Documentation

I. Introduction

MPO is a low-code platform based on Corezoid, which lets users create complex flows using little to no coding. This example uses processes to search for a movie, take its title, actors, writers and directors and search them on Wikipedia, and concatenate the articles to each of their arrays in the resulting JSON. Coding is used mostly for formatting the results.

II. Main Process

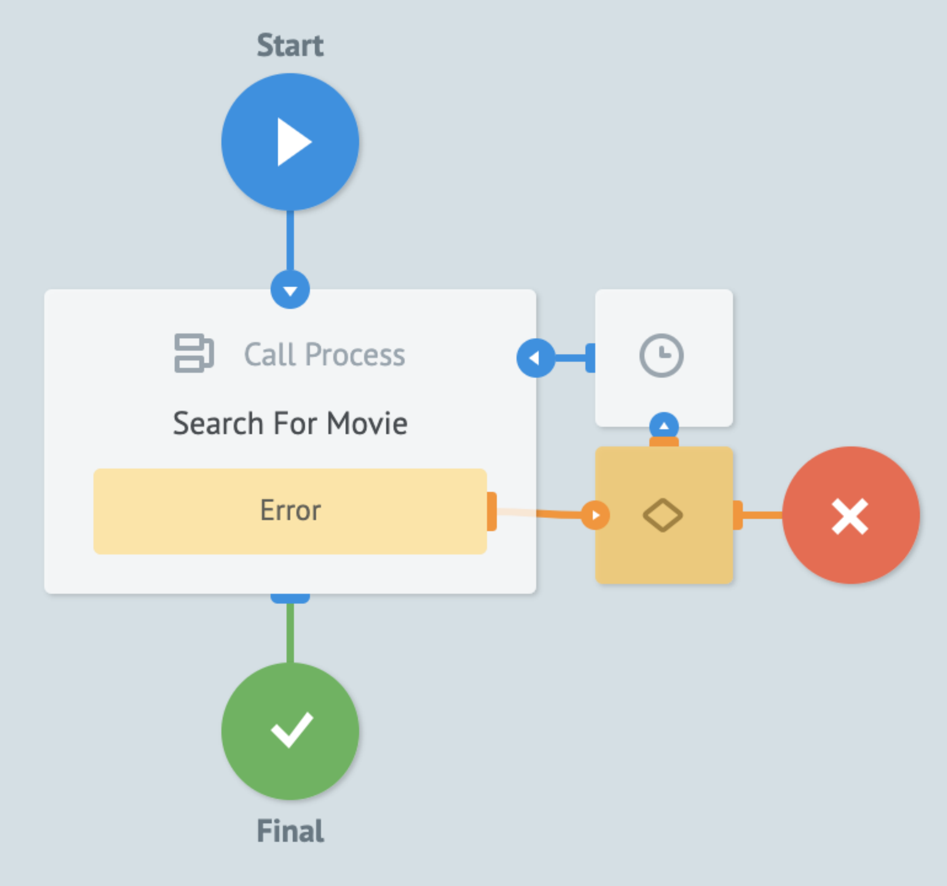
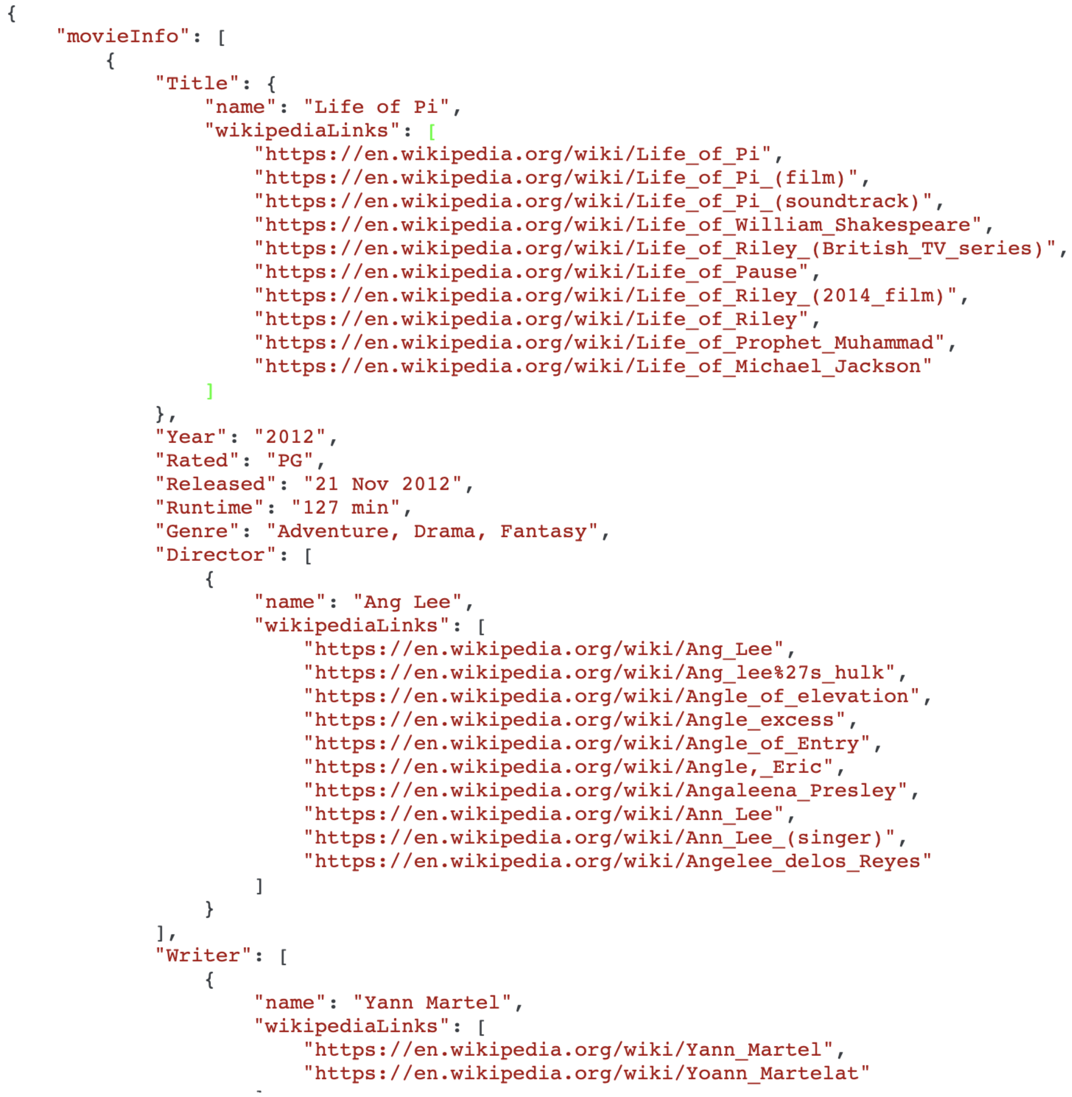


Fig. 1 - Main Process in MPO

The Main Process is used to send a query and get the response automatically. It calls one of the business processes and gets the result without knowing the logic behind it. This is useful in case of scalability, webhooks and ease of use.



```
{
    "movieInfo": [
        {
            "Title": {
                "name": "Life of Pi",
                "wikipediaLinks": [
                    "https://en.wikipedia.org/wiki/Life_of_Pi",
                    "https://en.wikipedia.org/wiki/Life_of_Pi_(film)",
                    "https://en.wikipedia.org/wiki/Life_of_Pi_(soundtrack)",
                    "https://en.wikipedia.org/wiki/Life_of_William_Shakespeare",
                    "https://en.wikipedia.org/wiki/Life_of_Riley_(British_TV_series)",
                    "https://en.wikipedia.org/wiki/Life_of_Pause",
                    "https://en.wikipedia.org/wiki/Life_of_Riley_(2014_film)",
                    "https://en.wikipedia.org/wiki/Life_of_Riley",
                    "https://en.wikipedia.org/wiki/Life_of_Prophet_Muhammad",
                    "https://en.wikipedia.org/wiki/Life_of_Michael_Jackson"
                ]
            },
            "Year": "2012",
            "Rated": "PG",
            "Released": "21 Nov 2012",
            "Runtime": "127 min",
            "Genre": "Adventure, Drama, Fantasy",
            "Director": [
                {
                    "name": "Ang Lee",
                    "wikipediaLinks": [
                        "https://en.wikipedia.org/wiki/Ang_Lee",
                        "https://en.wikipedia.org/wiki/Ang_lee%27s_hulk",
                        "https://en.wikipedia.org/wiki/Angle_of_elevation",
                        "https://en.wikipedia.org/wiki/Angle_excess",
                        "https://en.wikipedia.org/wiki/Angle_of_Entry",
                        "https://en.wikipedia.org/wiki/Angle,_Eric",
                        "https://en.wikipedia.org/wiki/Angaleena_Presley",
                        "https://en.wikipedia.org/wiki/Ann_Lee",
                        "https://en.wikipedia.org/wiki/Ann_Lee_(singer)",
                        "https://en.wikipedia.org/wiki/Angelee_delos_Reyes"
                    ]
                }
            ],
            "Writer": [
                {
                    "name": "Yann Martel",
                    "wikipediaLinks": [
                        "https://en.wikipedia.org/wiki/Yann_Martel",
                        "https://en.wikipedia.org/wiki/Yoann_Martelat"
```

Fig. 2 - Part of response given to “life of pi”

III. Business Processes

A. “Search Movie and Format”

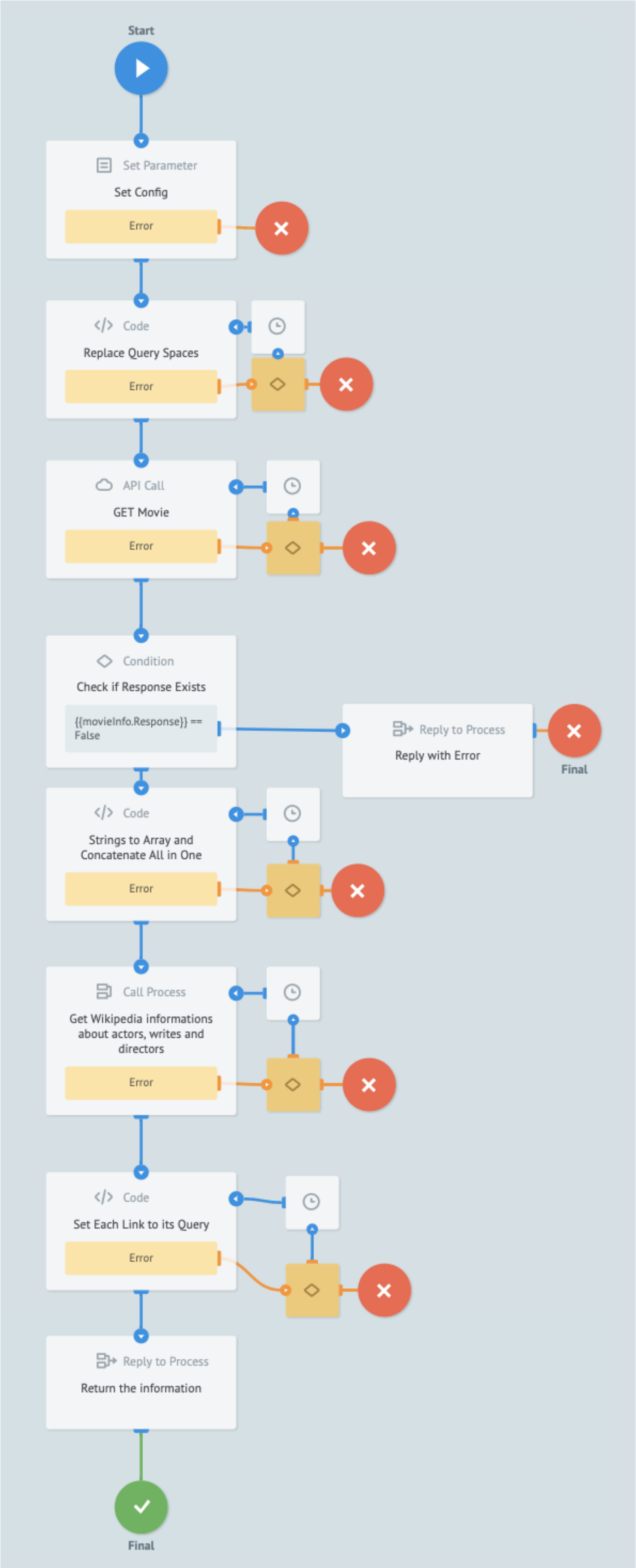
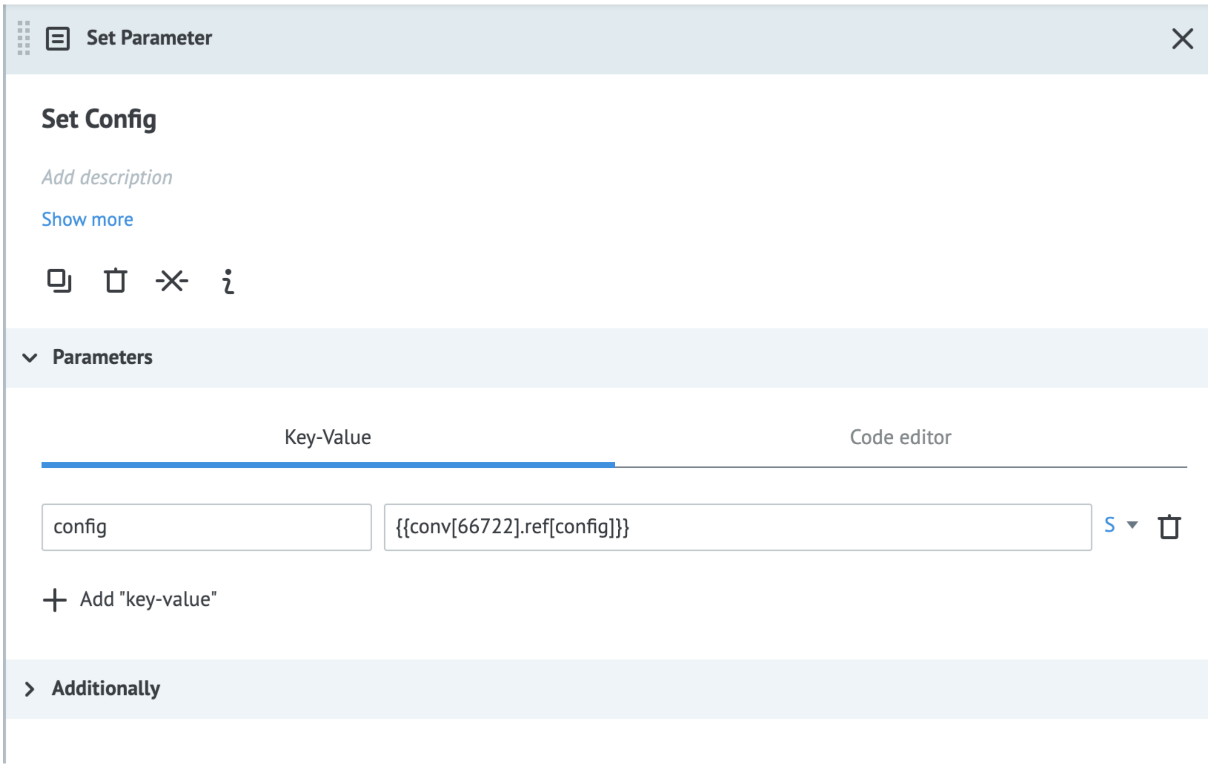


Fig. 3 - “Search Movie and Format” Process

This process contains the logic of searching a movie in the database, setting the query and formatting the response after the Wikipedia search is done. The first node gets the configuration object from the state diagram and sets it in a variable, in order to use the OMDB api key later.



Set Parameter

Set Config

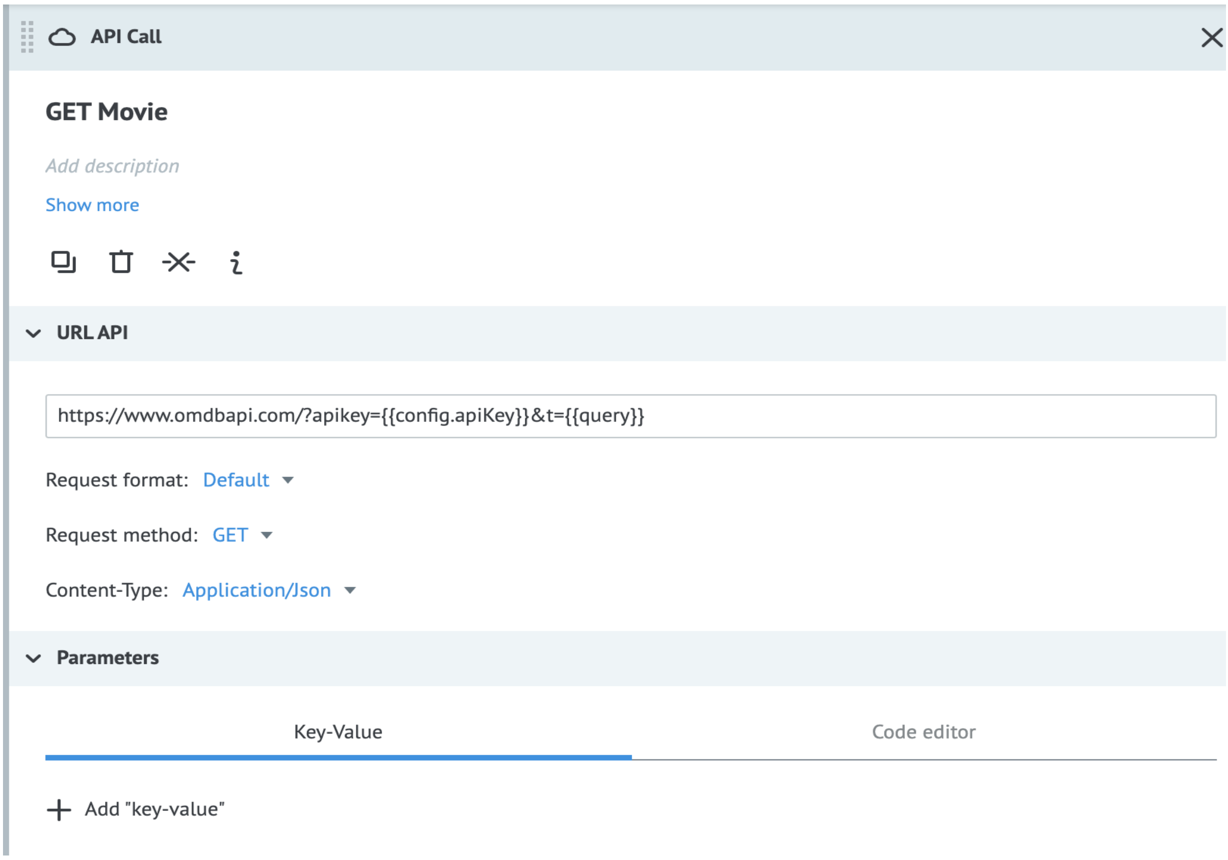
Add description

config

{{conv[66722].ref[config]}}

Fig. 4 - Set Parameter Node

After replacing the spaces in the search query with “%20”, we make an API Call to OMDB, sending it and using the api key from the config.



API Call

GET Movie

Add description

Show more

URL API

https://www.omdbapi.com/?apikey={{config.apiKey}}&t={{query}}

Request format: Default

Request method: GET

Content-Type: Application/Json

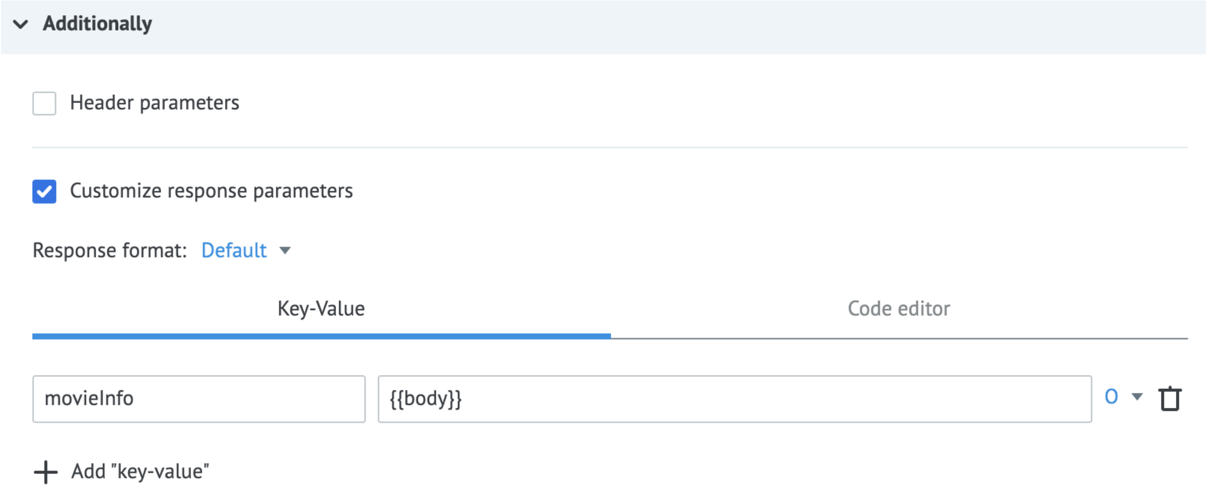
Parameters

Key-Value | Code editor

+ Add "key-value"

Fig. 5 - API Call

The call returns the results in a body object, which we rename to “movieInfo” to avoid confusion.



Additionally

□ Header parameters

☑ Customize response parameters

Response format: Default

Key-Value | Code editor

movieInfo

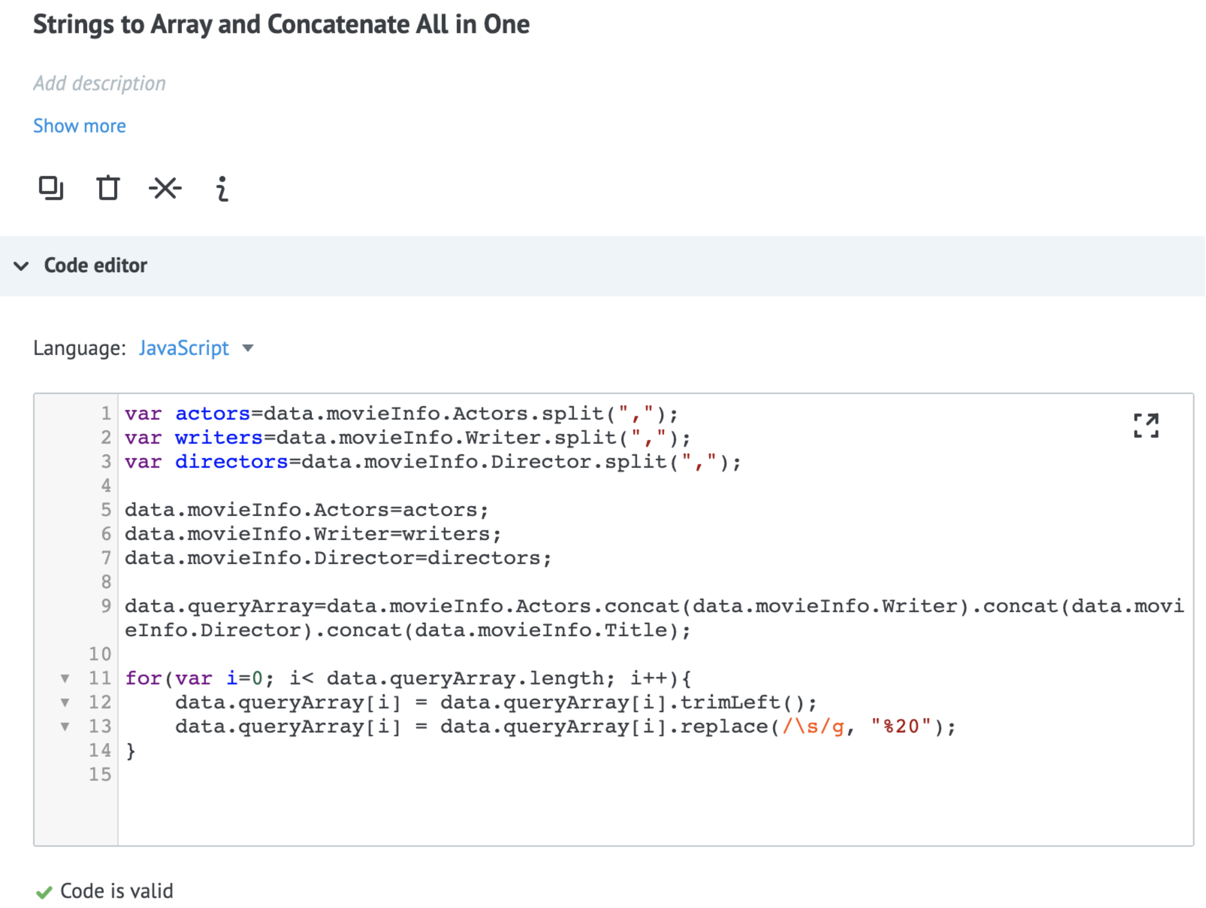
{{body}}

0

+ Add "key-value"

Fig. 6 - Custom response parameters

If no movie is found, the API sets “Response” as “false”. We use a condition node to check that and reply with a message to the main process. If the movie has been found, we take all data we want to search on Wikipedia and concatenate it into one single array, using a code node.



Strings to Array and Concatenate All in One

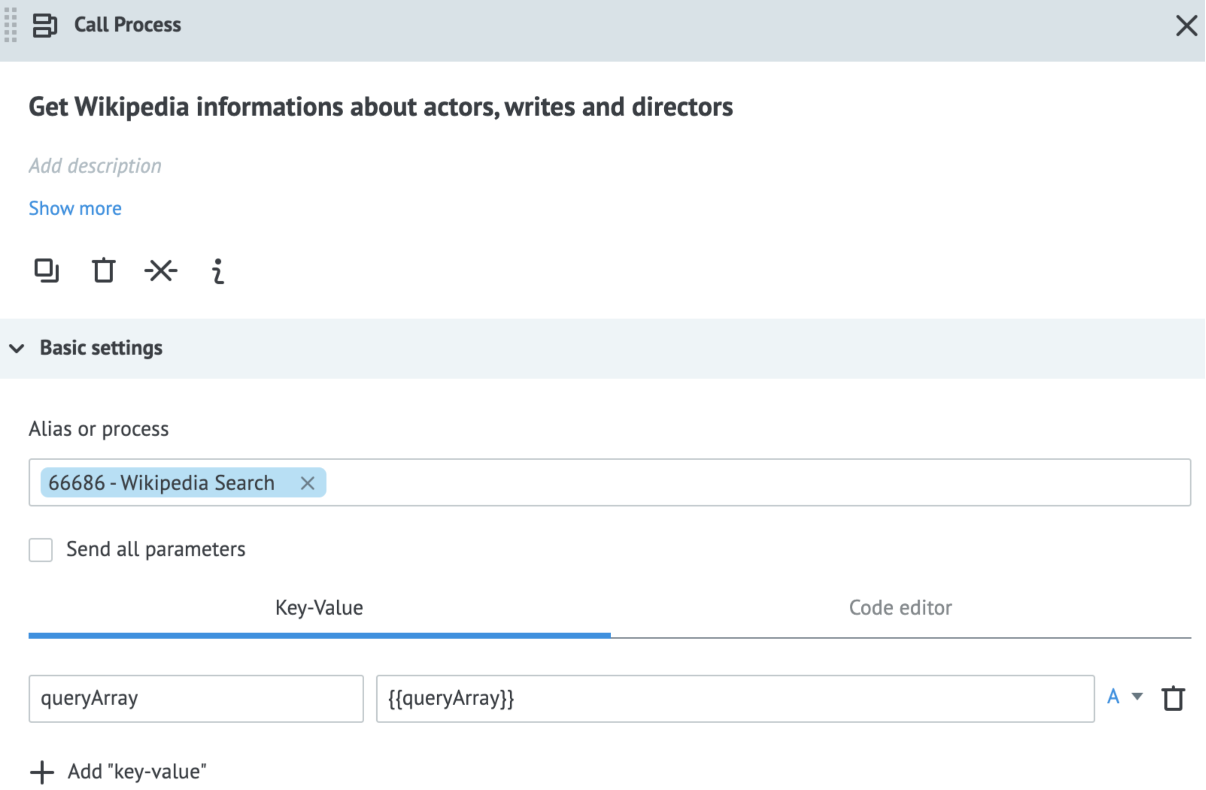
Language: JavaScript

```javascript
var actors=data.movieInfo.Actors.split(",");
var writers=data.movieInfo.Writer.split(",");
var directors=data.movieInfo.Director.split(",");

data.movieInfo.Actors=actors;
data.movieInfo.Writer=writers;
data.movieInfo.Director=directors;

data.queryArray=data.movieInfo.Actors.concat(data.movieInfo.Writer).concat(data.movieInfo.Director).concat(data.movieInfo.Title);

for(var i=0; i< data.queryArray.length; i++){
    data.queryArray[i] = data.queryArray[i].trimLeft();
    data.queryArray[i] = data.queryArray[i].replace(/\s/g, "%20");
}

```

✓ Code is valid

Fig. 7 - Code to merge arrays

We then send that array to the “Wikipedia Search” Process.



Call Process

Get Wikipedia informations about actors, writes and directors

Add description

Show more

Basic settings

Alias or process

66686 - Wikipedia Search

□ Send all parameters

Key-Value | Code editor

queryArray	{{queryArray}}

Add "key-value"

Fig. 8. - Sending the created array to another process

After getting the reply from the process, we format the response array to be user-friendly and send it to Main.

B. “Wikipedia Search”

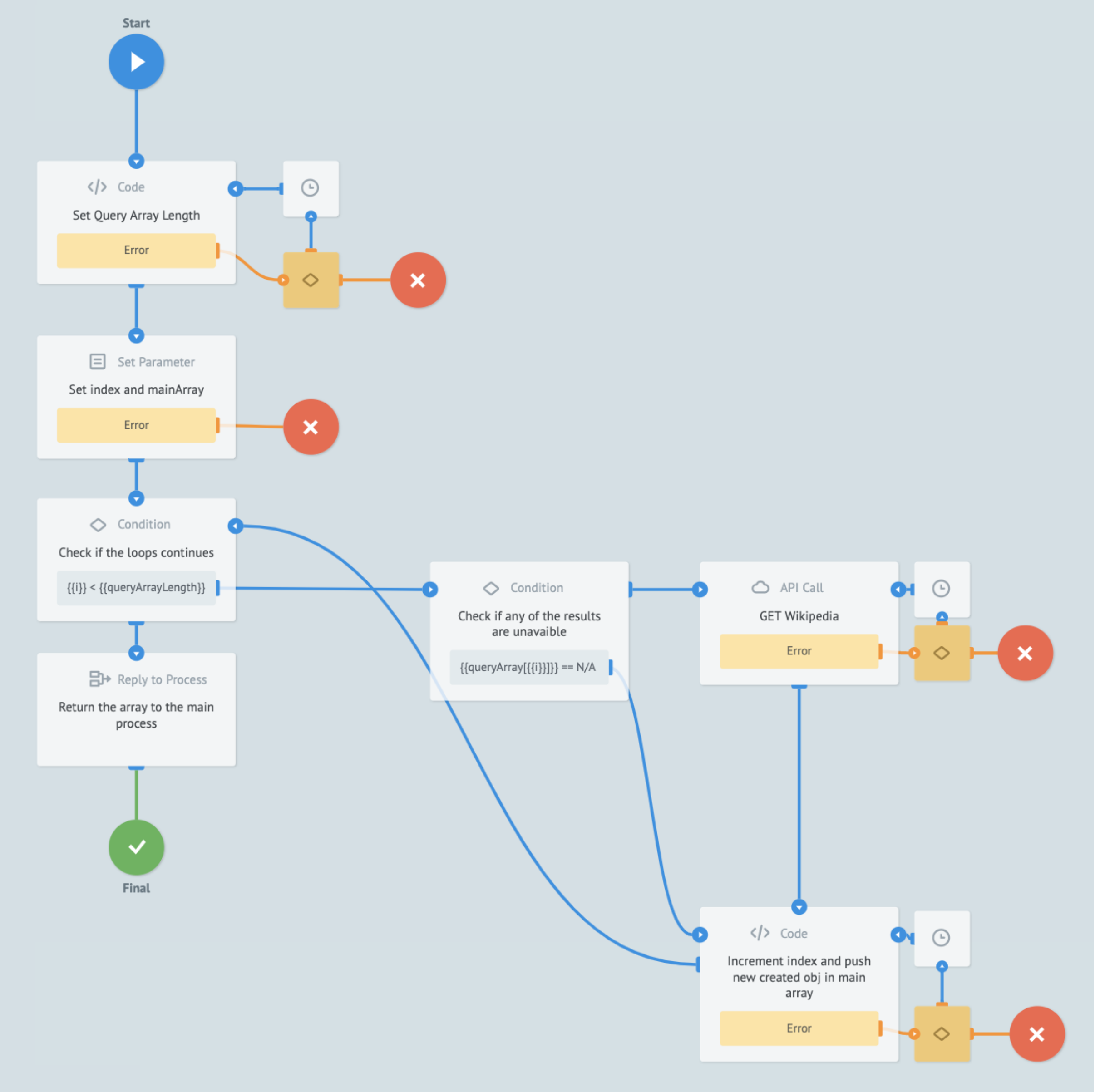
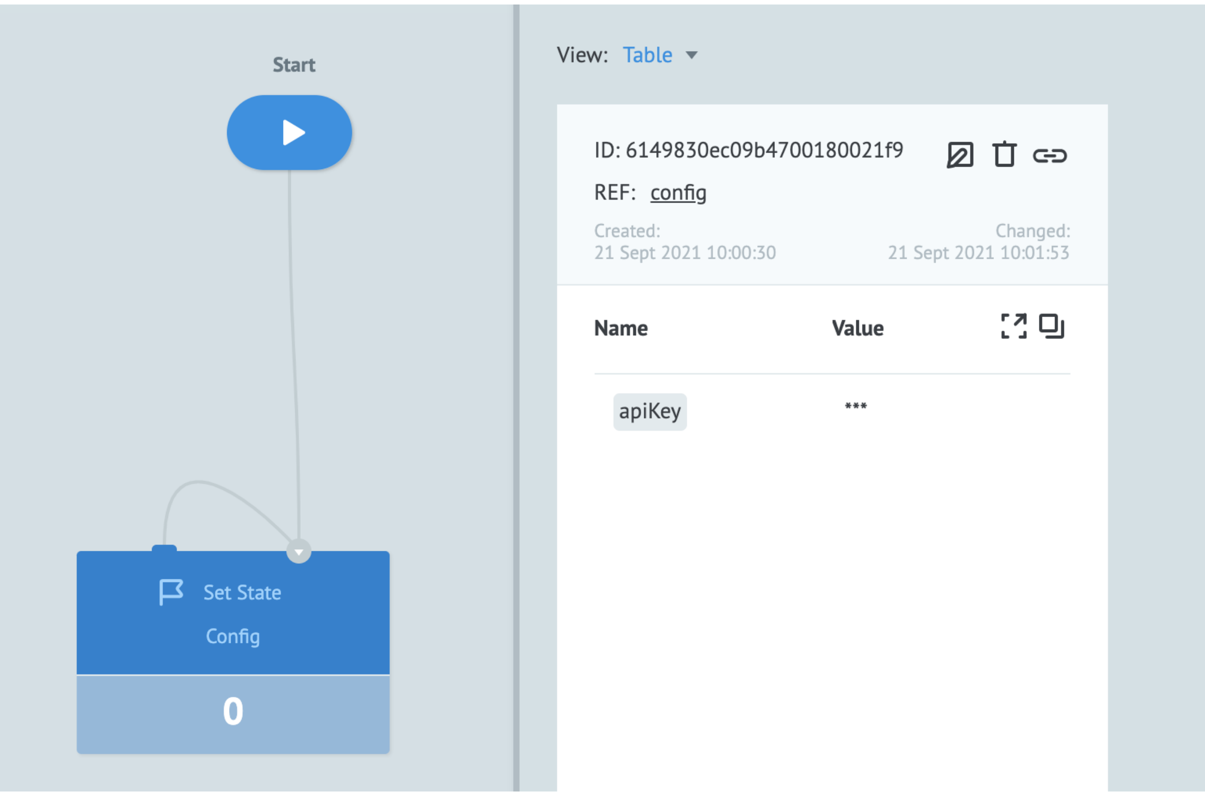

Start
Code
Set Query Array Length
Error
Set Parameter
Set index and mainArray
Error
Condition
Check if the loops continues
{{i}} < {{queryArrayLength}}
Reply to Process
Return the array to the main process
Final
Condition
Check if any of the results are unavaible
{{queryArray[{{i}}]}} == N/A
API Call
GET Wikipedia
Error
Code
Increment index and push new created obj in main array
Error

Fig. 9. - “Wikipedia Search” Process

In order to parse the query array and search each item using Wikipedia’s API, we need to create a loop, similar to a “while”. First, we use a code node to set the array’s length and a “Set Parameter” to set the index and an empty array to store the results. Then, while the index is lower than the array length, and if the item exists, we make an API Call and search it, save the results and increment the index. In the end we reply by sending the results array.

IV. Config



View: Table

ID: 6149830ec09b4700180021f9

REF: config

Created: 21 Sept 2021 10:00:30

Changed: 21 Sept 2021 10:01:53

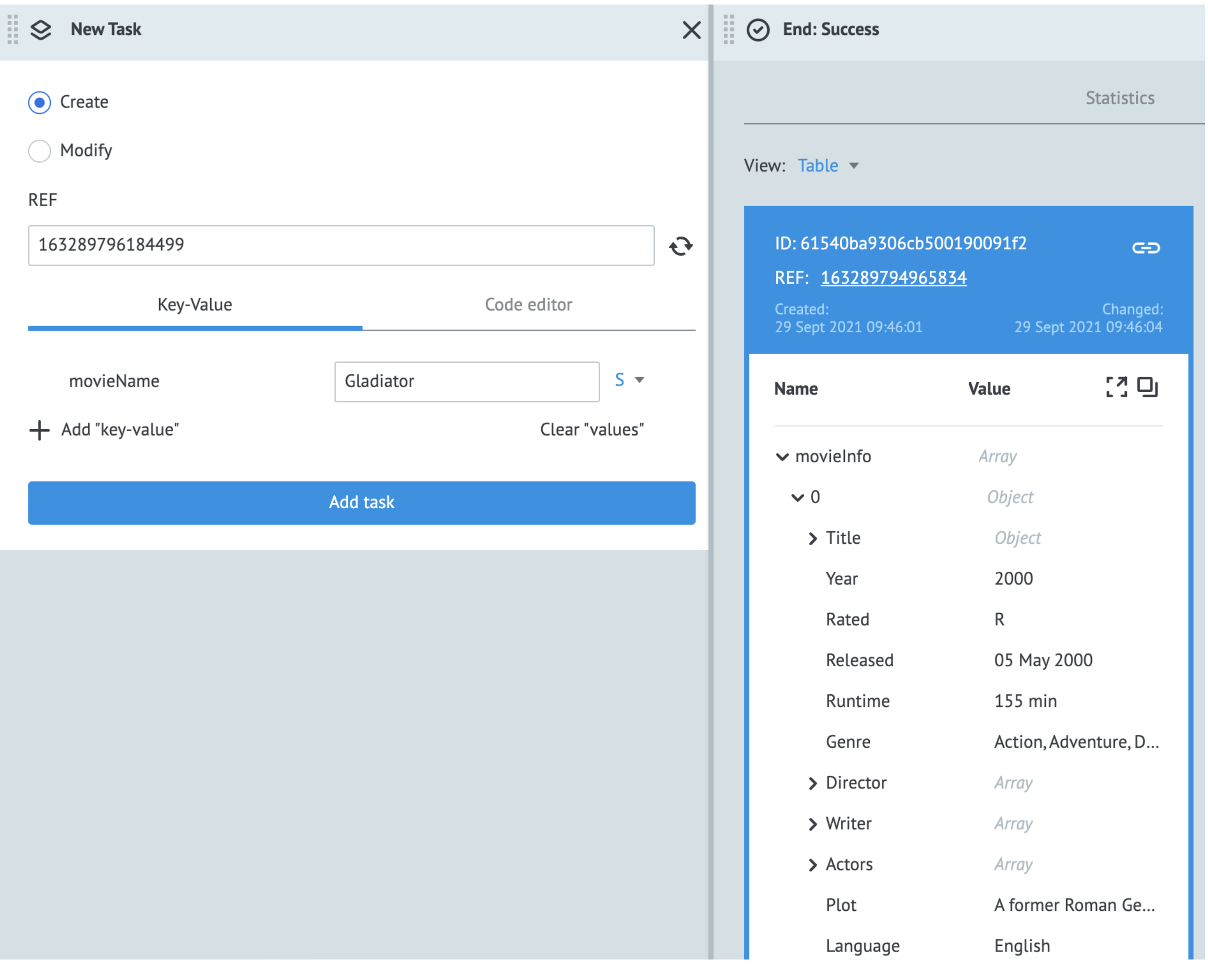
Name	Value
apiKey	***

Fig. 10 - Configuration Task

The config is a state diagram, which means it stores tasks, each with its own reference and parameters. The task named “config” contains an api key used for OMDB, being cleared so that it can only be read by MPO.

V. Use Cases

a. Existing Movie



● Create

○ Modify

REF: 163289796184499

movieName: Gladiator S

View: Table

ID: 61540ba9306cb500190091f2

REF: 163289794965834

Created: 29 Sept 2021 09:46:01

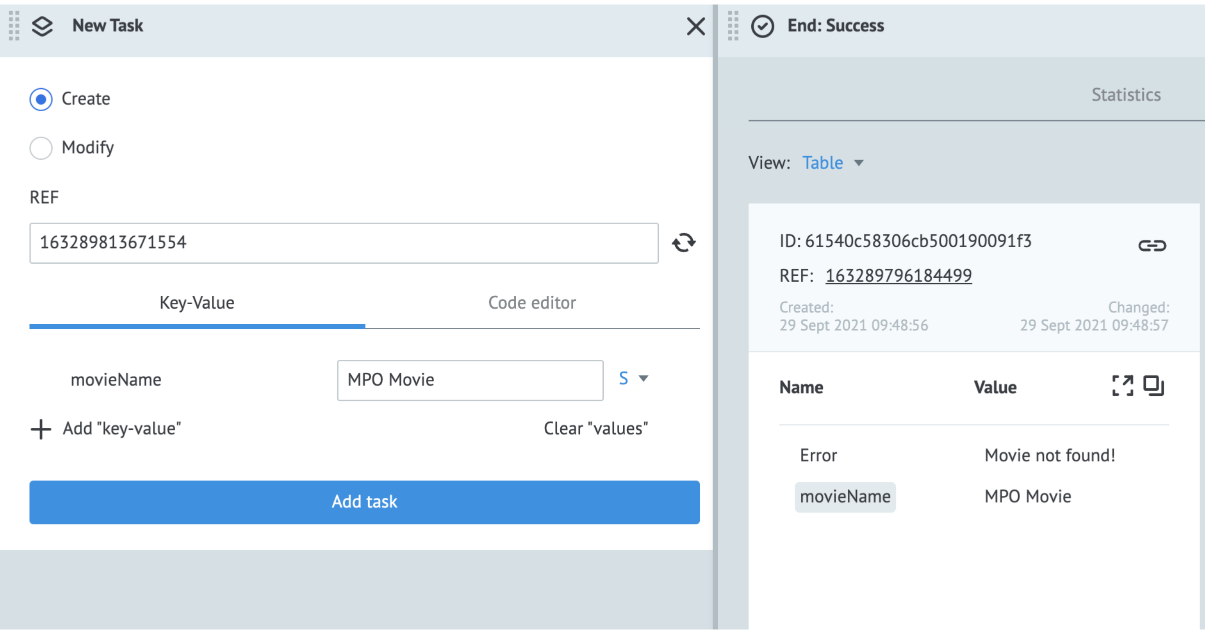
Changed: 29 Sept 2021 09:46:04

Name	Value
movieInfo	Array
0	Object
Title	Object
Year	2000
Rated	R
Released	05 May 2000
Runtime	155 min
Genre	Action, Adventure, D...
Director	Array
Writer	Array
Actors	Array
Plot	A former Roman Ge...
Language	English

Fig. 11 - Query and Task with Existing Movie

When searching for “Gladiator”, we get an array with all the information about the movie and the links found on Wikipedia.

b. Inexistent Movie



● Create
○ Modify

REF: 163289813671554

movieName: MPO Movie (S)

View: Table

ID: 61540c58306cb500190091f3
REF: 163289796184499
Created: 29 Sept 2021 09:48:56
Changed: 29 Sept 2021 09:48:57

Name	Value
Error	Movie not found!
movieName	MPO Movie

Fig. 12 - Query and Task for a Non-Existing Movie

When searching for “MPO Movie", we get notified that the movie was not found.

VI. Conclusions

Low-Code platforms offer an overall more pleasant experience, accessible to both programmers and inexperienced users. MPO is a versatile platform that can run code if needed, but offers nodes as a better alternative.

This example is able to search for a movie and get more information about it than what is normally available. By using API Calls, processes and a little bit of code, all visual, the most important part becomes the logic behind connecting nodes. Thus, creating an application implies more discussion and thinking than usual, and less time actually writing and refactoring.