



**PLATFORM DEEP DIVE / Core components / Core extensions / search-data-service**

# Contents

- PLATFORM DEEP DIVE / Core components / Core extensions / Search data service
  - Using search data

## 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**.

The screenshot displays the FLOWX.AI interface for configuring a workflow. The top section shows a workflow diagram titled "search\_in\_another\_proc (DRAFT)". The diagram consists of several nodes connected by arrows. A search bar above the diagram allows filtering by node name. The "search\_node" is highlighted. Below the diagram, the "Node: search\_node (ID: 648104)" configuration panel is open. The "Actions" tab is selected, showing a list of actions on the left and the "Action Edit" form on the right. The "Action Edit" form includes fields for "Name" (search\_for\_cnp), "Order" (0), and "Timer Expression". The "Kafka Send Action" is selected from a dropdown menu. The "Automatic" radio button is selected, and the "Optional" radio button is also selected. The "Repeatable" checkbox is checked. A "Save" button is located at the bottom right of the form.

Node: **search\_node** (ID: 648104)

Node Config | **Actions**

Actions +

- search\_for\_cnp +
- search\_data\_cnp +
- start\_subprocess +
- save\_data +

**Action Edit**

ID: 648254

Name: search\_for\_cnp

Order: 0

Timer Expression

Kafka Send Action

☐ Automatic ☒ Manual

☐ Mandatory ☒ Optional

☒ Repeatable

Save

### 4. Configure the following items:

- **Topic name** - the Kafka topic on which the search service listens for requests; **!** respect the **naming pattern**
- **Data to send** - (key) - used when data is sent from the frontend via an action to validate the data (you can find more information in the User Task configuration section)

- **Headers** - required
- **Body message:**
  - `searchKey` - it will hold the result received from the elastic search
  - `value` - value of the key
  - `processDefinitionNames` - the process definition names where to perform the search
  - `processStartDateAfter` - the service will look into process definitions created after the defined date

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

- Example (dummy values extracted from a process):

**Topics**

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

**Message**

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

**Advanced configuration**Show Headers ☒

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

**Data to send**[Add 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):

#### TIP

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

Process definition: `search_in_another_proc` Active process instance: `a4001bf0-...`

Variables Tokens Subprocesses

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

! If there is a list of results:

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

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

- Example (dummy values extracted from a process):



Process definition: `search_in_another_proc` Active process instance: `5929721e...`[Variables](#) [Tokens](#) [Subprocesses](#)

```
processInstanceId: 665917
searchResult:
  searchKey: "application.client.identificationData.personalIdentificationNumber"
  result:
    - 0: Object {"processInstanceUUID": "16ce7721-97a0-40be-b88b-61d6e906d208", "state": "FINISHED", "processStartDate": "2022-09-13T13:59:56.131Z", "data": {"application": {"client":
    - 1: Object {"processInstanceUUID": "c8716ea6-0b3b-448d-a814-c84db6347ff7", "state": "FINISHED", "processStartDate": "2022-09-13T14:28:48.988Z", "data": {"application": {"client":
    - 2: Object {"processInstanceUUID": "11f0503b-fd25-4e38-8744-42cef8838fbc", "state": "FINISHED", "processStartDate": "2022-09-13T14:44:59.061Z", "data": {"application": {"client":
  tooManyResults: false
resultsNumber: 3
redirectUrl: "https://designer.flowxai.dev/processes/instance?processUid=11f0503b-fd25-4e38-8744-42cef8838fbc"
tokenId: 665969
searchValue2: "1871201460000"
tokenUid: "a057acf5-36ea-d410-abbd-35c062398ebc"
websocketPath: "/ws/updates/process"
processInstanceUid: "5929721e-1a5a-4ebd-87d5-72eab51928f"
websocketAddress: "wss://public.qa.flowxai.dev/5929721e-1a5a-4ebd-87d5-72eab51928f"
```

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

» [Search data service setup guide](#)

Was this page helpful?