FLOWX.AI

PLATFORM DEEP DIVE / Core components / Core extensions / Content management / using-the-service

Contents

- 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 / 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?