FLOWX.AI

BUILDING BLOCKS / Node / Message events

Contents

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

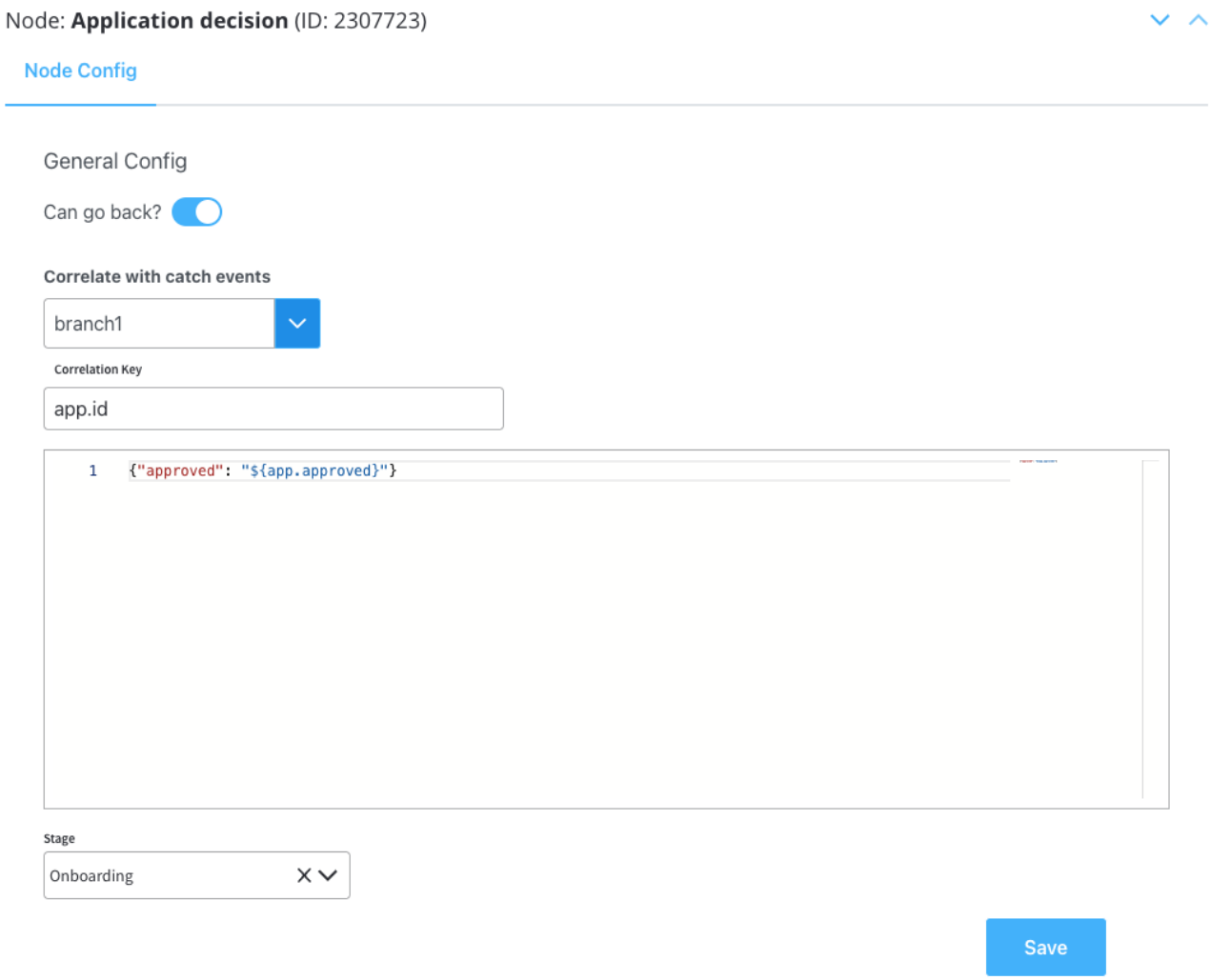
> information with each other.

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



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

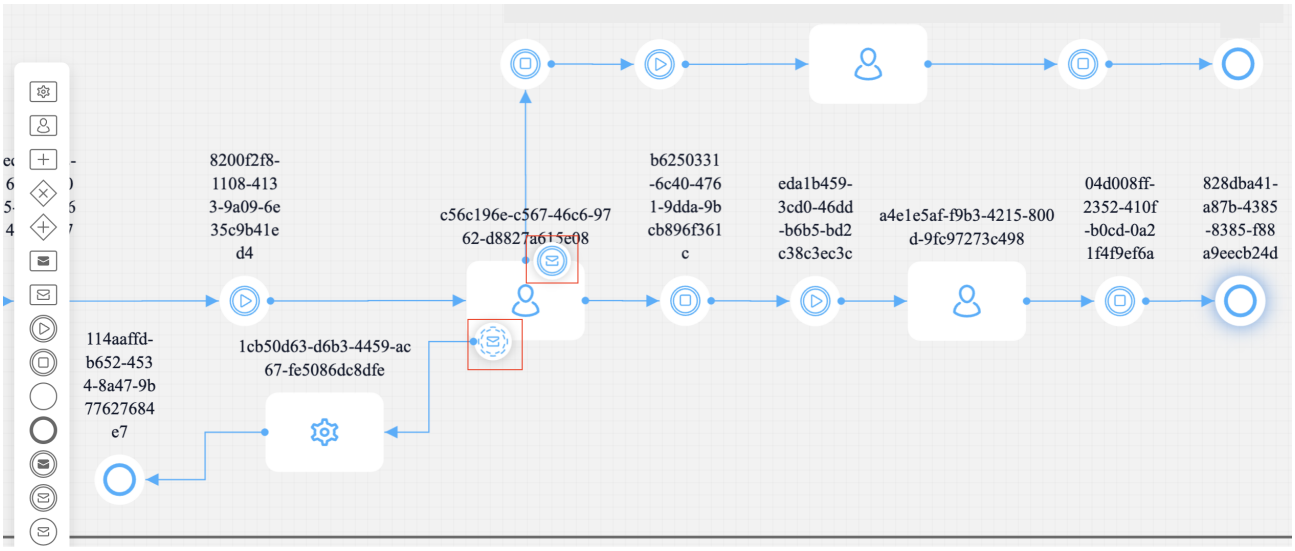
The fallback content to display on prerendering
that can be attached to a user task in a
The fallback content to display on prerendering
.

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

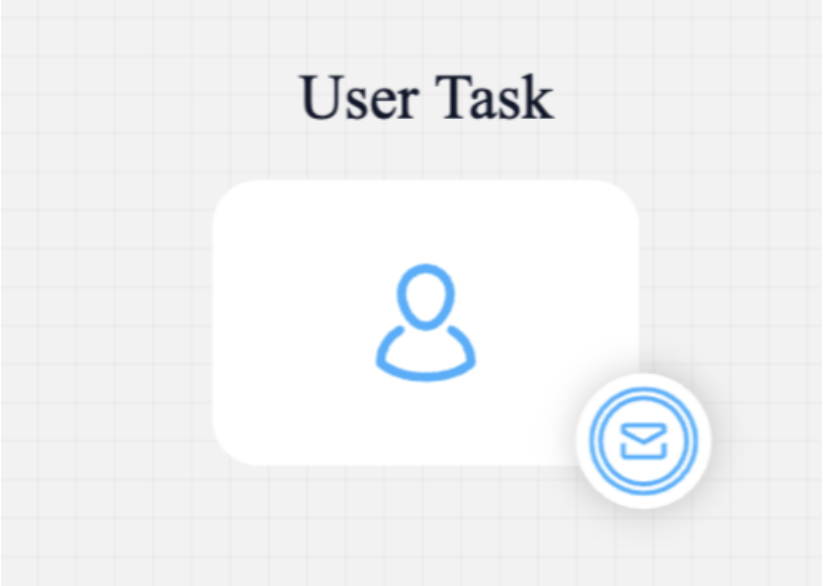
The fallback content to display on prerendering
. You can have multiple intermediate boundary events on the same user task, but only one can be activated at a time.



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.

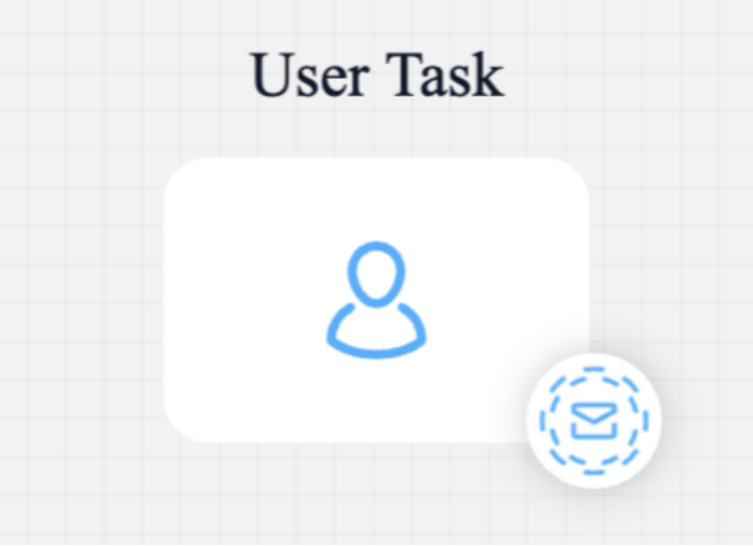
It can also be used as a standalone node, see more information on the following section:

» 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

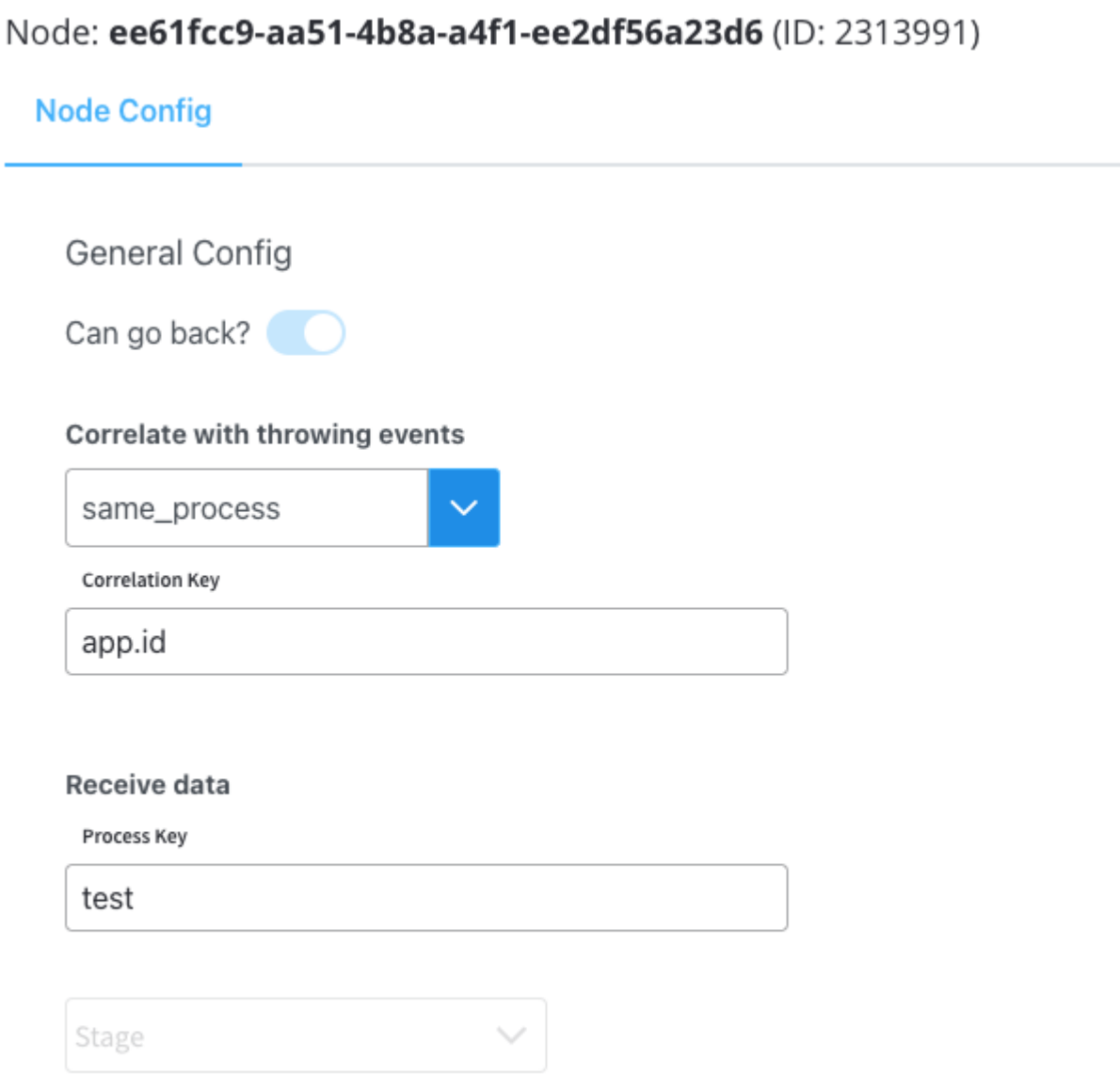
> Message Catch Intermediate Event can be used as a standalone node, this means that it will block a process until it receives an event.

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



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.

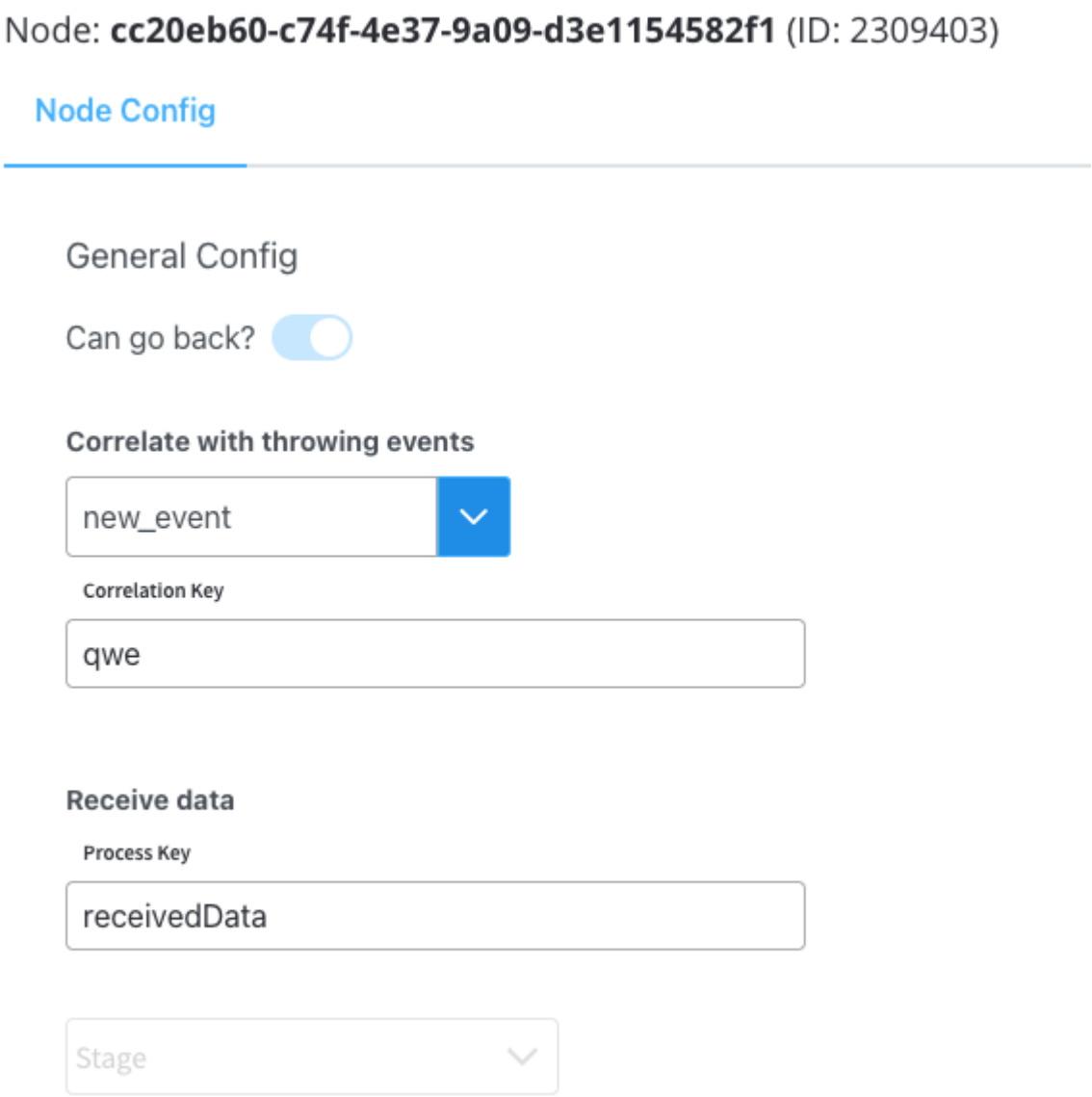
CAUTION

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



Was this page helpful?