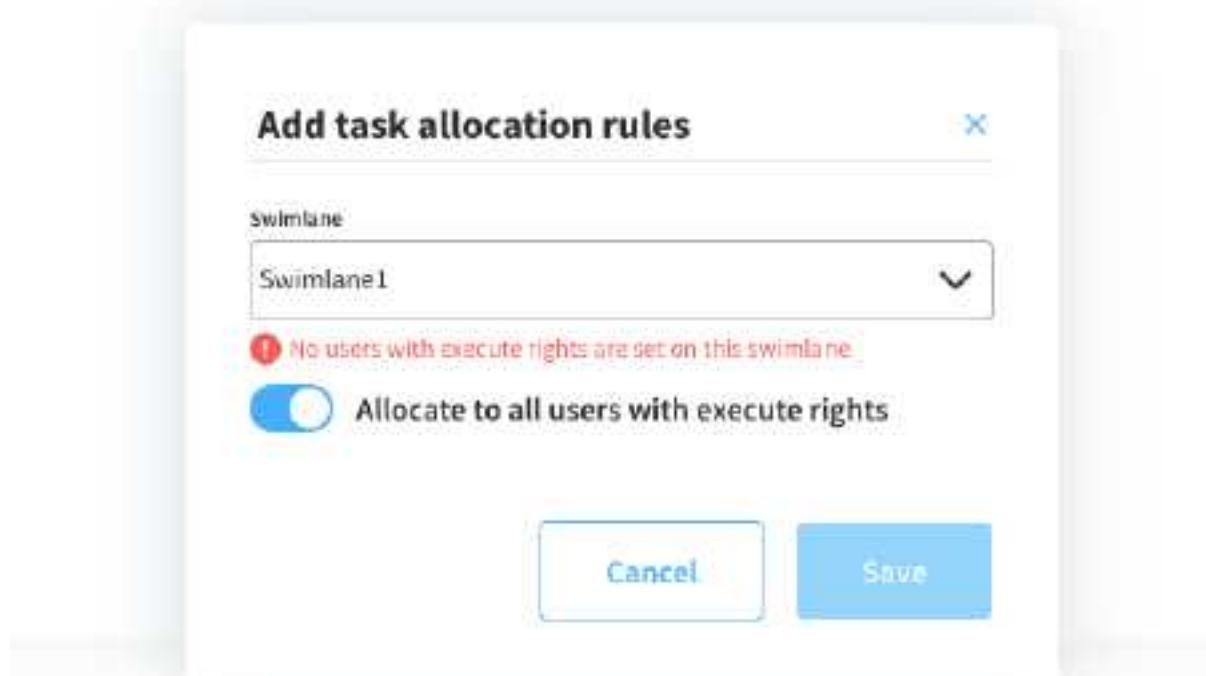
The screenshot shows the 'Allocation rules' page with the 'Add process' button highlighted in green, indicating it is the next step in the process configuration.

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

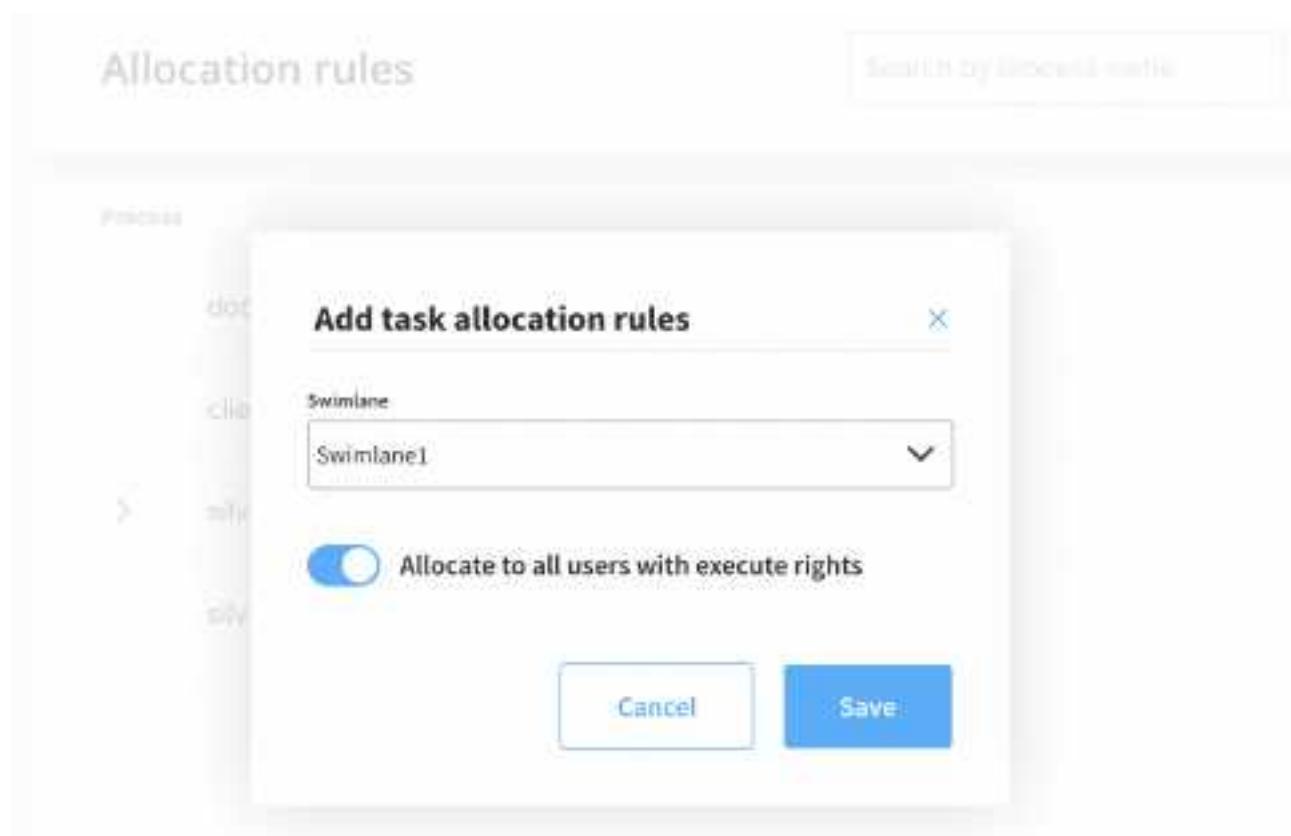
| Process     |                       |           |                          |
|-------------|-----------------------|-----------|--------------------------|
| Swimlane    | Allocation            | Edited at | Add swimlane allocations |
| BACKOFFICE  | 09 Aug 2022, 10:30 AM | John Doe  |            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

### ⚠ CAUTION

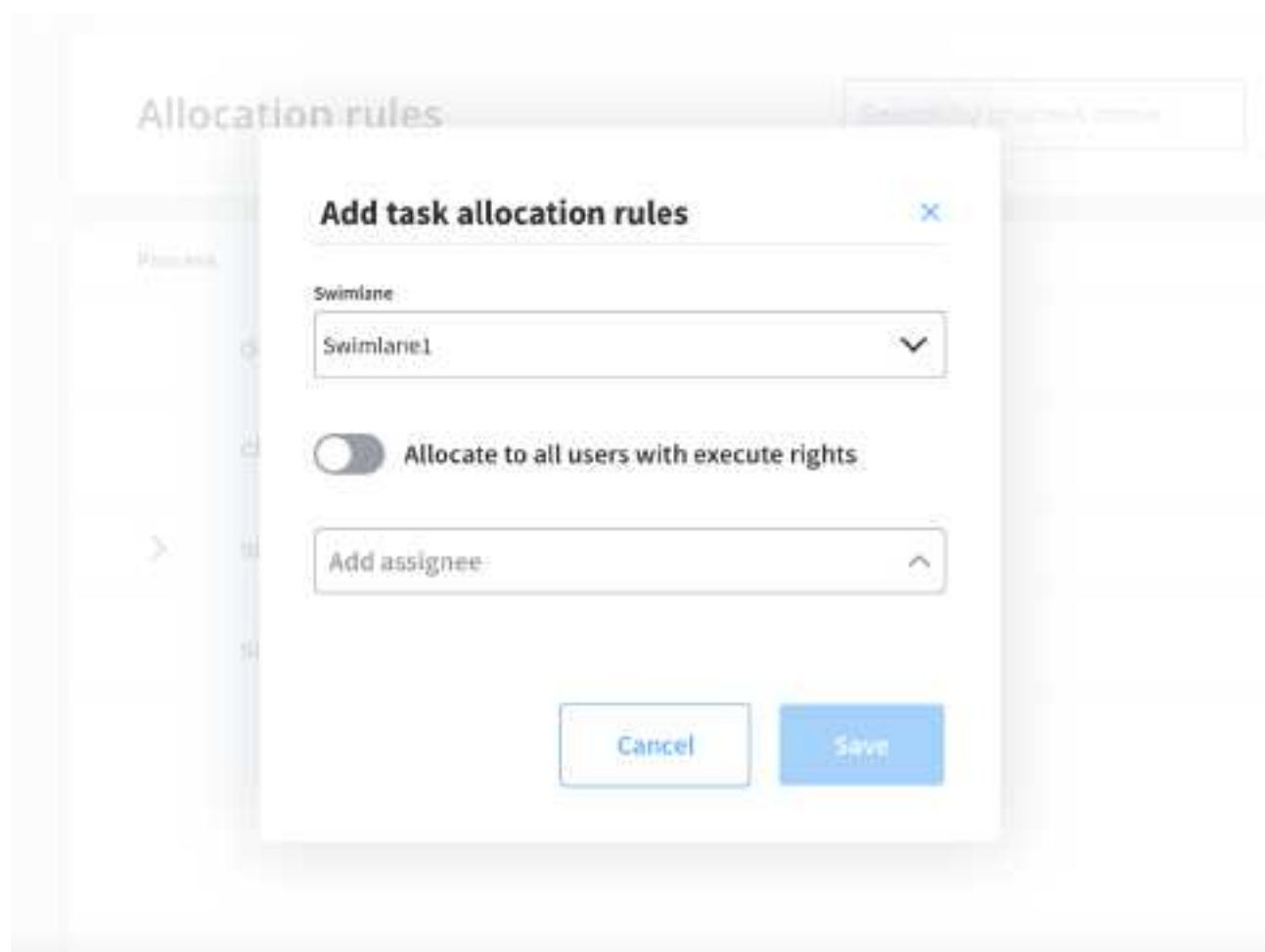
**NOTE!** If there are no users with execute rights in the swimlane you want to add (`hasExecute: false`), the following error message will be displayed:



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



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



6. Click **Save**.

#### !(INFO)

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

## Editing allocation rules

To edit allocation rules, follow the next steps:

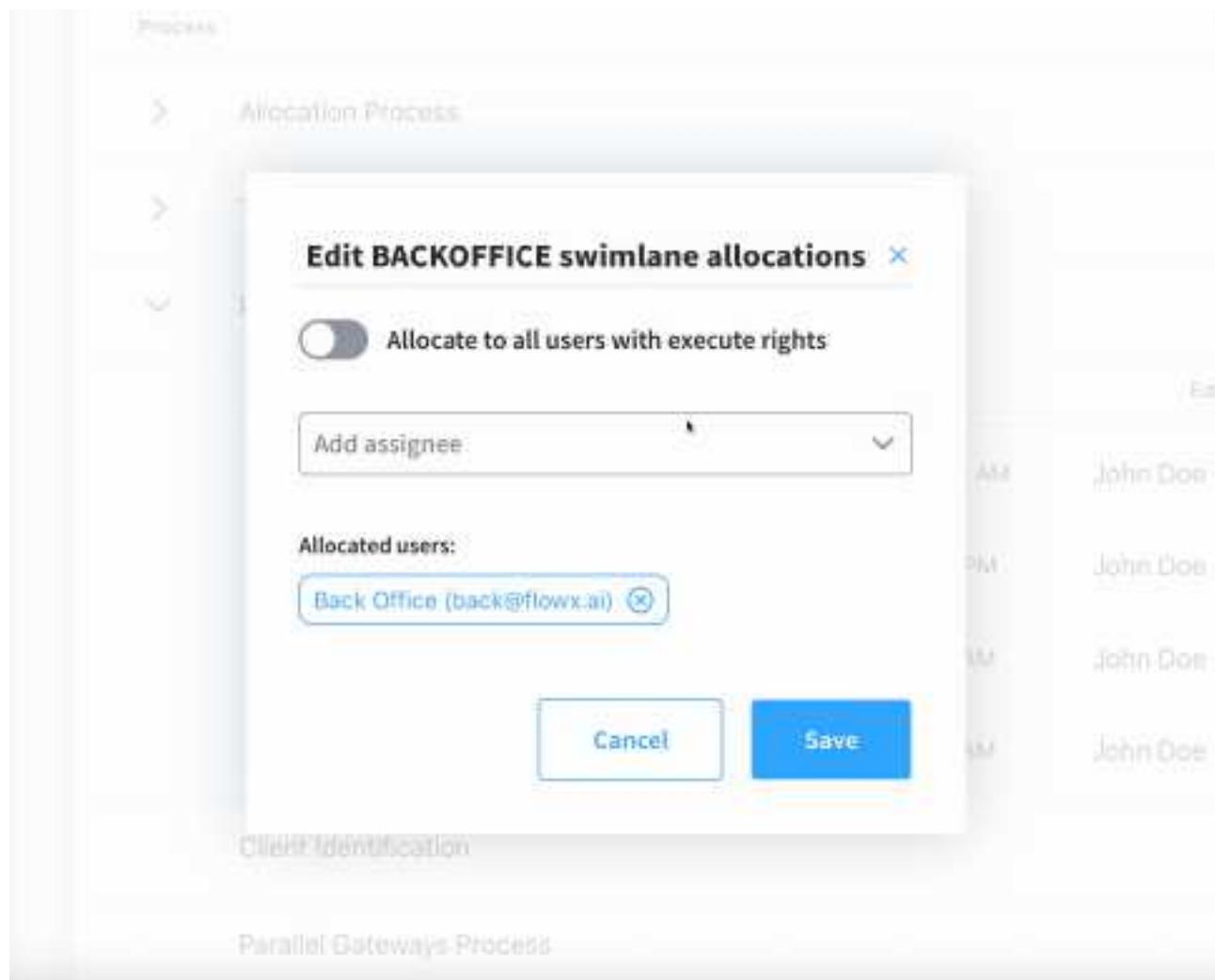
1. Click **Edit** button.

The screenshot shows a user interface for managing allocation rules. At the top, there is a search bar labeled "Search by process name" and a button labeled "Add process". Below this, a section titled "Allocation rules" lists three processes: "Allocation Process", "Test Process", and "Parallel Swimlanes Process".

| Process                    | Allocation  | Edited at             | Edited by |
|----------------------------|-------------|-----------------------|-----------|
| Allocation Process         | BACKOFFICE  | 09 Aug 2022, 10:30 AM | John Doe  |
| Test Process               | OFFICEADMIN | 03 Aug 2022, 7:00 PM  | John Doe  |
| Parallel Swimlanes Process | SUPERVISOR  | 03 Aug 2022, 9:16 AM  | John Doe  |
|                            | FRONTOFFICE | 03 Aug 2022, 9:16 AM  | John Doe  |

Each row in the table has an "Edit" button icon (a pencil and a speech bubble) next to it. The "Allocation" column lists swimlane names: BACKOFFICE, OFFICEADMIN, SUPERVISOR, and FRONTOFFICE. The "Edited at" column shows the timestamp of the last edit, and the "Edited by" column shows the name of the user who made the edit.

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

| Process                   | System      | Allocation               | Edited at             | Edited by |  |
|---------------------------|-------------|--------------------------|-----------------------|-----------|--|
| 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, 8:18 AM  | John Doe  |  |
|                           | FRONTOFFICE | frontoffice@email.com    | 03 Aug 2022, 9:16 AM  | John Doe  |  |
| Parallel Gateways Process |             |                          |                       |           |  |

## 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:

| Process | Swinlanes Process | Swinlane    | Allocation | Period at             | Allocated by | Actions |
|---------|-------------------|-------------|------------|-----------------------|--------------|---------|
|         |                   | BACKOFFICE  | Allocated  | 08 Aug 2022, 10:30 AM | John Doe     |         |
|         |                   | OFFICEADMIN | Allocated  | 03 Aug 2022, 7:00 PM  | John Doe     |         |
|         |                   | SUPERVISOR  | Allocated  | 03 Aug 2022, 3:15 AM  | John Doe     |         |
|         |                   | FRONTOFFICE | Allocated  | 03 Aug 2022, 3:15 AM  | John Doe     |         |

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, there is a sidebar with navigation links: PROCESSES (Definitions, Active process), CONTENT MANAGEMENT (Enumerations, Submission stage, Content model, Languages, Source system), and PLUGINS (Task Manager, Plugins, Scripts). The main area is titled "Hooks" and contains a table with three rows. The columns are "Hook Name", "Type", and "Active". The rows are: Hook1 (Process, Active), Hook2 (Swimlane, Active), and Hook3 (Swimlane, Active). There is also a "New hook" button at the top right of the table.

| Hook Name | Type     | Active |
|-----------|----------|--------|
| 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.

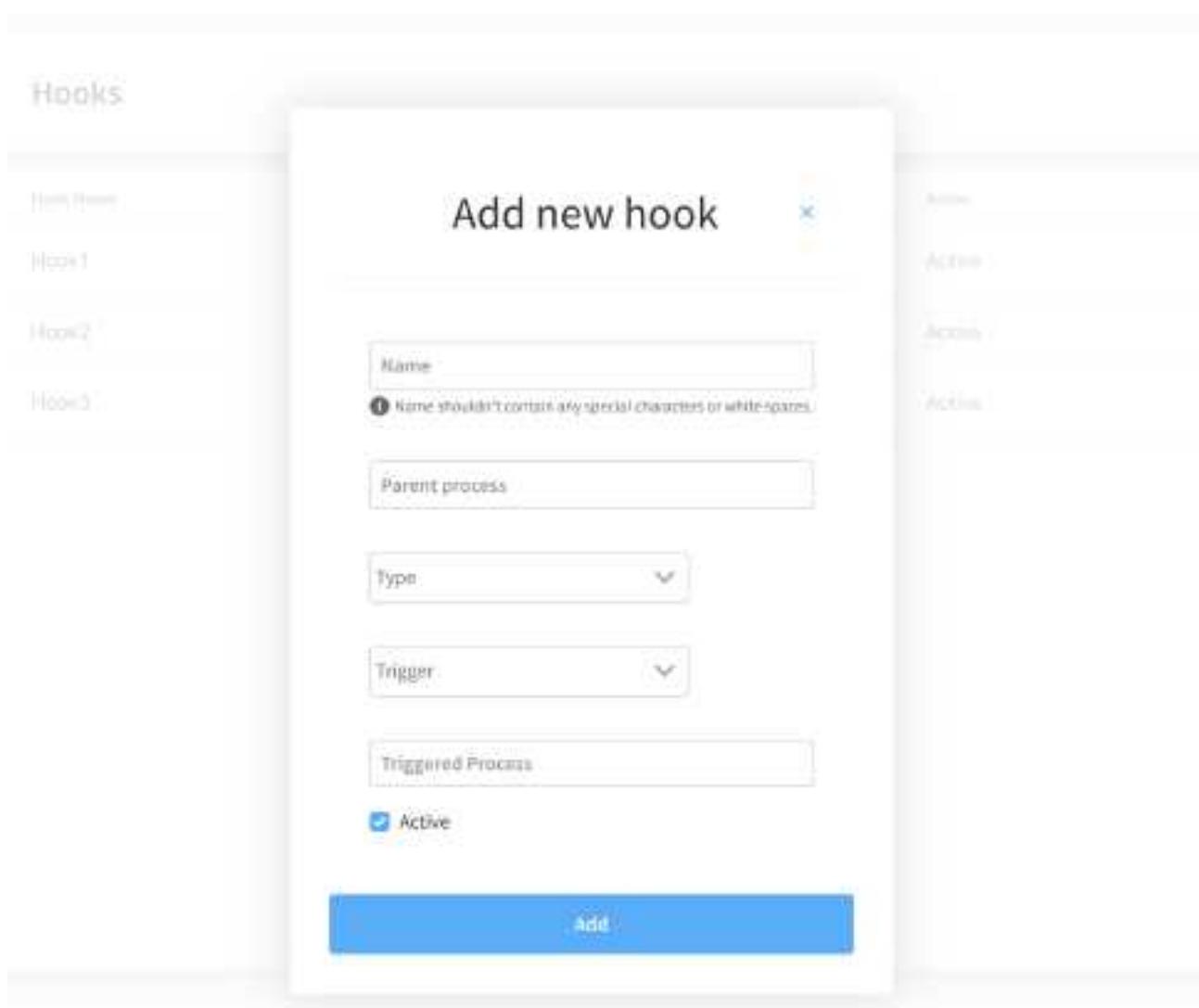
#### Process      Swimlane      Stage

- 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.



## 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).

## Add new hook

Name:  (\*) Name shouldn't contain any special characters or white spaces

Parent process:

Type:  Swimlane

Swimlane:

Trigger:  Swimlane Entered

Triggered Process:

Add SLA  
Time threshold:  (\*) 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.

| Out of office              |             |             |                      |           |
|----------------------------|-------------|-------------|----------------------|-----------|
| 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.twi...) | 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, 9:26 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's sidebar navigation on the left and a main content area on the right. The sidebar includes sections for Plugins, Task Manager (with sub-options: Activities, Workflows, Stages, Allocation rules, and Out of office), Notification templates, Document templates, General Settings (with sub-options: Generic parameters, Integration management, Licensing, Access management, and a 'Create' button), and a user profile section for John Doe. The main content area is titled 'Out of office' and contains a table with three rows of data. The columns are labeled: User, Start date, End date, Edited at, and Edited by. The data rows are: Jane Smith (Jane...), 03 Aug 2022, 07 Aug 2022, 03 Aug 2022, 3:24 PM, John Doe; Ewa Twinkins (Ew...), 03 Aug 2022, 31 Dec 2040, 03 Aug 2022, 3:16 PM, John Doe; and Gena Eva (gena...), 08 Aug 2022, 27 Aug 2022, 03 Aug 2022, 9:25 AM, John Doe. There is also a 'Search by user' input field and a 'Add out-of-office' button.

| 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  |
| Ewa Twinkins (Ew...) | 03 Aug 2022 | 31 Dec 2040 | 03 Aug 2022, 3:16 PM | John Doe  |
| Gena Eva (gena...)   | 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 a user interface for managing out-of-office records. At the top, there's a header bar with the title 'Out of office' and a search bar labeled 'Search by user'. Below this, a modal window titled 'Add out-of-office' is open. The modal 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). To the left of the modal, a sidebar lists users: Jane Sm, Bess Tw, and Gene Ew, each with a small profile picture and a timestamp below their name: 'Aug 2022, 3:24 PM', 'Aug 2022, 3:16 PM', and 'Aug 2022, 9:25 AM' respectively.

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**.

The screenshot shows a modal dialog titled "Edit out-of-office for John Doe". Inside the dialog, there are two date input fields: "start date" containing "11.08.2022" and "end date" containing "28.08.2022". Below the date fields are two buttons: "Cancel" and "Save". The "Save" button is highlighted with a yellow border. In the background, there is a list of users on the left and a list of events on the right.

| User        | Event                       |
|-------------|-----------------------------|
| Jane Smith  | Meeting 2022-08-11 10:00 AM |
| Bob Johnson | Meeting 2022-08-11 11:00 AM |
| Grace Evans | Meeting 2022-08-11 12:00 PM |

## 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."*

| Out of office               |             |             |                      |           | Search by user | Add out-of-office |
|-----------------------------|-------------|-------------|----------------------|-----------|----------------|-------------------|
| User                        | Start Date  | End Date    | Edited at            | Edited by |                |                   |
| Jane Smith (jane.smith)     | 03 Aug 2022 | 07 Aug 2022 | 03 Aug 2022, 3:24 PM | John Doe  |                |                   |
| Bess Twishes (bess.twishes) | 03 Aug 2022 | 31 Dec 2046 | 03 Aug 2022, 3:18 PM | John Doe  |                |                   |
| Gene Eva (gene.eva)         | 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   |              |             |                      |  |
|---|--------------|-------------|----------------------|--|
| <input type="text" value="Search by user"/> <a href="#">Add out-of-office</a> |              |             |                      |  |
| 1 User  | 2 Start Date | 3 End Date  | 4 Edited at          | 5 Edited by  |
| Jane Smith (jane.s...)  | 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?

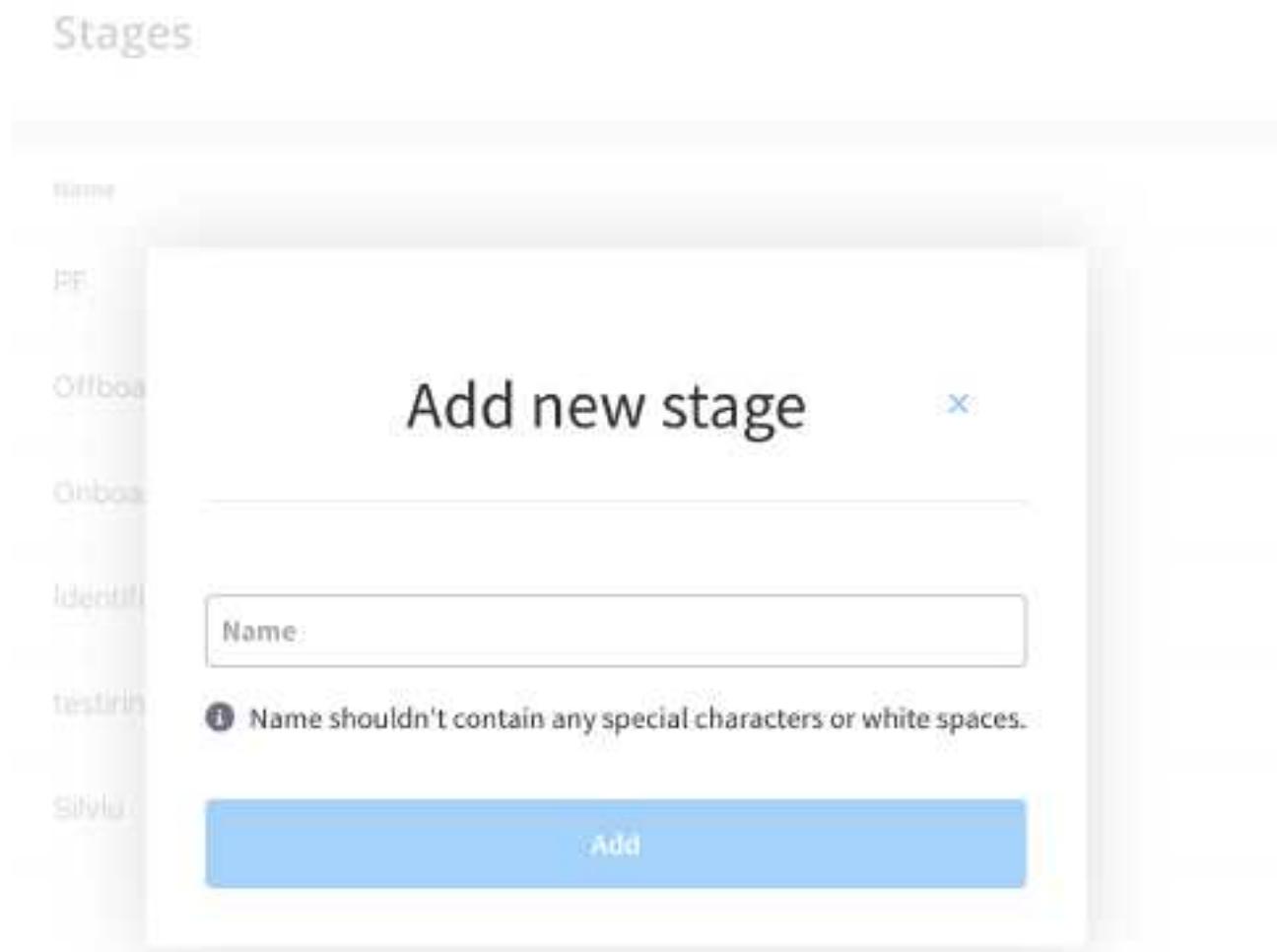
## PLATFORM DEEP DIVE / Plugins / Custom Plugins / Task management / Using stages

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.



## 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)

Node Config    Actions

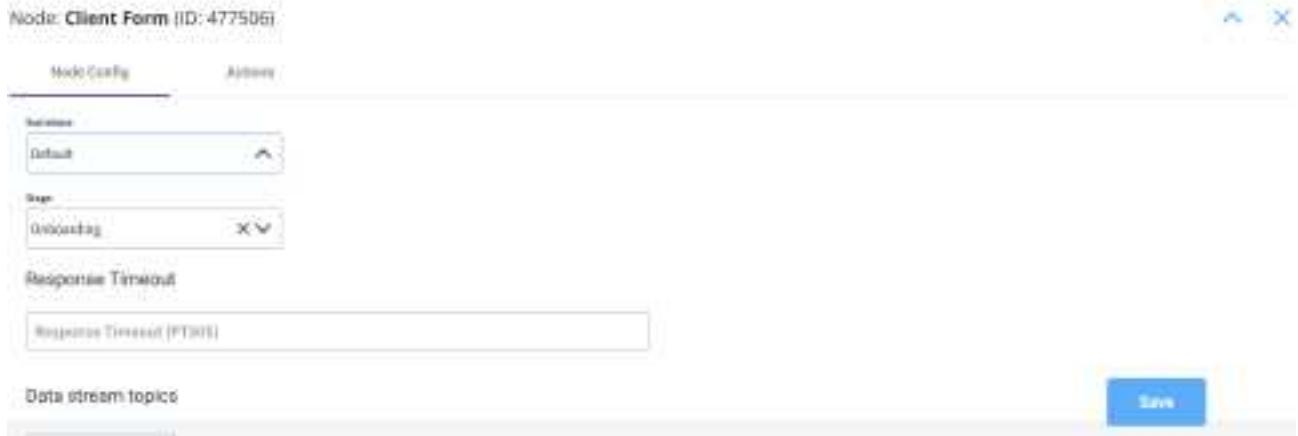
Backend: Default

Stage: Unpublished

Response Timeout: Response Timeout (PT0S)

Data stream topics:

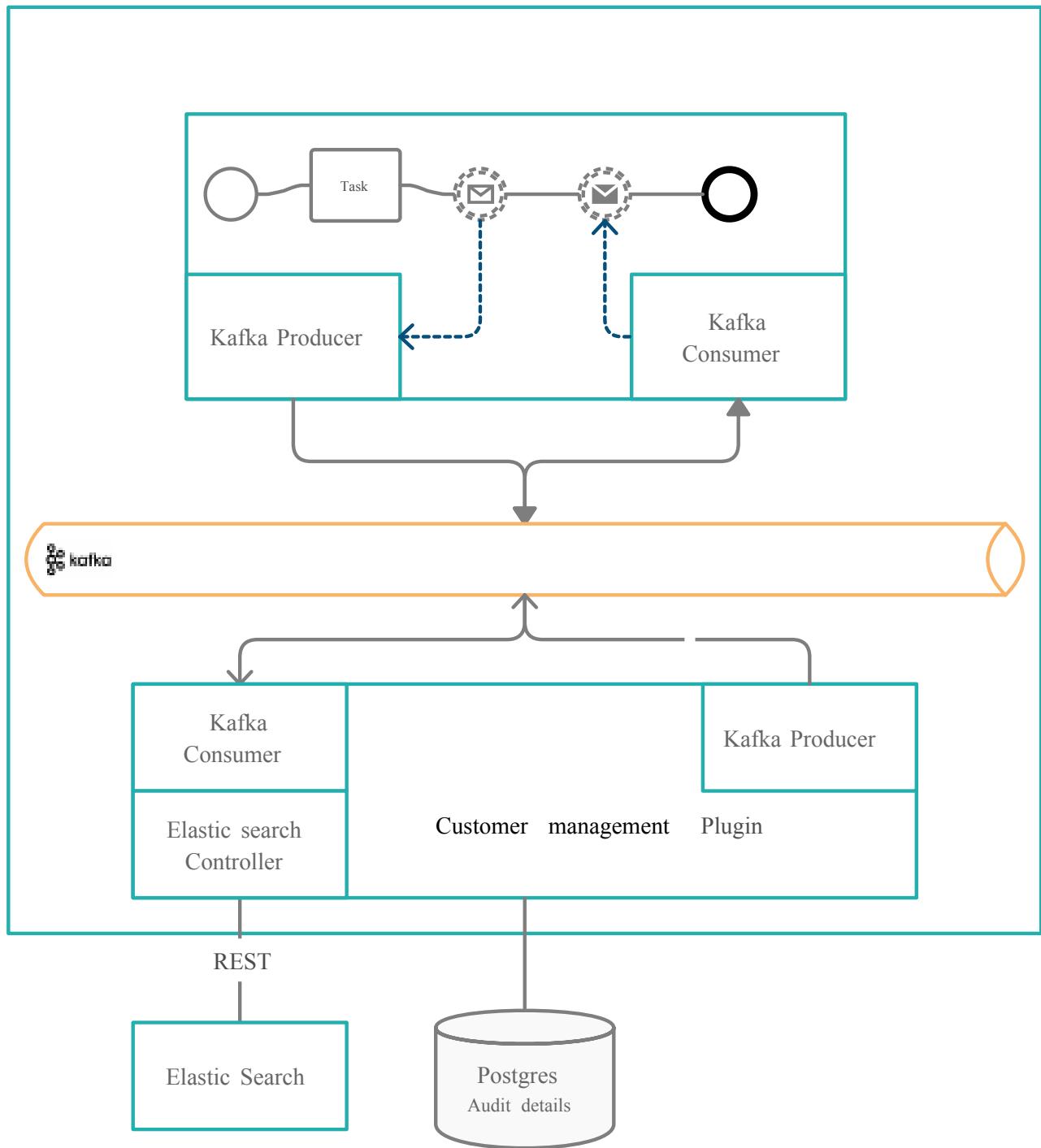
**Save**



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": "${clientIdentification.id}",
3    "cii": "${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: ai.flowx.updates.qa.customer.management.response.

Key Name: receiveReply

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 for Processes (Definitions, Active process, Process instances, Failed process start), Content Management (Enumerations, Substitution tags, Content models, Languages, Source systems), Plugins (Task Manager, AI tasks, HOOKS, Stages, Allocation rules, Out of office, Notification templates), and a user profile for John Doe.

The main area is titled "Process Definitions" and contains two tables:

- Drafts / In progress**:

| Name | Version | Edited at             | Edited by      |
|------|---------|-----------------------|----------------|
| s... | 1       | 30 Sep 2022, 5:41 PM  | Silviu Grigore |
| d... | 1       | 30 Sep 2022, 2:32 PM  | andrei anali   |
| t... | 1       | 30 Sep 2022, 11:15 AM | QA FlowX       |
| T... | 1       | 30 Sep 2022, 10:29 AM | QA FlowX       |
- Published**:

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

2. Create a process and add a **Kafka Send Action** to a **Message event send** node. Here you specify the **kafka topic** (address) where the template will be generated.

The screenshot shows the 'Action Edit' screen for an action named 'send\_main\_applicant\_covenant'. The action has an ID of 584701 and is currently set to 'Automatic' and 'Mandatory'. It is not repeatable and does not have any children. The 'Parameters' tab is selected, showing two tabs: 'Custom' (selected) and 'From Integration'. Under the 'Topics' tab, the topic 'al.21rock.00-wamtime-document-.html-generate.v3' 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 4
2 "documentList": [
3   {
4     "customId": "1234",
5     "templateName": "Agreement",
6     "language": "ru",
7     "data": {
8       "offerName": "The greatest offer - deluxe edition",
9       "companyName": "Test Company SRL",
10      "cui": "R01234567",
11      "firstName": "Bess",
12      "lastName": "Tuishes",
13      "offerExpiryDate": "31.12.2899"
14    },
15    "includeBarcode": true
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.

**!** **INFO**

- 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 / Custom 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

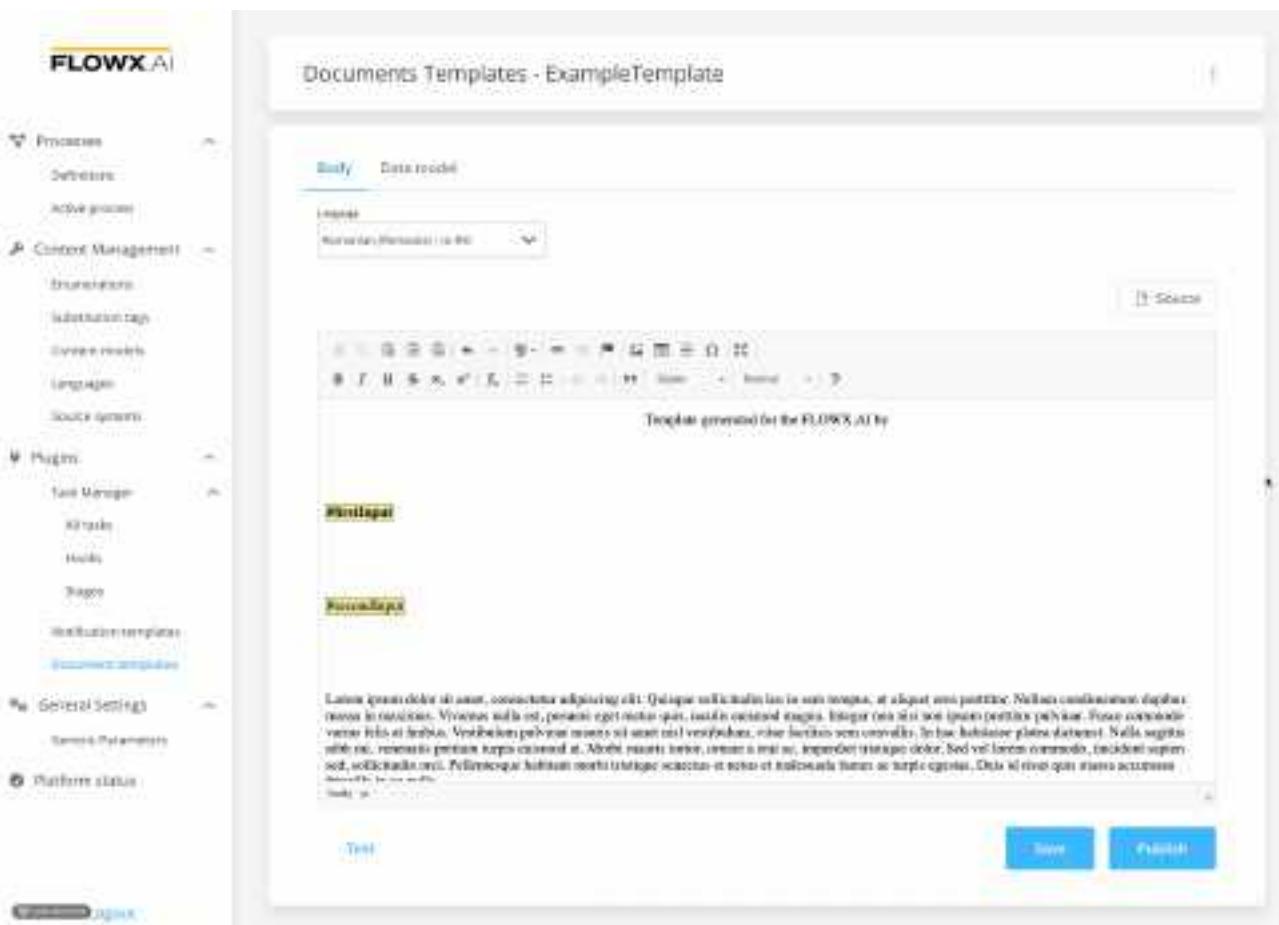
The Body is the main part of the editor where you can edit your template.

#### !( 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.



## 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

At the bottom of the sidebar, there is a "Logout" link.

The main content area is titled "Documents Templates - ExampleTemplate". It has tabs for "Body" and "Data model". The "Body" tab is selected, showing the following content:

- Language: Romanian (Romanian - ro-RO)
- Source button
- Rich text editor toolbar
- Text area containing:

Template generated for the FLOWX.AI Academy by

#firstInput

#secondInput

Leem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sollicitudin leo in sem tempus. ut aliquet eros poenitioe. Nullam condimentum dapibus massa in maximus. Vivamus nulla est, posuere eget metus quis, lacinia euismod magna. Integer non nisi non ipsum poenitior 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
- Text, 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

The screenshot shows the FLOWX.AI platform interface for managing notifications. On the left, a sidebar navigation includes 'Processes' (Definitions, Active process), 'Content Management' (Experiments, Test Automation tags, Content review, Languages, Service gateway), 'Plugins' (Task Manager, Metrics, Hooks, Steps), 'Notification templates' (Notification templates, Document templates), 'General settings' (General Parameters), and 'Platform status'. The main area is titled 'Notifications Template - ExampleTemplate'. It has tabs for 'Body' (selected) and 'Data model'. Under 'Body', the 'Type' is set to 'Mail'. There is a checkbox for 'Forward on Kafka' which is unchecked. The 'Language' dropdown is set to 'Russian (UTF-8)'. Below these are sections for 'Message parameters' and 'Contract (Definition)'. The 'Body' section contains a WYSIWYG editor toolbar and a preview window showing the text 'FLOWX.AI'. Below the preview is a 'Text' input field containing the placeholder 'Salut #firstinput #secondinput, я бывал на авен аллея!' and a note 'Обработка скрипта в браузере не рекомендована.' At the bottom right are 'Save' and 'Publish' buttons.

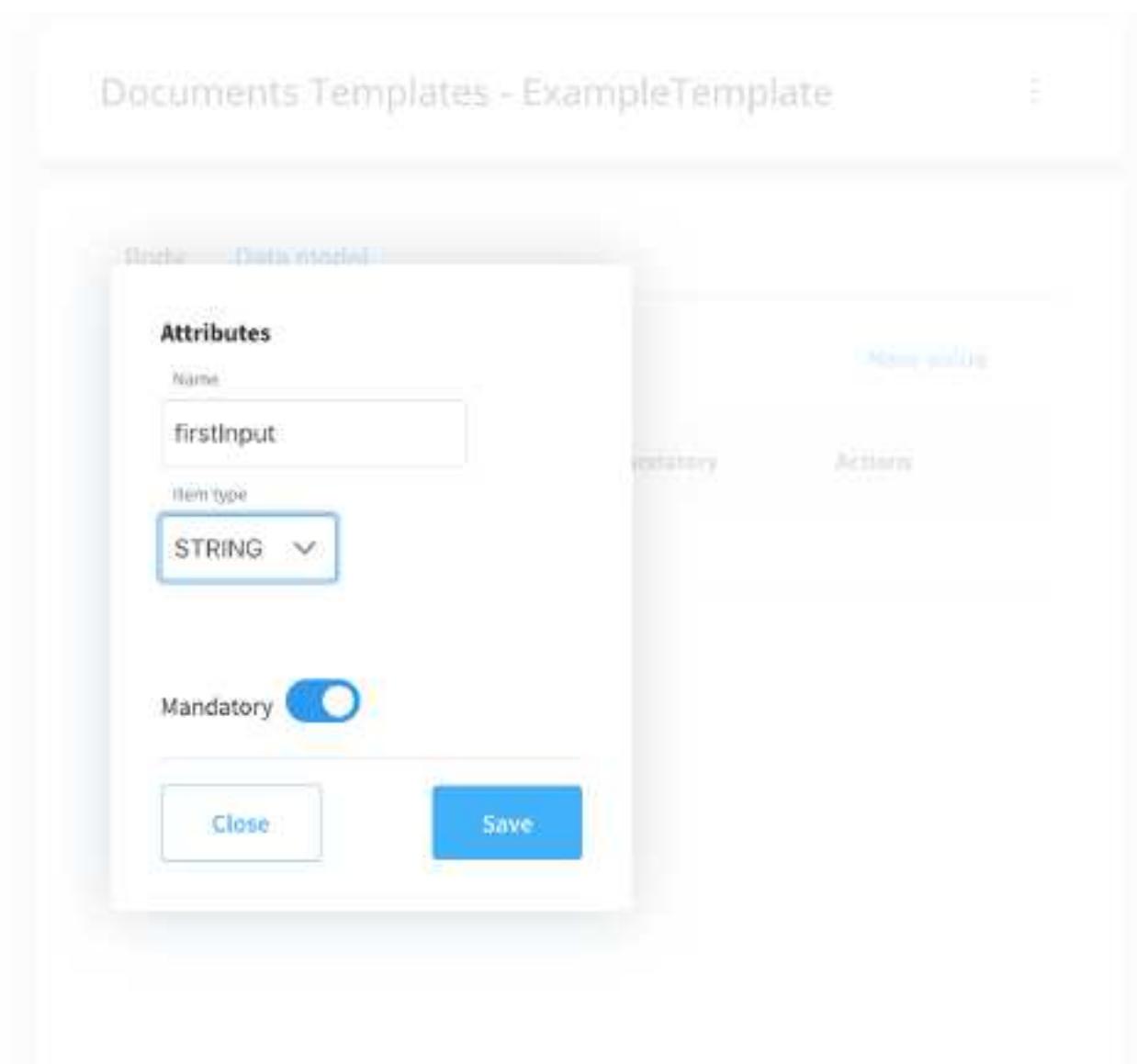
» Managing notifications templates

## Data Model

### Data Model

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

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



### (!) 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 / Plugins / Plugins setup guides / Customer management plugin setup

## Infrastructure Prerequisites:

The Customer management plugin is available as a docker image so we need to configure:

### Elastic Search

In order to install elasticsearch instance Elastic Cloud on Kubernetes (ECK) can be used.

Use ECK quickstart to deploy CRDs and create elasticsearch instances:

Elasticsearch instance:

```
apiVersion: elasticsearch.k8s.elastic.co/v1
kind: Elasticsearch
metadata:
  name: elasticsearch-flowx
  namespace: elastic-system
spec:
  version: 7.9.3
  updateStrategy:
    changeBudget:
      maxSurge: 3
```

```
    maxUnavailable: 1
nodeSets:
# 3 dedicated master nodes
- name: master
  count: 3
  config:
    node.master: true
    node.data: false
    node.ingest: false
    node.remote_cluster_client: false
    # this allows ES to run on nodes even if their
    vm.max_map_count has not been increased, at a performance
    cost
    # node.store.allow_mmap: false
podTemplate:
  spec:
    initContainers:
    - name: sysctl
      securityContext:
        privileged: true
      command: ['sh', '-c', 'sysctl -w
vm.max_map_count=262144']
    - name: install-plugins
      command:
      - sh
      - -c
      - |
        bin/elasticsearch-plugin install --batch
repository-gcs
  containers:
  - name: elasticsearch
    resources:
      limits:
        memory: 6Gi
        cpu: 2
```

```
        requests:
          memory: 2Gi
          cpu: 1
        env:
        - name: ES_JAVA_OPTS
          value: "-Xms2g -Xmx2g"
        - name: READINESS_PROBE_TIMEOUT
          value: "10"
        readinessProbe:
          exec:
            command:
            - bash
            - -c
            - /mnt/elasticsearch/internal/scripts/readiness-
probe-script.sh
            failureThreshold: 3
            initialDelaySeconds: 10
            periodSeconds: 12
            successThreshold: 1
            timeoutSeconds: 12
        affinity:
          podAntiAffinity:
            preferredDuringSchedulingIgnoredDuringExecution:
            - weight: 100
              podAffinityTerm:
                labelSelector:
                  matchLabels:
                    elasticsearch.k8s.elastic.co/cluster-
name: elasticsearch-flowx
                topologyKey: kubernetes.io/hostname
                # request 2Gi of persistent data storage for pods in
                # this topology element
              volumeClaimTemplates:
              - metadata:
```

```
        name: elasticsearch-data
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
  storageClassName: standard
# 3 ingest-data nodes
- name: ingest-data
  count: 3
  config:
    node.master: false
    node.data: true
    node.ingest: true
    # this allows ES to run on nodes even if their
    vm.max_map_count has not been increased, at a performance
    cost
    # node.store.allow mmap: false
podTemplate:
  spec:
    initContainers:
      - name: sysctl
        securityContext:
          privileged: true
          command: ['sh', '-c', 'sysctl -w
vm.max_map_count=262144']
    containers:
      - name: elasticsearch
        resources:
          limits:
            memory: 8Gi
            cpu: 2
          requests:
            memory: 4Gi
```

```
        cpu: 1
      env:
        - name: ES_JAVA_OPTS
          value: "-Xms2g -Xmx2g"
      affinity:
        podAntiAffinity:

preferredDuringSchedulingIgnoredDuringExecution:
  - weight: 100
    podAffinityTerm:
      labelSelector:
        matchLabels:
          elasticsearch.k8s.elastic.co/cluster-
name: elasticsearch-flowx
      topologyKey: kubernetes.io/hostname
      # nodeSelector:
      #   diskType: ssd
      #   environment: production
      # request 2Gi of persistent data storage for pods in
      # this topology element
    volumeClaimTemplates:
      - metadata:
          name: elasticsearch-data
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 20Gi
      storageClassName: standard
```

(Optional) Kibana instance:

```
apiVersion: kibana.k8s.elastic.co/v1
kind: Kibana
metadata:
  name: kibana-flowx
  namespace: elastic-system
spec:
  version: 7.9.3
  count: 1
  elasticsearchRef:
    name: elasticsearch-flowx
    namespace: elastic-system
  config:
    elasticsearch.requestHeadersWhitelist:
      - authorization
  podTemplate:
    spec:
      containers:
        - name: kibana
          resources:
            requests:
              memory: 1Gi
              cpu: 0.5
            limits:
              memory: 3Gi
              cpu: 2
```

The index used by customer management plugin should be created.

## Postgres database

This plugin can work without this database, it will not store the audit data.

## Basic Postgres configuration

```
crmdb:
  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 it you will need to configure the connection details and index use to store customers.

```
elasticsearch:  
  ssl: false  
  nodes:  
    -  
      hostname: ${ELASTICSEARCH_HOST}  
      port: ${ELASTICSEARCH_PORT}  
      scheme: ${ELASTICSEARCH_HTTP_SCHEME}  
      user: ${ELASTICSEARCH_USER}  
      password: ${ELASTICSEARCH_PASSWORD}  
      customer-index: ${ELASTICSEARCH_CUSTOMER_INDEX}  
      size: 10
```

## Kafka configuration

The following Kafka related configurations can be set by using environment variables:

`SPRING_KAFKA_BOOTSTRAP_SERVERS` - address of the Kafka server

`SPRING_KAFKA_CONSUMER_GROUP_ID` - group of consumers

`KAFKA_CONSUMER_THREADS` - the number of Kafka consumer threads

`KAFKA_AUTH_EXCEPTION_RETRY_INTERVAL` - the interval between retries after `AuthorizationException` is thrown by `KafkaConsumer`

**KAFKA\_MESSAGE\_MAX\_BYTES** - this is the largest size of the message that can be received by the broker from a producer.

Each action available in the service corresponds to a Kafka event. A separate Kafka topic must be configured for each use-case.

#### CAUTION

The Engine is listening for messages on topics with names of a certain pattern, make sure to use correct outgoing topic names when configuring the documents plugin.

Needed topics:

**KAFKA\_TOPIC\_CUSTOMER\_SEARCH\_IN**

**KAFKA\_TOPIC\_CUSTOMER\_SEARCH\_OUT**

#### CAUTION

In order to match a request made to the customer management plugin, the engine will have to send the process id on a Kafka header.

## Logging

The following environment variables could be set in order to control log levels:

**LOGGING\_LEVEL\_ROOT** - root spring boot microservice logs

**LOGGING\_LEVEL\_APP** - app level logs

Was this page helpful?

# PLATFORM DEEP DIVE / Plugins / Plugins setup guides / Documents plugin setup / Configuring access rights for Documents

Granular access rights can be configured for restricting access to the Documents plugin component.

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

1. **Manage-document-templates** - for configuring access for managing document templates

Available scopes:

- import - users are able to import document templates
  - read - users are able to view document templates
  - edit - users are able to edit document templates
  - admin - users are able to publish or delete document templates
- The Document plugin is preconfigured with the following default users roles for each of the access scopes mentioned above:

- manage-document-templates
  - import:
    - ROLE\_DOCUMENT\_TEMPLATES\_IMPORT
    - ROLE\_DOCUMENT\_TEMPLATES\_EDIT
    - ROLE\_DOCUMENT\_TEMPLATES\_ADMIN
  - read:
    - ROLE\_DOCUMENT\_TEMPLATES\_READ
    - ROLE\_DOCUMENT\_TEMPLATES\_IMPORT
    - ROLE\_DOCUMENT\_TEMPLATES\_EDIT
    - ROLE\_DOCUMENT\_TEMPLATES\_ADMIN
  - edit:
    - ROLE\_DOCUMENT\_TEMPLATES\_EDIT
    - ROLE\_DOCUMENT\_TEMPLATES\_ADMIN
  - admin:
    - ROLE\_DOCUMENT\_TEMPLATES\_ADMIN

### ROLES

These roles need to be defined in the chosen identity provider solution.

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

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

```
SECURITY_ACCESSAUTHORIZATIONS_AUTHORIZATIONNAME_SCOPES_SCOPENAM  
E_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_RI  
ROLE_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_SCOPEROLE_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         |

| Environment Variable       | Definition  | Default Value |
|----------------------------|---|---------------|
| KAFKA_ADDRESS              | <p>The address of the Kafka bootstrap server in the format<br/><code>&lt;hostname&gt;:&lt;port&gt;</code></p> | -             |
| KAFKA_CONSUME_SCHEDULE     | <p>The interval (in seconds) at which Kafka messages are consumed</p>   | 30            |
| KAFKA_INPUT_TOPIC          | <p>The Kafka topic from which input messages are consumed</p>   | -             |
| KAFKA_OCR_CONSUMER_GROUPID | <p>The consumer group ID for the OCR Kafka consumer</p>   | ocr_group     |

| Environment Variable                | Definition   | Default Value |
|-------------------------------------|--|---------------|
| KAFKA_CONSUMER_AUTO_COMMIT          | Determines whether Kafka consumer commits offsets automatically                      | True          |
| 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         |

| Environment Variable             | Definition  | Default Value |
|----------------------------------|---|---------------|
| KAFKA_CONSUMER_MAX_POLL_INTERVAL | The maximum interval (in milliseconds) between consecutive polls for Kafka consume                            | 25000         |
| 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 |

| Environment Variable     | Definition   | Default Value | Example                       |
|--------------------------|--|---------------|-------------------------------|
| OAUTH_TOKEN_ENDPOINT_URI | The URI of the token endpoint for OAuth authentication | -             | <code>https://oauth...</code> |

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 | Example                   |
|----------------------|--|---------------|---------------------------|
| STORAGE_S3_HOST      | The host address of the S3 storage service | -             | <code>minio:west-1</code> |

| Environment Variable         | Definition  | Default Value |
|------------------------------|---|---------------|
| 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                              | -<br>eu-wes   |
| STORAGE_S3_OCR_SCANS_BUCKET  | The name of the S3 bucket for storing OCR scans                     | -<br>pdf-sc   |

| Environment Variable              | Definition  | Default Value |         |
|-----------------------------------|---|---------------|---------|
| STORAGE_S3_OCR_SIGNATURE_BUCKET   | The name of the S3 bucket for storing OCR signatures    | -             | extract |
| STORAGE_S3_OCR_SIGNATURE_FILENAME | The filename pattern for extracted OCR signatures       | -             | extract |
| 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 |
|------------------------|---|---------------|
| OCR_WORKER_COUNT       | Number of workers   | 5             |
| 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 user 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_ROLESALL  
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 interface with the 'Source' tab selected in the 'Data model' section of the 'ExampleTemplate' editor. The left sidebar contains navigation links for Processes, Client Management, Plugins, General Settings, and Platform status. The main area displays an HTML editor with a toolbar at the top, a preview area below it, and a code editor at the bottom. The code editor contains placeholder text: "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sollicitudin lacus non magna, at aliquet enim pellentesque. Nullam condimentum dapibus erat, non massa in euismus. Vivamus sed laus, posuere eget ultricies sapien, non in euismus euismus. Integer non nisi non ipsum pellentesque. Fusce congue odio, venenatis felis et fermentum. Vestibulum pulvinar metus vel sem et luctus vestibulum, non fermentum non convallis. In hac habitasse platea dictumst. Nulla sagittis nibh nec, elementum pretium turpis euismus et. Aenean viverra tortor, ornare nisl mi, imperdiet tristique dolor. Sed vel lorem commodo, blandit sapiens sed, sollicitudin orci. Pellentesque habitasse platea dictumst. Ut sed et tellus velit fermentum ac turpis egestas. Duis id rhoncus quam, interdum accumsan.

## 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 two tabs: "Body" (selected) and "Data model". The "Body" tab contains a rich text editor toolbar and a preview area. The preview area displays the following content:

```
Template generated for the FLOWX.AI Academy by

#firstInput

#secondInput



Leem 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, facilis euismod magna. Integer non nisi non ipsum porttitor pulvinar. Fusce commodo varius felis at tincidunt. Vestibulum pulvinar mauris sit amet nisl vestibulum, vitae facilisis sem convallis. In hac habitasse platea dictumst. Nulla sagittis nibh mi, venenatis


```

Below the preview area are three buttons: "Test", "Save", and "Publish".

» Documents plugin

## Notification Templates

Notification WYSIWYG body has some additional fields (other than documents template):

- **Type** - that could be either MAIL or SMS (SMS, only if there is an external adapter)
- **Forward on Kafka** - if this box is checked, the notification is not being sent directly by the plugin to the destination, but forwarded to another adapter

The screenshot shows the FLOWX.AI platform's Notifications Template editor. The left sidebar contains navigation links for Processors, Content Management, Plugins, General Settings, and Platform Status. The main area is titled "Notifications Template - ExampleTemplate". It has tabs for "Body" and "Data model". Under "Body", there is a "Type" dropdown set to "Mail", a "Forwarder Ref ID" input field, a "Message" input field containing "Hello {{Processor}} {{Processor}}", and a "Subject" input field containing "Contract {{Processor}}". Below these fields is a date picker. The "Data model" tab shows a JSON-like structure with fields "Title" (set to "Hello") and "Text" (set to "Salut {{firstinput}} {{secondinput}} ac lucioz sa te avea alaturi"). At the bottom are "Save" and "Delete" buttons.

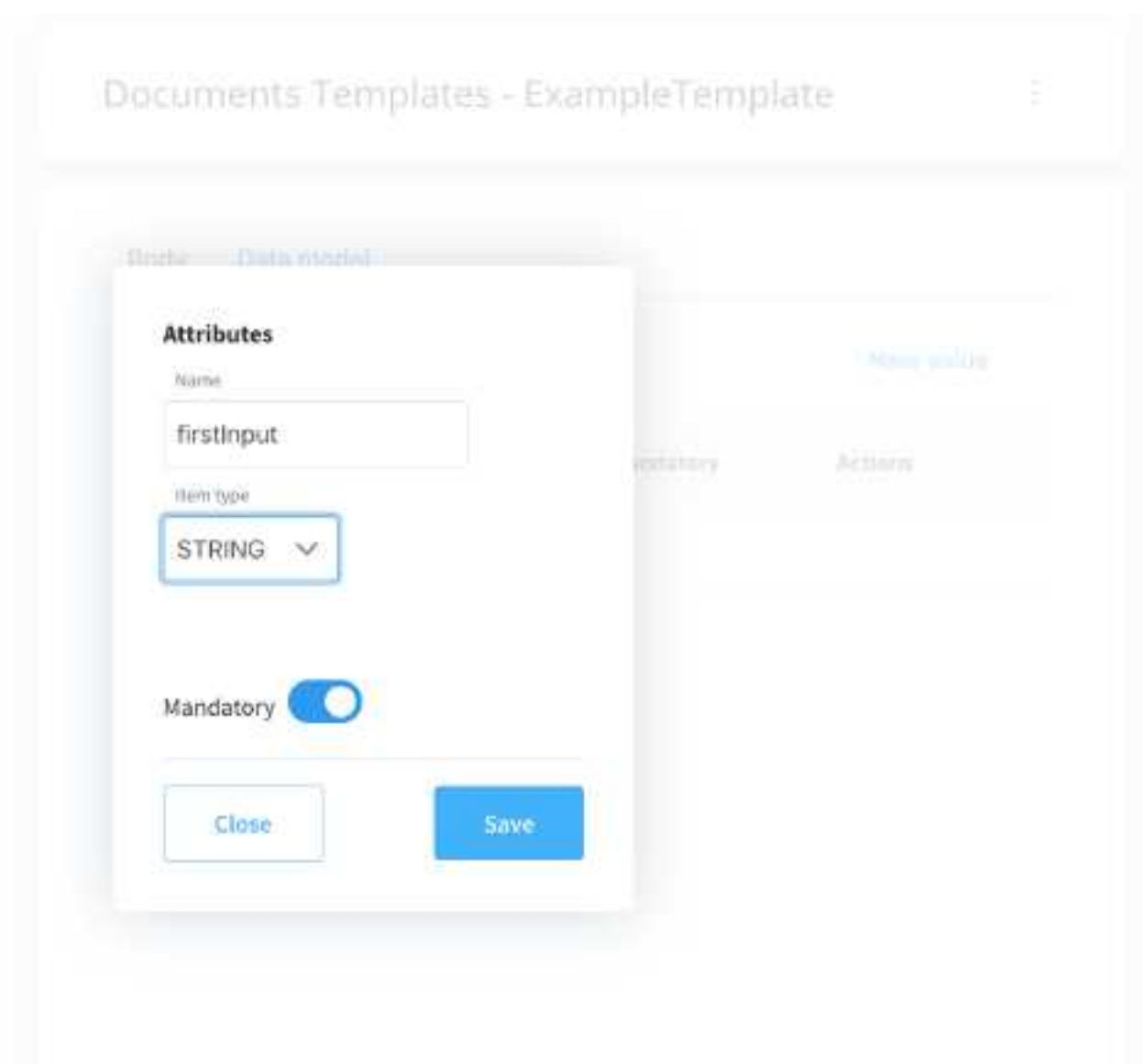
## » Managing notifications templates

# Data Model

## Data Model

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

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

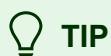


### (!) INFO

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                    |

| <b>FLOWX.AI Platform Version</b> | <b>Component name</b>        | <b>Supported/tested versions</b> |
|----------------------------------|------------------------------|----------------------------------|
| 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

| <b>FLOWX.AI Platform version</b> | <b>Component name</b> | <b>Supported/tested versions</b> |
|----------------------------------|-----------------------|----------------------------------|
| 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

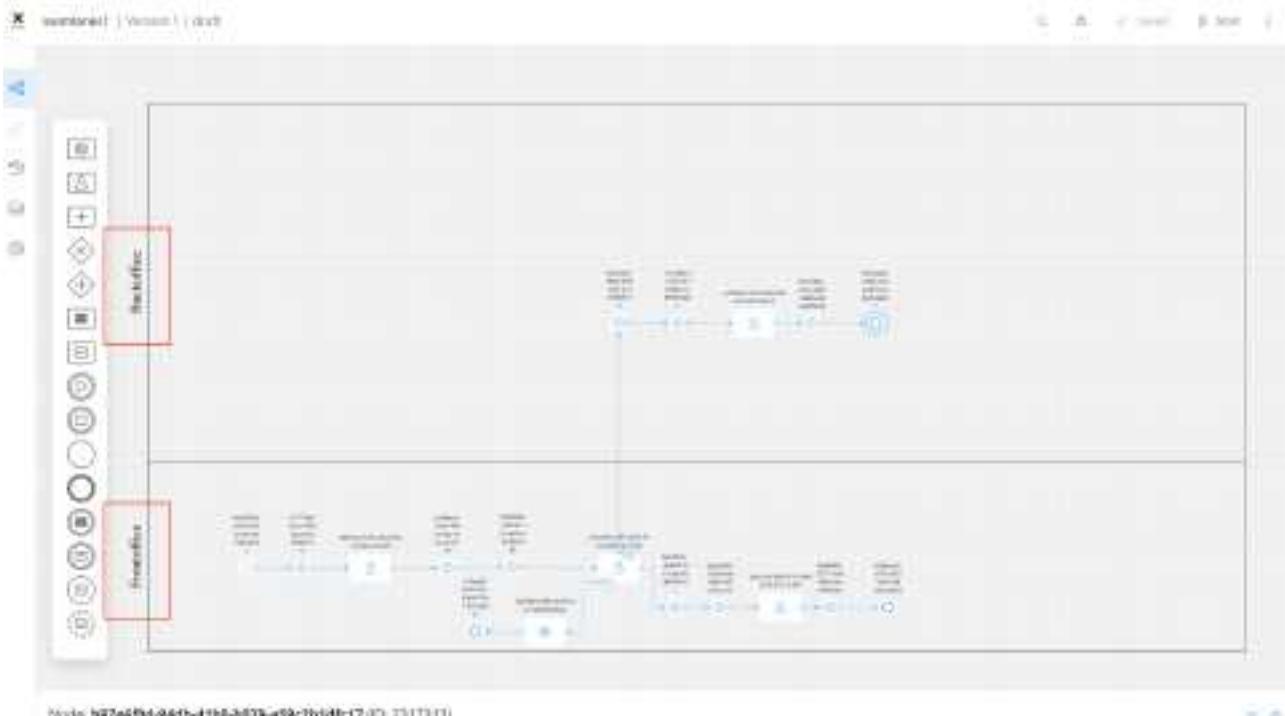
**What is it?** Swimlanes provide a way of grouping process nodes by process participants.

**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

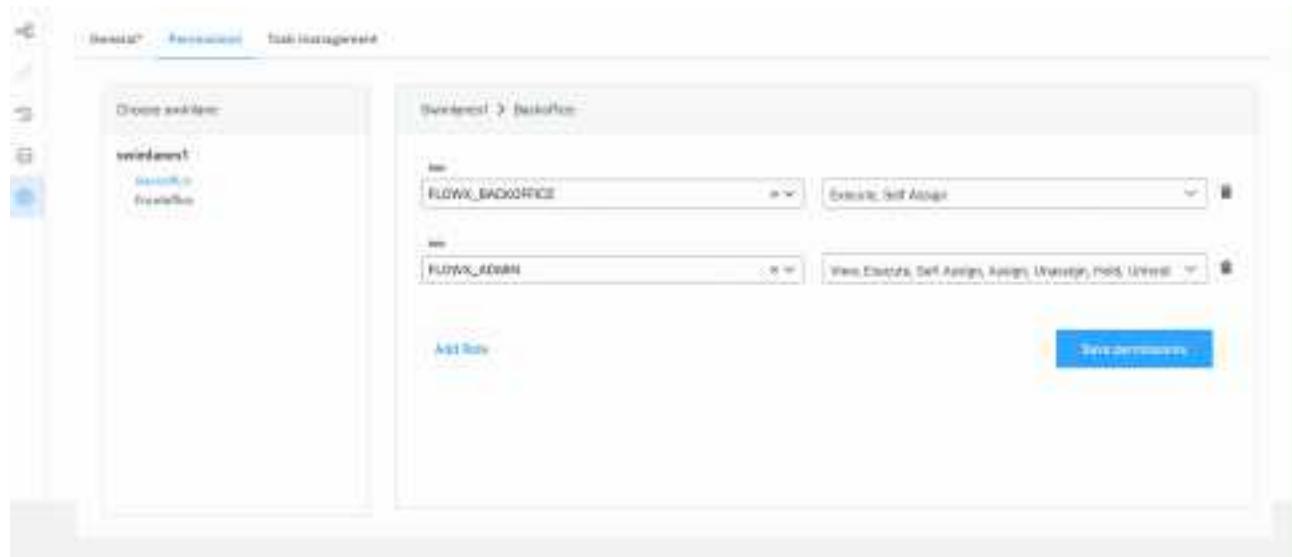
The fallback content to display on prerendering based on user roles. This can be achieved by organizing nodes into different swimlanes.

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 | 0.1     | 24 May 2023, 1:36 PM | admin FlowX  |
| m_35                        | 0.1     | 24 May 2023, 9:28 PM | QA FlowX     |
| Name                        | Version | Published at         | Published by |
| AutoTechProcess-2062203127  | 0.1     | 07 Apr 2023, 9:30 AM | QA FlowX     |
| AutoTechProcess-888325220   | 0.1     | 07 Apr 2023, 9:31 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.AI administrative interface. On the left, there is a sidebar with various menu items: Media Library, Plugins, Task Manager, All tasks, Home, Slides, Allocation rules, Octo office, Notification templates, Document templates, General Settings (which is currently selected), Generic Parameters, Environment management, Learning, Access management (highlighted with a red box), Audit Log, Platform status, and Admin Flowx. The main content area is titled 'Roles' and lists several roles with their descriptions and edit/delete icons:

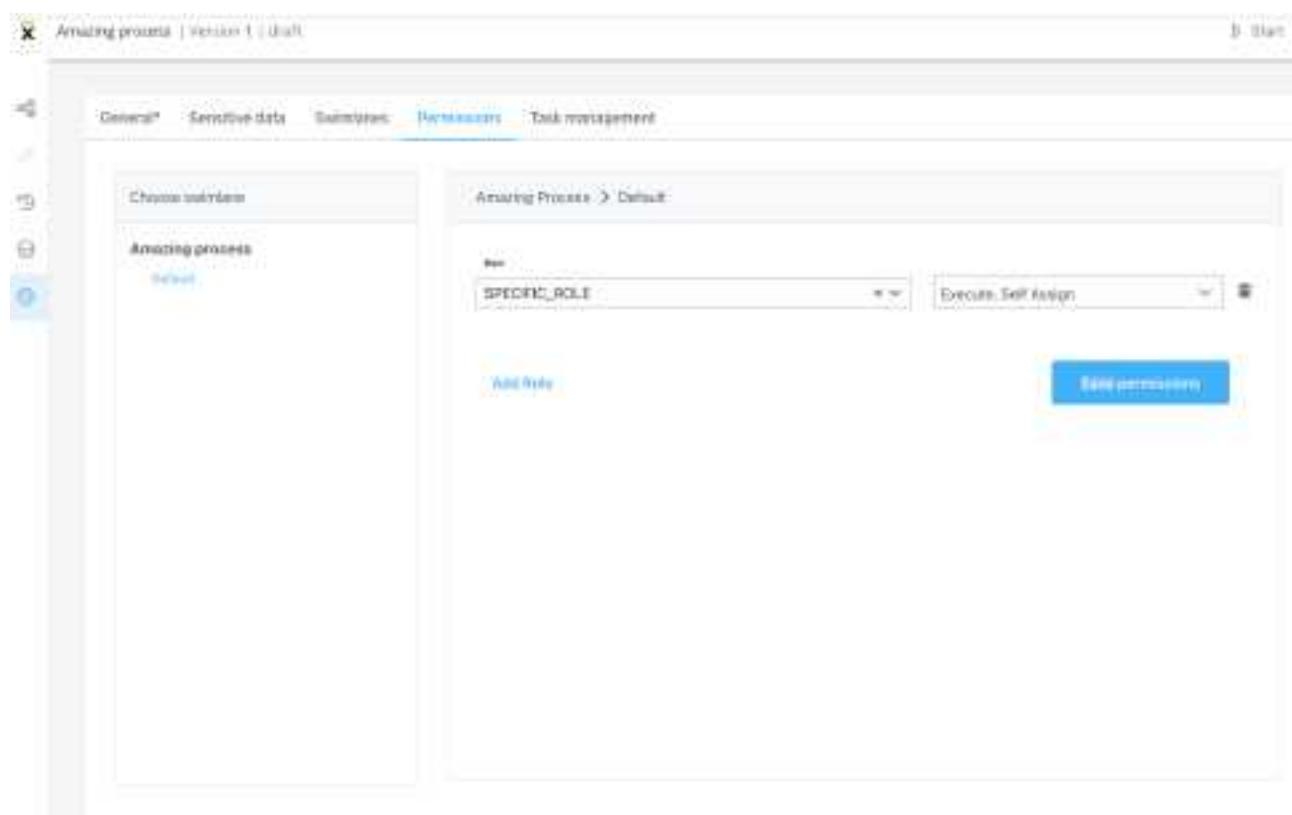
| Name                | Description   |
|---------------------|---|
| FLOWX_ADMIN         |   |
| FLOWX_BACKOFFICE    | FLOWX_BACKOFFICE  |
| FLOWX_FRONTOFFICE   | FLOWX_FRONTOFFICE   |
| FLOWX_ROLE          | Permission to start processes   |
| FLOWX_SUPERVISOR    | FLOWX_SUPERVISOR  |
| ROLE_ADMIN_MANAGE_C | Admin access - delete generic parameters menu item                        |
| ROLE_ADMIN_MANAGE_E | Edit access - create/edit generic parameters menu item                    |
| ROLE_ADMIN_IMPORT_C | Import access - Import generic parameters menu item                       |
| ROLE_ADMIN_MANAGE_R | Read access - View generic parameters menu item/export generic parameters |
| ROLE_ADMIN_MANAGE_D |   |

## 🔥 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.



- **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

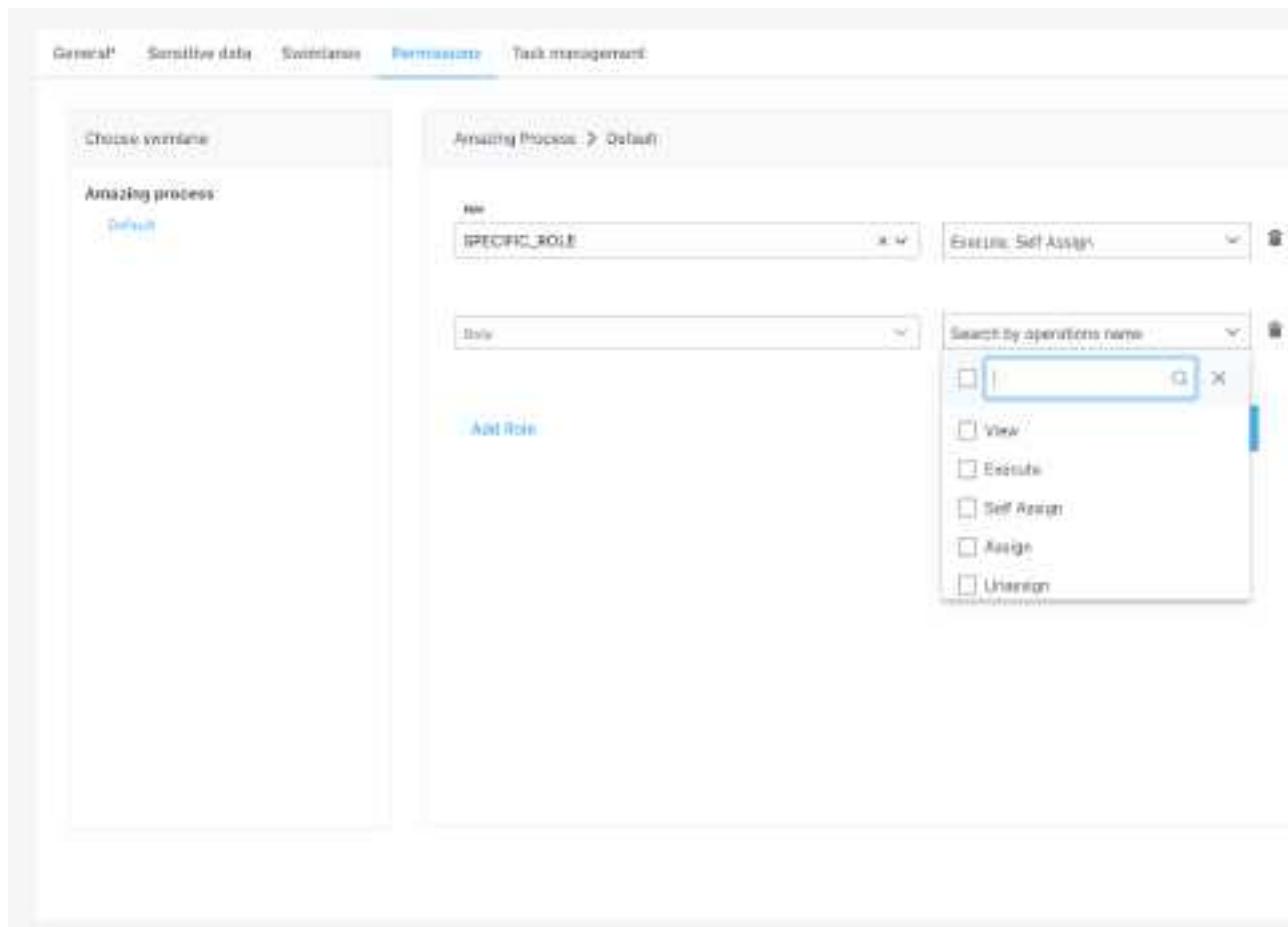


### DANGER

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)



### 🔥 DANGER

< 2.11.0 platform release - if no role is configured on an operation, no restrictions will be applied.

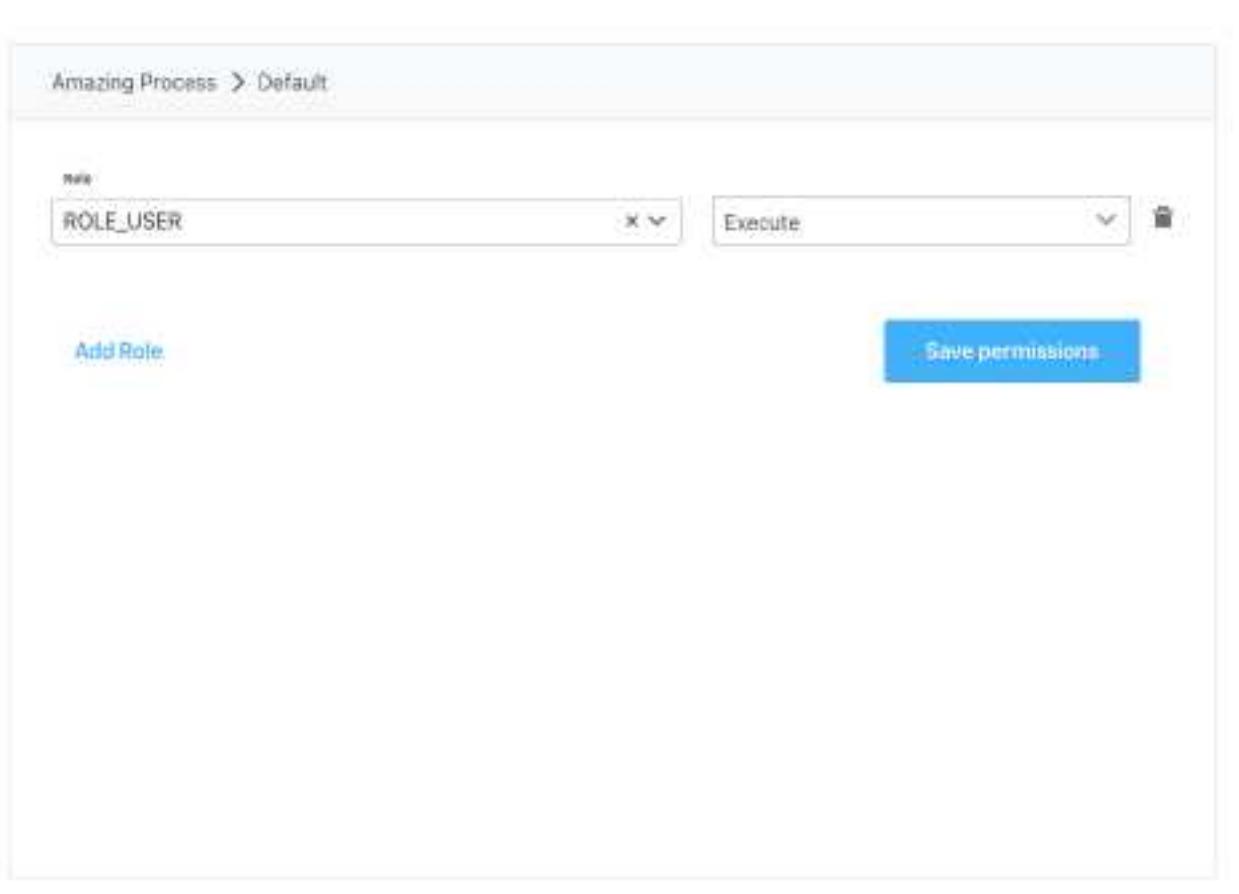
## Configuration examples

### ⚠ CAUTION

Valid for < 2.11.0 release version.

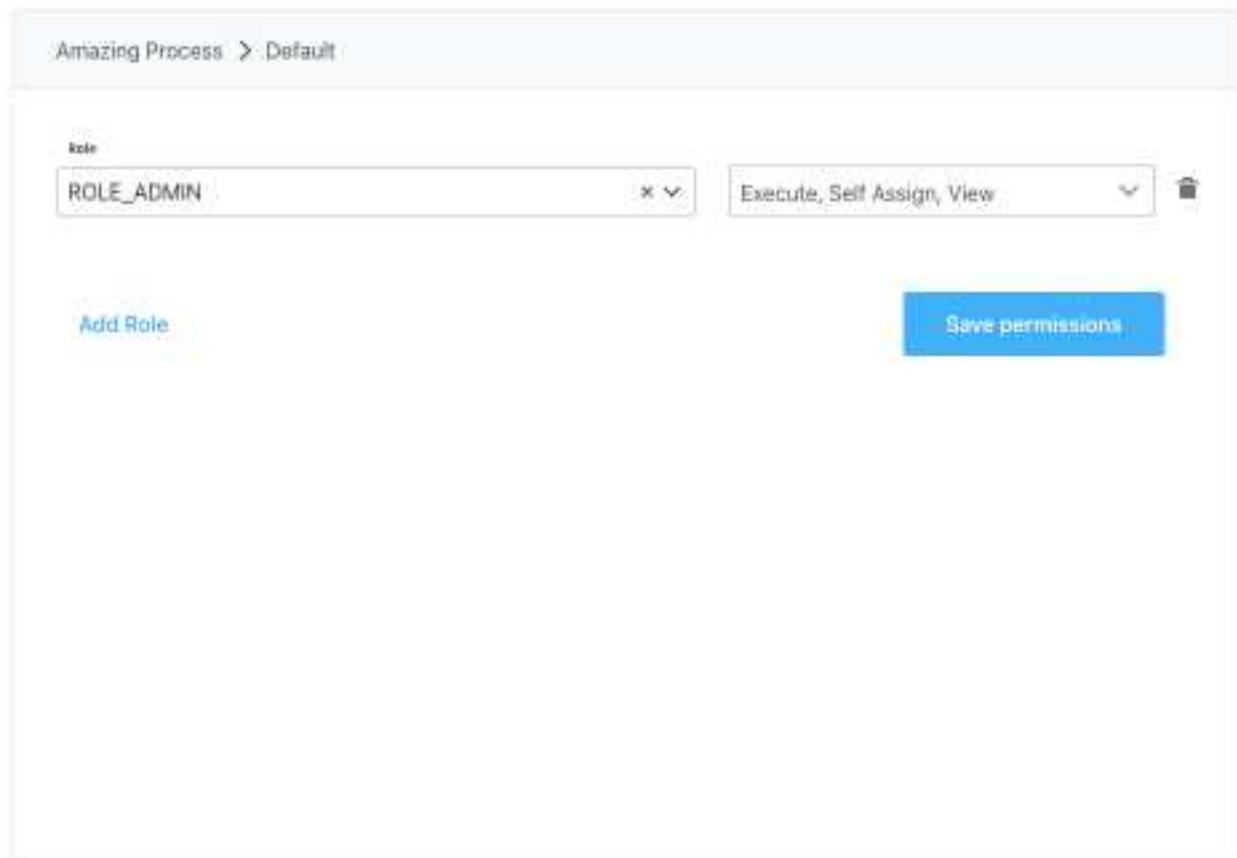
## Regular user

Below you can find an example of configuration of roles for a regular user:



## Admin

Below you can find an example of configuration of roles for an admin user:



### ⚠ 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 `FLOWX_ROLE`, but its name can be changed from the application configuration of the Engine by setting the following environment variable:

`FLOWX_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 `FLOWX_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**

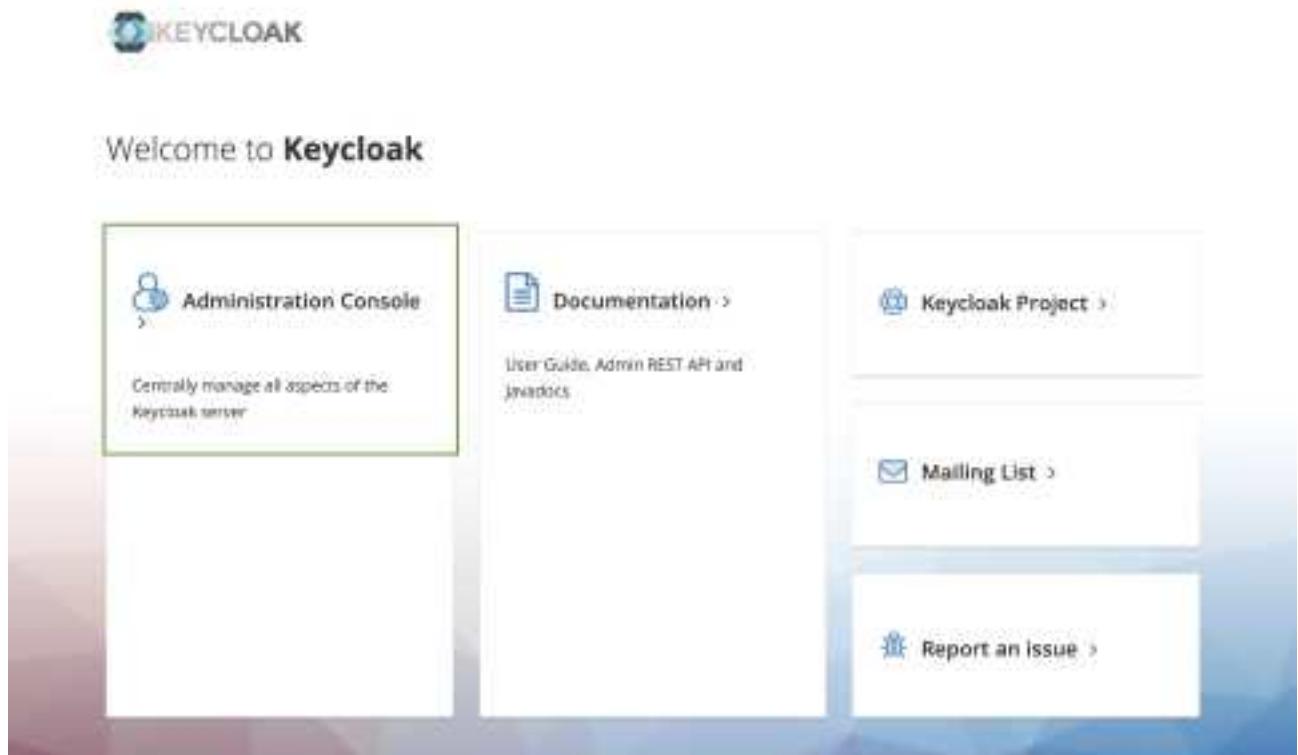
For more detailed information, refer to the official Keycloak documentation:

» [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).

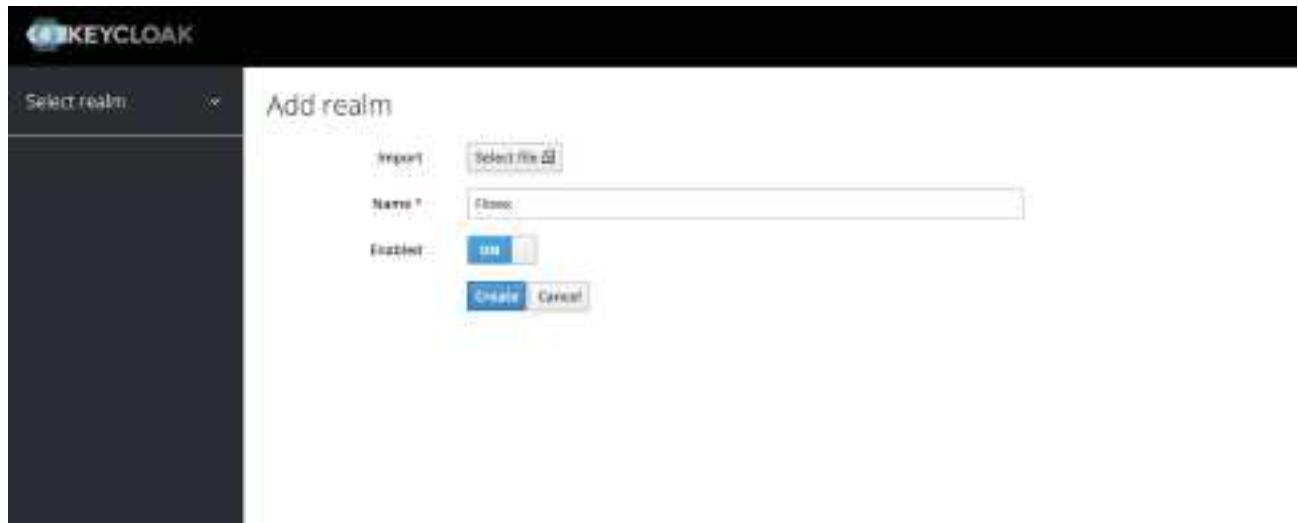


2. In the top left corner dropdown menu, click **Add Realm**.

**!** **INFO**

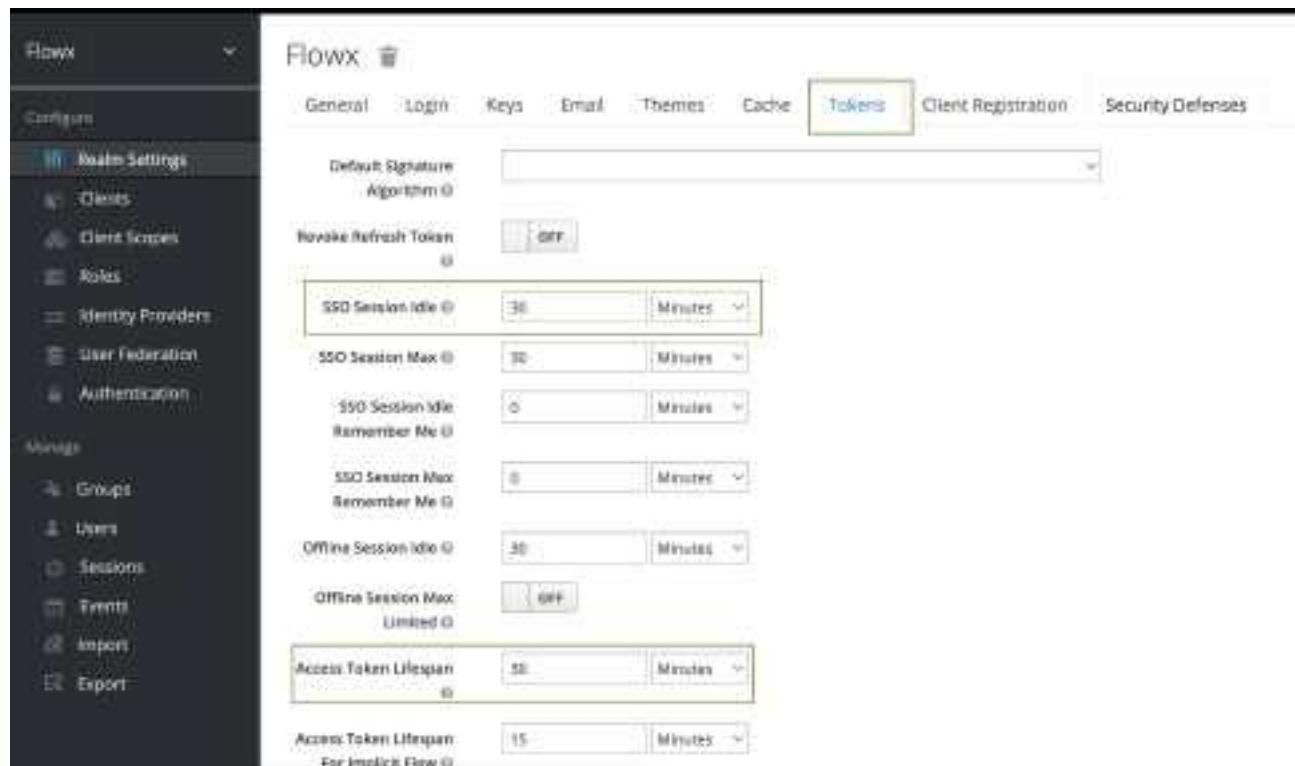
If you are logged in to the master realm this dropdown menu lists all the realms created. The **Add Realm** page opens.

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**



## 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.

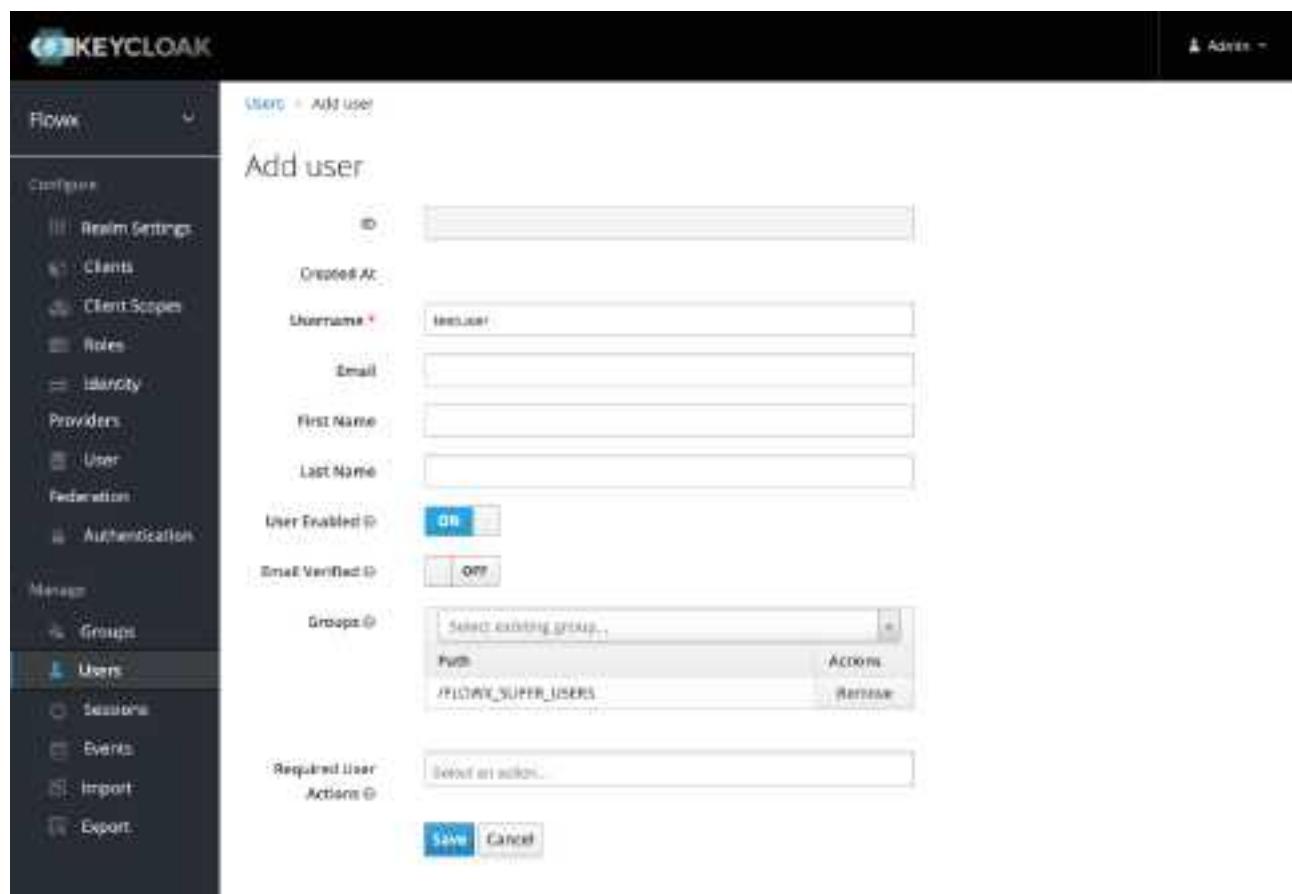
Make sure to validate the imported roles by checking the following section:

### » 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:  
**FL0WX\_SUPER\_USERS**.



The screenshot shows the 'Add user' form in the Keycloak admin interface. The 'Groups' dropdown menu is open, displaying a single option: '/FL0WX\_SUPER\_USERS'. This indicates that the user being created will be assigned to the 'FL0WX\_SUPER\_USERS' group.

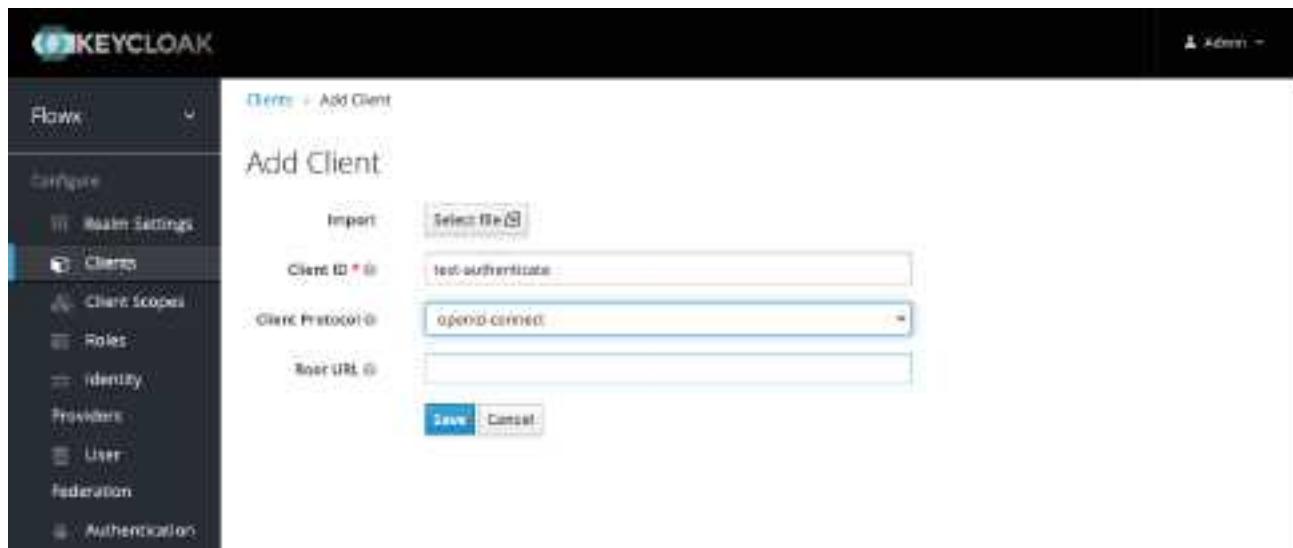
5. Save the user, go to the **Credentials** tab, and set a temporary password.

The screenshot shows the Keycloak administration interface. On the left, there's a sidebar with a 'Flowx' logo and several navigation items under 'Configure': Realm Settings, Clients, Client Scopes, Roles, Identity Providers, User, Federation, Authentication, Groups, and Users. The 'Users' item is currently selected. In the main content area, the URL 'Users > Testuser' is at the top, followed by the user name 'Testuser'. Below it, there are tabs: Details, Attributes, **Credentials**, Role Mappings, Groups, Consents, and Sessions. The 'Credentials' tab is active. Under 'Manage Credentials', there's a table with columns: Position, Type, User Label, Data, and Actions. A 'Set Password' section follows, containing fields for 'Password' and 'Password Confirmation', both of which are empty. Below these fields is a 'Temporary' checkbox, which is checked. At the bottom right 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`.



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 UI with the following details:

- Client ID:** test-authenticate
- Name:** (empty)
- Description:** test-authenticate
- Enabled:** ON
- Always Display in Console:** OFF
- Consent Required:** OFF
- Login Theme:** (empty)
- 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:8080/\*
- 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**.

The screenshot shows the 'Create Protocol Mapper' interface in Keycloak. The 'Mapper Type' is set to 'Group Membership'. The 'Token Claim Name' is 'groups'. Under 'Add group path', the 'Add to id token' and 'Add to access token' checkboxes are checked. The 'Add to user info' checkbox is unchecked. The 'Save' button is visible at the bottom.

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 'Business Filter Mapper' configuration page in the Keycloak interface. The page title is 'Business Filter Mapper'. The configuration fields are as follows:

- Protocol: openid-connect
- Protocol mappers: [empty]
- Name: businessFilterMapper
- Mapper Type: user Attribute
- User Attribute: businessFilters
- Token Claim Name: attributes.businessFilters
- Claim JSON Type: Select One... (dropdown menu)
- Add to ID token: OFF (radio button)
- Add to access token: ON (radio button)
- Add to refresh token: ON (radio button)
- Multivalued: ON (radio button)
- Aggregate attribute values: off (radio button)

At the bottom right 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

The screenshot shows a configuration form for a realm role. The fields are as follows:

- Protocol: OpenID Connect
- Name: admin-role
- Realm-type: User Realm-Role
- Realm Data profile: (empty)
- Multivalued: checked
- Token claim name: role
- Client (SAML Type): (empty)
- Add as ID token: checked
- Add to access token: checked
- Add to refresh token: checked

At the bottom are 'Save' and 'Cancel' buttons.

## 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 Keycloak administration interface under the 'Clients' section for a client named 'Flowx-platform-authorize'. The 'Settings' tab is active, showing the following configuration:

- Client ID:** Flowx-platform-authorize
- Name:** (empty)
- Description:** (empty)
- Enabled:** ON
- Always Display in Console:** OFF
- Consent Required:** OFF
- Login Theme:** (empty)
- 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 CMA Grant Enabled:** OFF
- Authorization Enabled:** OFF
- Front Channel Logout:** OFF
- Root URL:** (empty)
- Valid Redirect URLs:** http://localhost:8080/\*

## 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 'Clients' section selected under 'Configure'. The main area is titled 'Add Client' with a sub-header 'Clients / Add Client'. It contains fields for 'Client ID' (with placeholder 'Select file'), 'Client Protocol' (set to 'openid-connect'), and 'Root URL' (empty). There are 'Save' and 'Cancel' buttons at the bottom. The top right corner shows a user icon and the text 'Admin'.

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                      | ON                    |               |         |       |            |          |                |              |
| Consent Required             | OFF                   |               |         |       |            |          |                |              |
| Login Theme                  |                       |               |         |       |            |          |                |              |
| Client Protocol              | openid-connect        |               |         |       |            |          |                |              |
| Access Type                  | confidential          |               |         |       |            |          |                |              |
| Standard Flow Enabled        | ON                    |               |         |       |            |          |                |              |
| Implicit Flow Enabled        | OFF                   |               |         |       |            |          |                |              |
| Direct Access Grants Enabled | ON                    |               |         |       |            |          |                |              |
| Service Accounts Enabled     | ON                    |               |         |       |            |          |                |              |

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**:

The screenshot shows the 'Service Account Roles' tab selected in the top navigation bar. A search bar at the top right contains the text 'service account Flowx-admin-service account'. Below it, there are two sections: 'Realm Roles' and 'Client Roles'. Under 'Realm Roles', the 'Available Roles' list includes FLOWX\_ADMIN, FLOWX\_BACKOFFICE, FLOWX\_MONITOR, FLOWX\_B2B, and FLOWX\_MANAGEMENT. The 'Assigned Roles' list contains 'admin-role-flowx'. The 'Effective Roles' list includes 'admin-role-flowx', 'offline-access', and 'uma\_authorization'. Under 'Client Roles', the 'Available Roles' list includes 'realm-management', 'manage-clients', 'introspection', 'manage-authentications', 'manage-clients', and 'manage-realm'. The 'Assigned Roles' list contains 'query-groups', 'query-users', and 'view-users'. The 'Effective Roles' list includes 'realm-management', 'query-groups', and 'view-users'. Buttons for 'Add selected' and 'Remove selected' are visible between the lists.

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**.

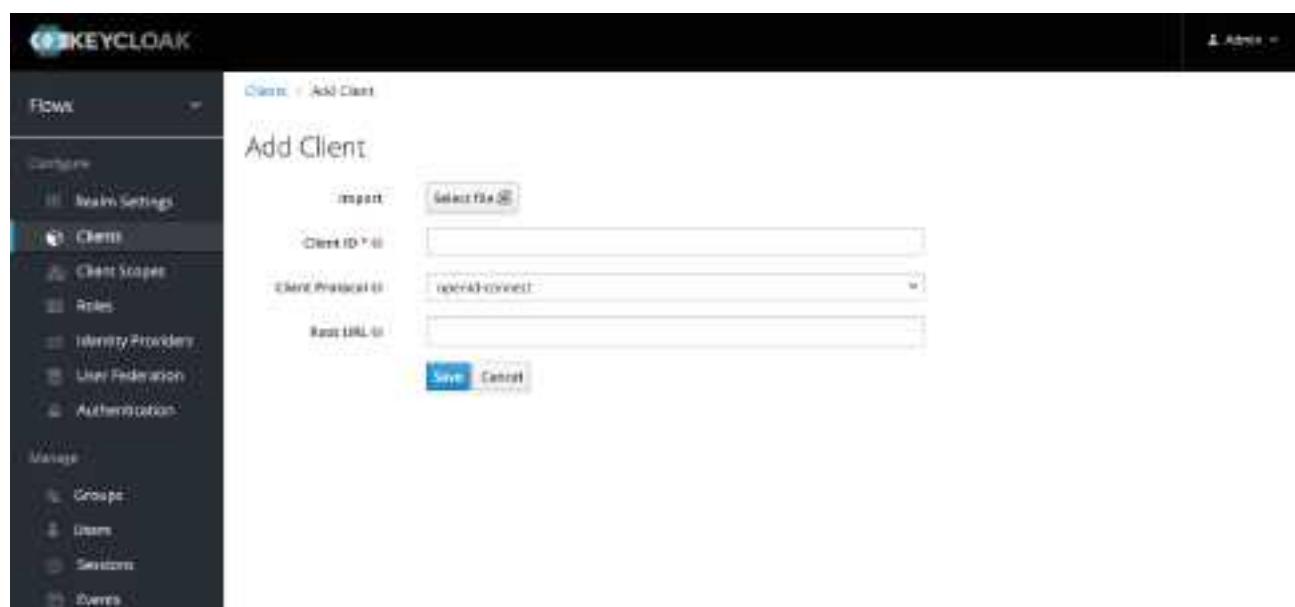
For detailed information, refer to the following section:

» [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**.



2. Next, set the following properties:

- **Access type** - confidential
- **Service Accounts Enabled** - ON

## Task-man-service-account

Settings    Credentials    Roles    Client Scopes     Mappers     Scope     Revocation

Installation     Service Account Roles 

|   |  |
|---|--|
| Client ID                          | task-man-service-account   |
| Name                               |  |
| Description                        |  |
| Enabled                            | ON   |
| Consent Required                   |  OFF  |
| Login Theme                      |  |
| Client Protocol                  | openid-connect   |
| Access Type                      | confidential   |
| Standard Flow Enabled            |  ON |
| Implicit Flow Enabled            |  ON |
| Direct Access Grants<br>Enabled  |  ON |
| Service Accounts<br>Enabled      |  ON |

3. Go to **Clients** → **realm-management** → **Roles** and add the following **service account client roles**:

- **view-users**
- **query-groups**
- **query-users**

Client > Identity Management > Roles > view-user

### View-users

Details   Attributes   Users in Role

Role Name: view-user

Description: Role\_view-user

Composite Roles:

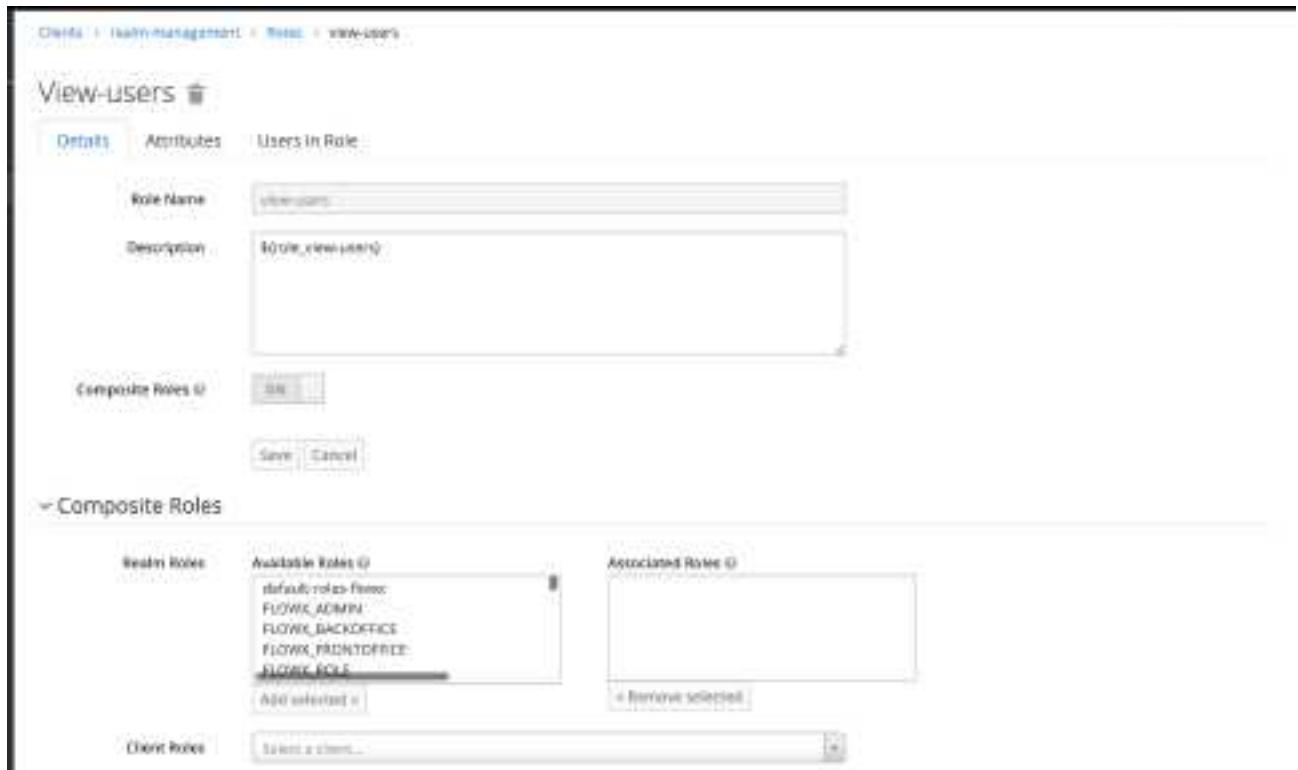
Realm Roles: Available Roles: FLOWX\_ROLE (selected)

Associated Roles:

Client Roles: Select a client...

Save   Cancel

Composite Roles



#### 4. Configure a realm roles mapper:

Realm-roles

Protocol ID: openid-connect

ID:

Name ID:

Mapper Type: User-Name-Mapper

Realm Role profile:

Multivalued:

Token Claim Name:

Claim-JSON Type:

Add to ID token:

Add to access token:

Add to user info:

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: FLOWX\_ADMIN, FLOWX\_BACKOFFICE, FLOWX\_FRONTOFFICE, FLOWX\_SUPERVISOR, view-users  
Selected: view-users  
Add selected Remove selected

Assigned Roles: default-roles-flowx, FLOWX\_ROLE  
Effective Roles: default-roles-flowx, FLOWX\_ROLE, offline-access, view-administration

Client Roles Available Roles: create-client, impersonate, manage-authorization, manage-items, manage-events  
Selected: task-management  
Add selected Remove selected

Assigned Roles: query-groups, query-users, view-users  
Effective Roles: query-groups, query-users, view-users

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                                      | ON                      |
| Always Display in Console                    | OFF                     |
| Consent Required                             | OFF                     |
| Login Theme                                  |                         |
| Client Protocol                              | openid-connect          |
| Access Type                                  | confidential            |
| Standard Flow Enabled                        | OFF                     |
| Implicit Flow Enabled                        | OFF                     |
| Direct Access Grants Enabled                 | OFF                     |
| Service Accounts Enabled                     | ON                      |
| OAuth 2.0 Device Authorization Grant Enabled | OFF                     |
| OIDC CBA Grant Enabled                       | 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                         | ON                    |
| Consent Required                | OFF                   |
| Login Theme                     |                       |
| Client Protocol                 | openid-connect        |
| Access Type                     | confidential          |
| Standard Flow Enabled           | ON                    |
| Implicit Flow Enabled          | OFF                   |
| Direct Access Grants Enabled  | ON                    |
| Service Accounts Enabled      | ON                    |

 INFO

This service account does not require client roles.

3. Assign the necessary service account roles, including `FL0WX_ROLE`.

The screenshot shows the 'Service Accounts' section of the FLOWX.AI Identity and Access Management interface. It includes four tables:

- Admin Roles:** Contains roles such as FULL\_ACCESS\_ADMIN, FULL\_ACCESS\_USER, etc.
- Assigned Roles:** Contains roles such as ROLE\_ADMIN, ROLE\_USER, etc.
- Client Roles:** Contains roles such as ADMINISTRATOR, OPERATOR, etc.
- Default Roles:** Contains roles such as OPERATOR, etc.

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 |
|--------|--------|--------------------|
|--------|--------|--------------------|

| <b>Module</b>       | <b>Scopes</b> | <b>Role default value</b>            |
|---------------------|---------------|--------------------------------------|
| 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  |

| <b>Module</b>         | <b>Scopes</b> | <b>Role default value</b>             |
|-----------------------|---------------|---------------------------------------|
| 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       |

| <b>Module</b>    | <b>Scopes</b> | <b>Role default value</b>         |
|------------------|---------------|-----------------------------------|
| 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             |

| <b>Module</b>        | <b>Scopes</b> | <b>Role default value</b>  |
|----------------------|---------------|----------------------------|
| 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  |

| <b>Module</b>                      | <b>Scopes</b> | <b>Role default value</b>                            |
|------------------------------------|---------------|--|
| 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   |

| <b>Module</b>                      | <b>Scopes</b> | <b>Role default value</b>            |
|------------------------------------|---------------|--------------------------------------|
| 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          |

| <b>Module</b>                 | <b>Scopes</b> | <b>Role default value</b>          |
|-------------------------------|---------------|------------------------------------|
| 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?