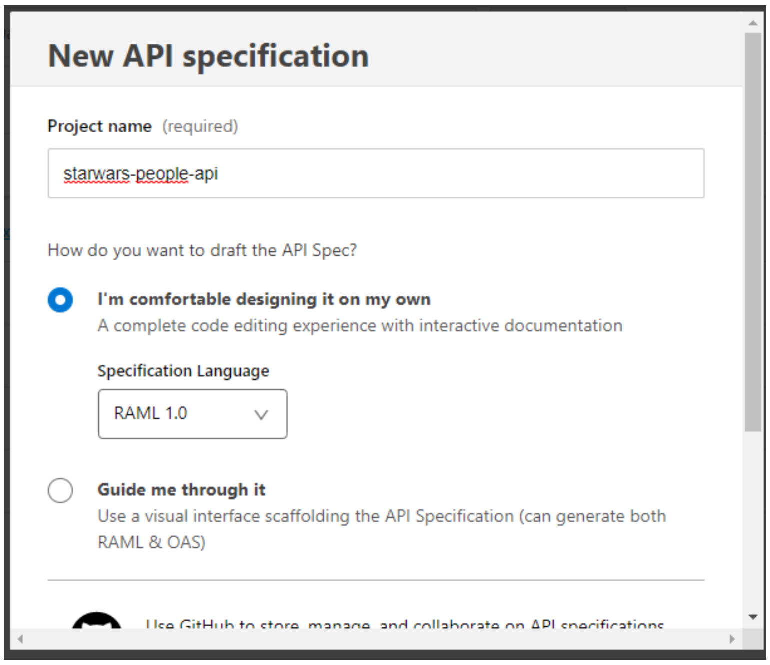
Practice documentation

API Specification

It starts with the creation of the specification. A descriptive name is assigned.



New API specification

Project name (required)

starwars-people-api

How do you want to draft the API Spec?

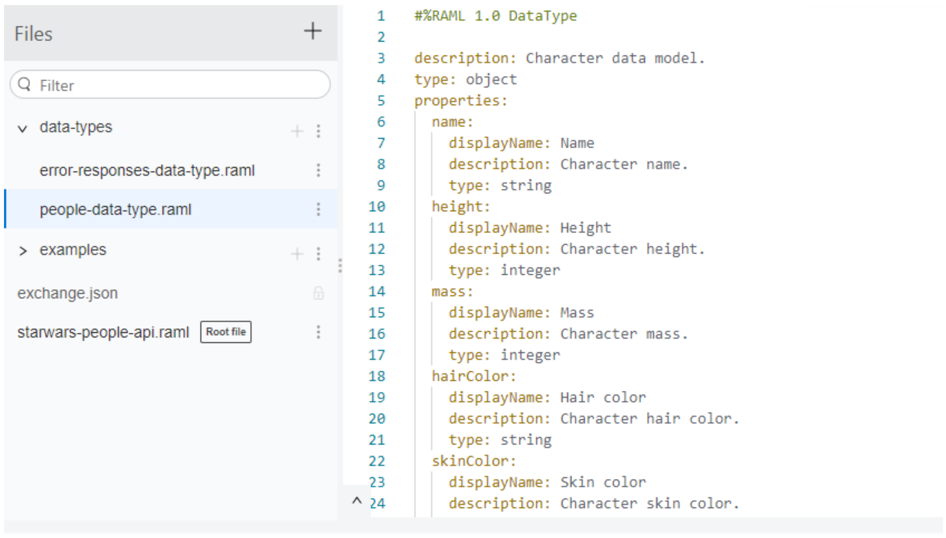
(●) I'm comfortable designing it on my own
A complete code editing experience with interactive documentation

Specification Language: RAML 1.0

( ) Guide me through it
Use a visual interface scaffolding the API Specification (can generate both RAML & OAS)

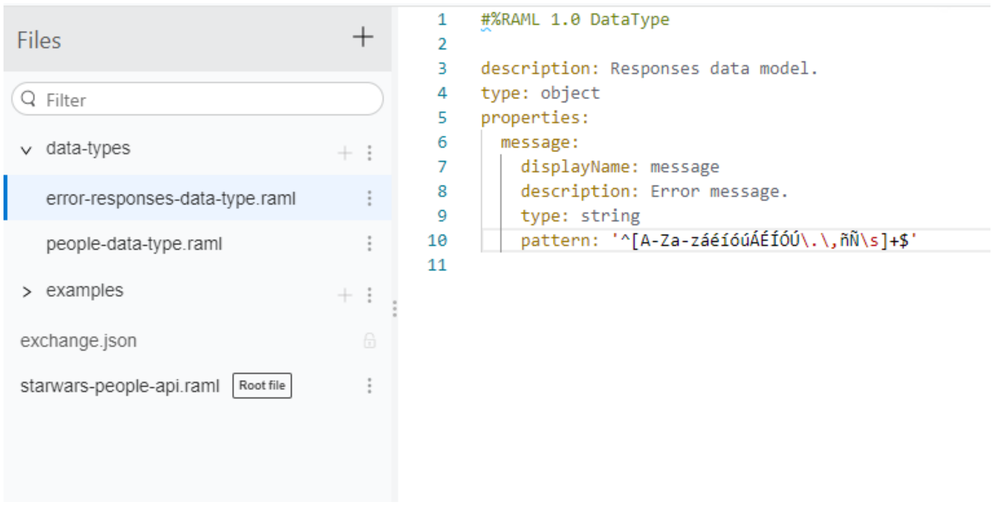
Use GitHub to store, manage, and collaborate on API specifications

A data type file is created. The schema to work with is created, in this case the characteristics of the characters are documented.



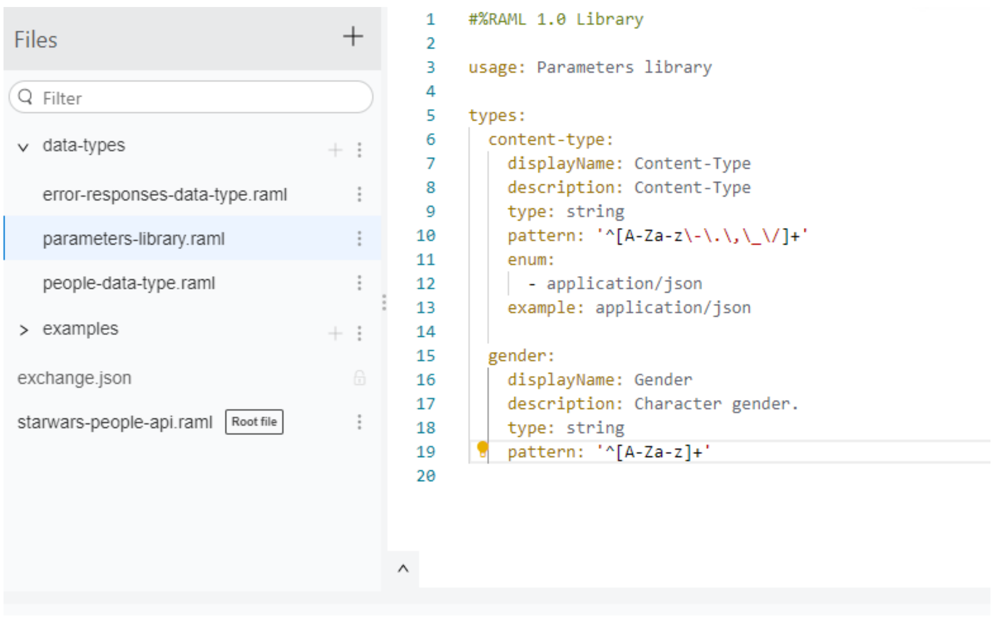
```
#%RAML 1.0 DataType

description: Character data model.
type: object
properties:
  name:
    displayName: Name
    description: Character name.
    type: string
  height:
    displayName: Height
    description: Character height.
    type: integer
  mass:
    displayName: Mass
    description: Character mass.
    type: integer
  hairColor:
    displayName: Hair color
    description: Character hair color.
    type: string
  skinColor:
    displayName: Skin color
    description: Character skin color.
```

The data type with which errors are described is also documented.



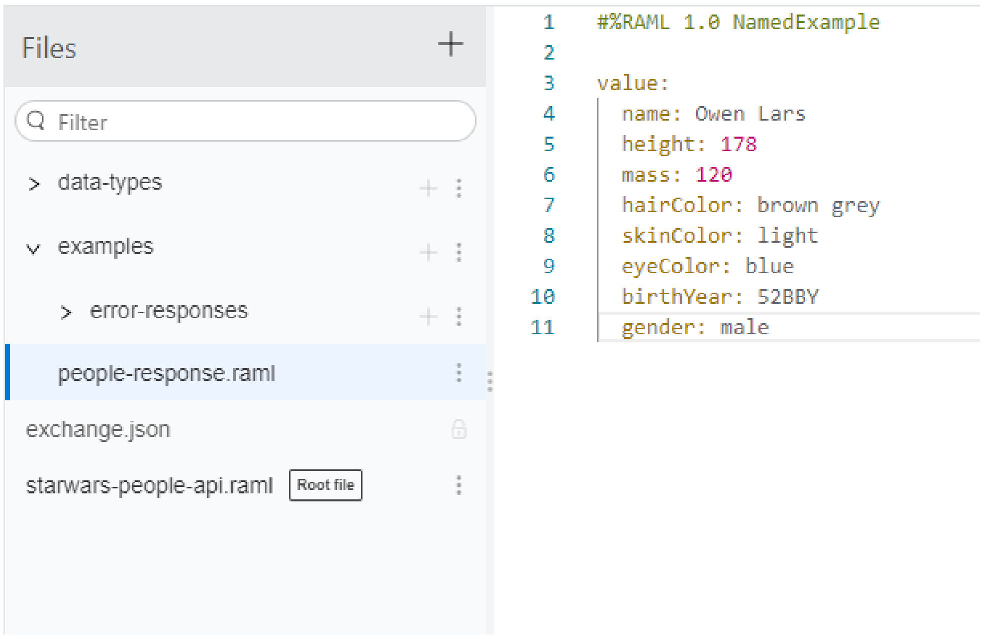
```
#%RAML 1.0 DataType

description: Responses data model.
type: object
properties:
  message:
    displayName: message
    description: Error message.
    type: string
    pattern: '^[A-Za-záéíóúÁÉÍÓÚ\.\,ñÑ\s]+$'

```

I also included a library to describe the different parameters that will be found in the API.

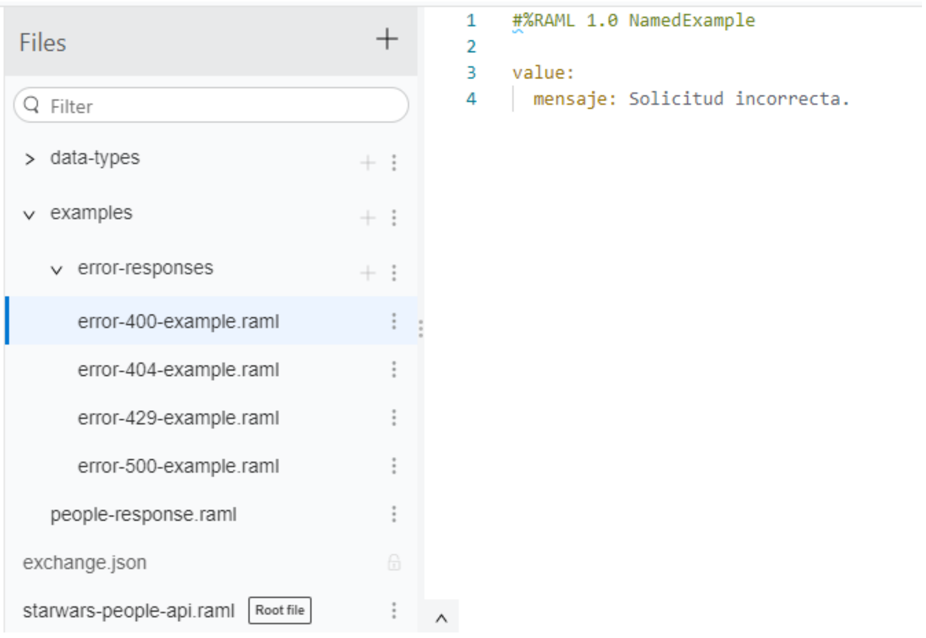


```
#%RAML 1.0 Library

usage: Parameters library

types:
  content-type:
    displayName: Content-Type
    description: Content-Type
    type: string
    pattern: '^[A-Za-z\-\.\,\_\/]+'
    enum:
      - application/json
    example: application/json

  gender:
    displayName: Gender
    description: Character gender.
    type: string
    pattern: '^[A-Za-z]+'

```

Once the data types have been described, example files are created to show the different responses that can be encountered

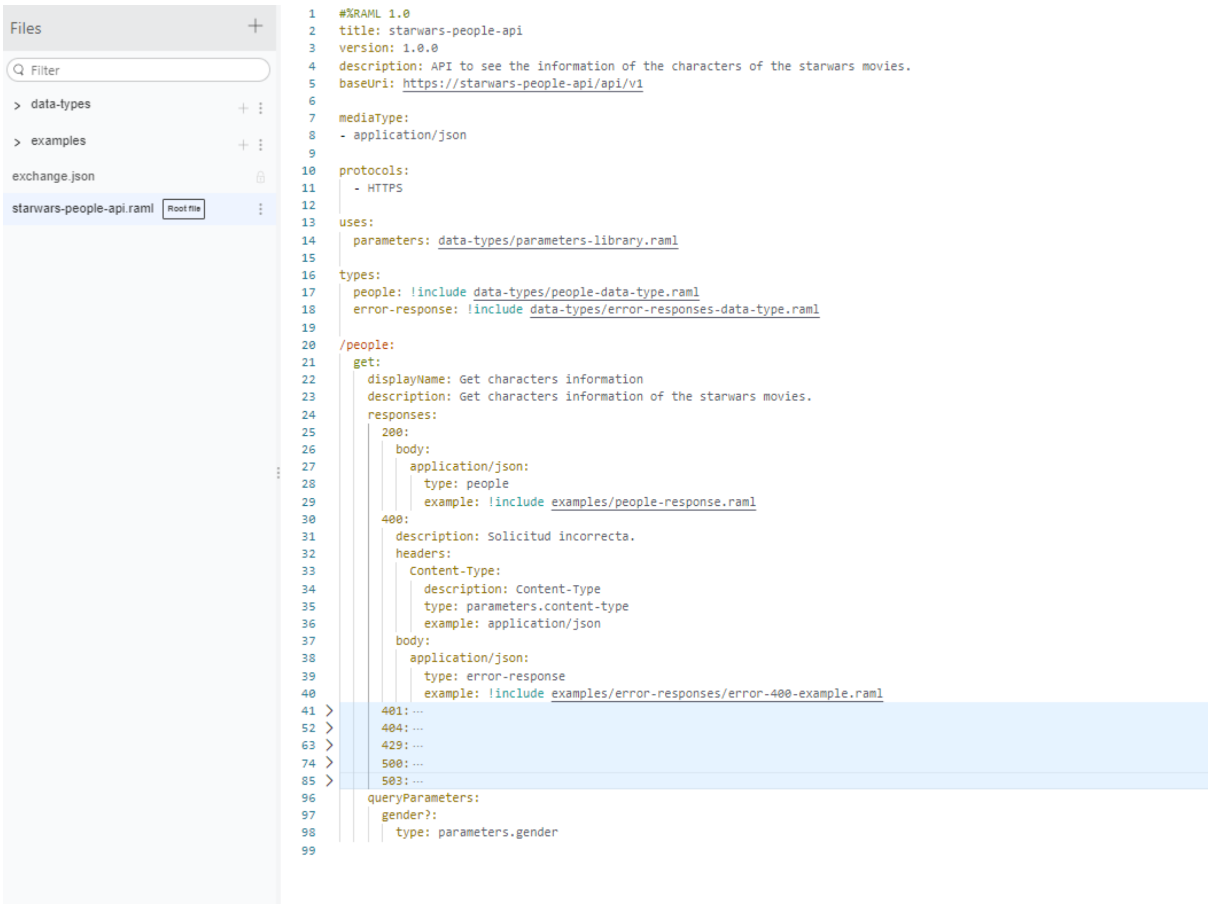


```
#%RAML 1.0 NamedExample

value:
  name: Owen Lars
  height: 178
  mass: 120
  hairColor: brown grey
  skinColor: light
  eyeColor: blue
  birthYear: 52BBY
  gender: male
```



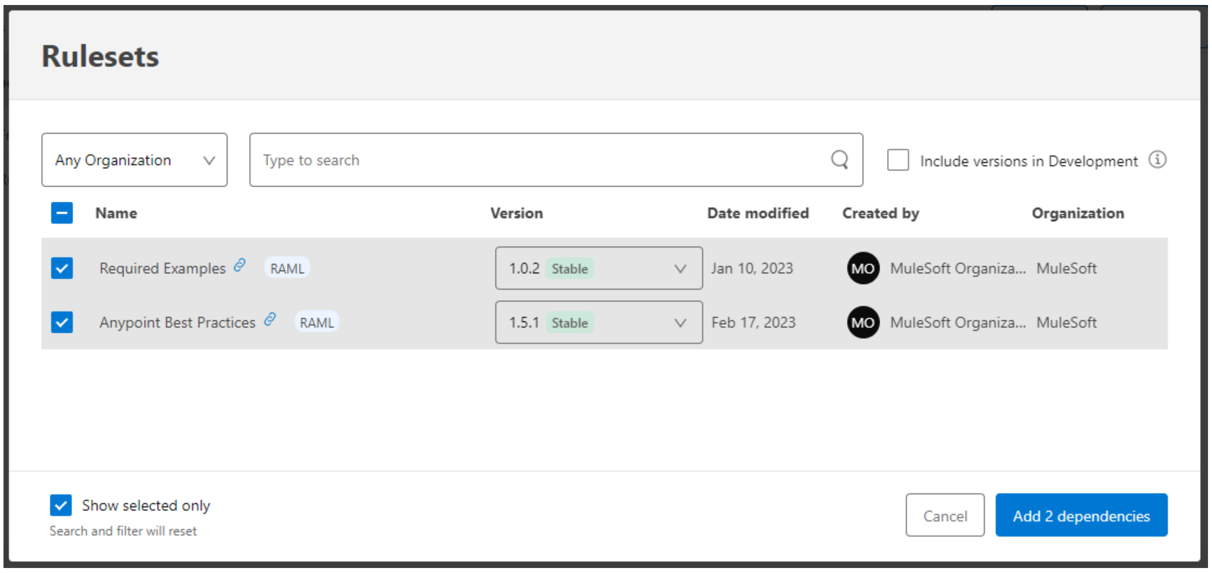
```
#%RAML 1.0 NamedExample

value:
  mensaje: Solicitud incorrecta.
```

The main file is created, a brief description of the API functionality, a version, the different protocols, the different data type files are mapped.
The endpoints and operations to be managed are defined, a description is given and the different responses are mapped with their respective examples.
Likewise, the query/uri params that the operation has are documented.



```
#%RAML 1.0
title: starwars-people-api
version: 1.0.0
description: API to see the information of the characters of the starwars movies.
baseUri: https://starwars-people-api/api/v1

mediaType:
- application/json

protocols:
  - HTTPS

uses:
  parameters: data-types/parameters-library.raml

types:
  people: !include data-types/people-data-type.raml
  error-response: !include data-types/error-responses-data-type.raml

/people:
  get:
    displayName: Get characters information
    description: Get characters information of the starwars movies.
    responses:
      200:
        body:
          application/json:
            type: people
            example: !include examples/people-response.raml
      400:
        description: Solicitud incorrecta.
        headers:
          Content-Type:
            description: Content-Type
            type: parameters.content-type
            example: application/json
        body:
          application/json:
            type: error-response
            example: !include examples/error-responses/error-400-example.raml
      401: ···
      404: ···
      429: ···
      500: ···
      503: ···
    queryParameters:
      gender?:
        type: parameters.gender
```

Once we have the specification, we make an evaluation of the code with the rulesets offered by mulesoft.


Rulesets

	Name		Version	Date modified	Created by	Organization
✓	Required Examples	RAML	1.0.2 Stable	Jan 10, 2023	MuleSoft Organiza...	MuleSoft
✓	Anypoint Best Practices	RAML	1.5.1 Stable	Feb 17, 2023	MuleSoft Organiza...	MuleSoft

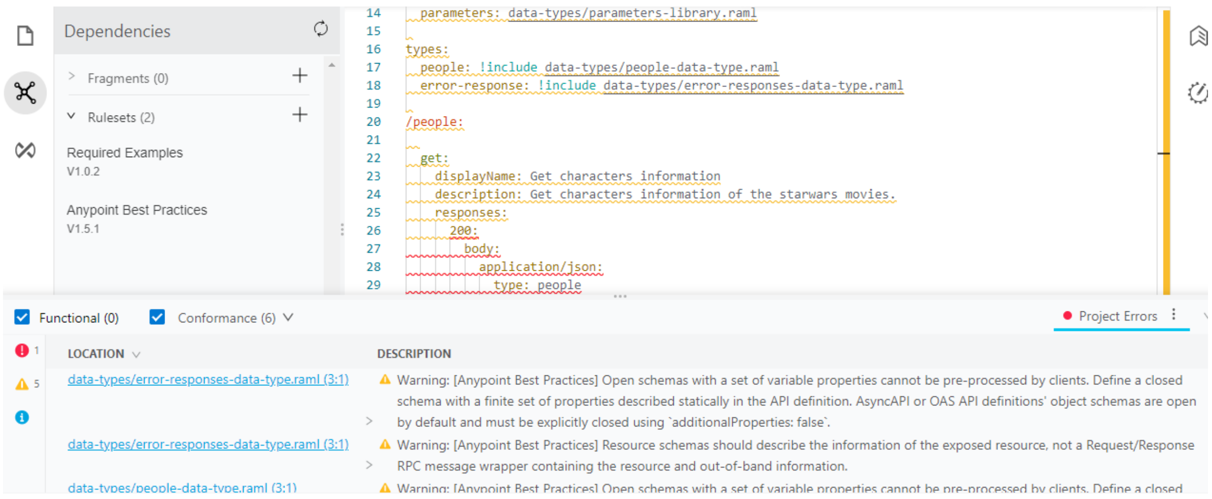
✓ Show selected only
Search and filter will reset

If the evaluation detects issues or warnings, they are resolved.

In this case, it was detected that in the data types it was necessary to define closed schemas, to solve it, the following was added in each schema:

```
additionalProperties: false
```

And a missing header was added in the successful response.



Dependencies

Fragments (0)

Rulesets (2)

Required Examples
V1.0.2

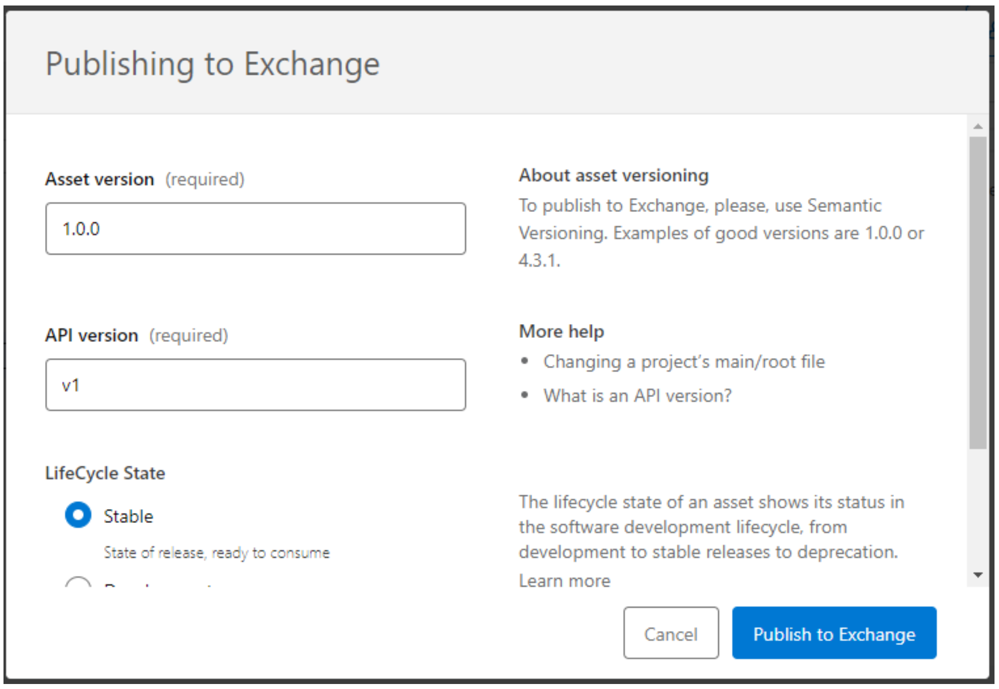
Anypoint Best Practices
V1.5.1

```
  parameters: data-types/parameters-library.raml

types:
  people: !include data-types/people-data-type.raml
  error-response: !include data-types/error-responses-data-type.raml

/people:

  get:
    displayName: Get characters information
    description: Get characters information of the starwars movies.
    responses:
      200:
        body:
          application/json:
            type: people
```

Functional (0) Conformance (6) Project Errors

LOCATION	DESCRIPTION
data-types/error-responses-data-type.raml (3:1)	Warning: [Anypoint Best Practices] Open schemas with a set of variable properties cannot be pre-processed by clients. Define a closed schema with a finite set of properties described statically in the API definition. AsyncAPI or OAS API definitions' object schemas are open by default and must be explicitly closed using `additionalProperties: false`.
data-types/error-responses-data-type.raml (3:1)	Warning: [Anypoint Best Practices] Resource schemas should describe the information of the exposed resource, not a Request/Response RPC message wrapper containing the resource and out-of-band information.
data-types/people-data-type.raml (3:1)	Warning: [Anypoint Best Practices] Open schemas with a set of variable properties cannot be pre-processed by clients. Define a closed

Once everything is correct, the specification is published in exchange to make it public and to be able to consume it.



Publishing to Exchange

Asset version (required): 1.0.0

API version (required): v1

LifeCycle State

[✓ Stable] State of release, ready to consume

About asset versioning

To publish to Exchange, please, use Semantic Versioning. Examples of good versions are 1.0.0 or 4.3.1.

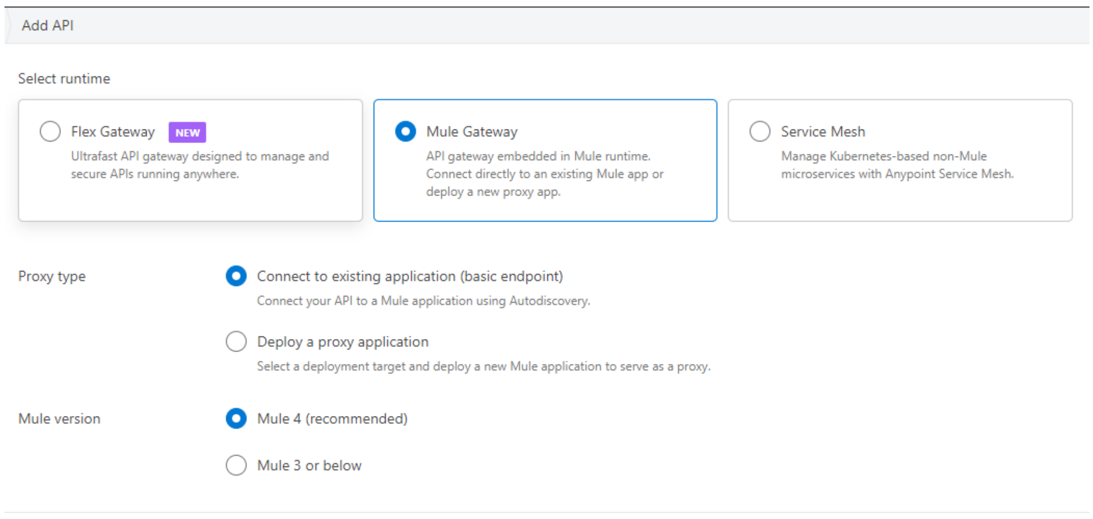
More help

- Changing a project's main/root file
- What is an API version?

The lifecycle state of an asset shows its status in the software development lifecycle, from development to stable releases to deprecation.
Learn more

API Manager

To allow the application to connect to the API Manager you need to create an API ID.



Add API

Select runtime

( ) Flex Gateway NEW
Ultrafast API gateway designed to manage and secure APIs running anywhere.

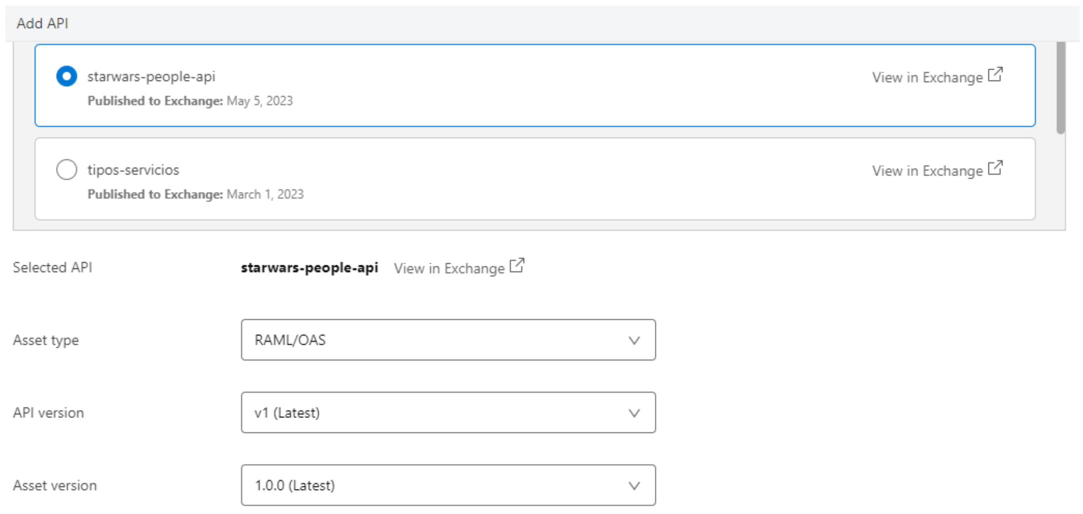
[✓ Mule Gateway]
API gateway embedded in Mule runtime. Connect directly to an existing Mule app or deploy a new proxy app.

( ) Service Mesh
Manage Kubernetes-based non-Mule microservices with Anypoint Service Mesh.

Proxy type: [✓ Connect to existing application (basic endpoint)]
Connect your API to a Mule application using Autodiscovery.
( ) Deploy a proxy application
Select a deployment target and deploy a new Mule application to serve as a proxy.

Mule version: [✓ Mule 4 (recommended)] ( ) Mule 3 or below

The specification is selected from the exchange.

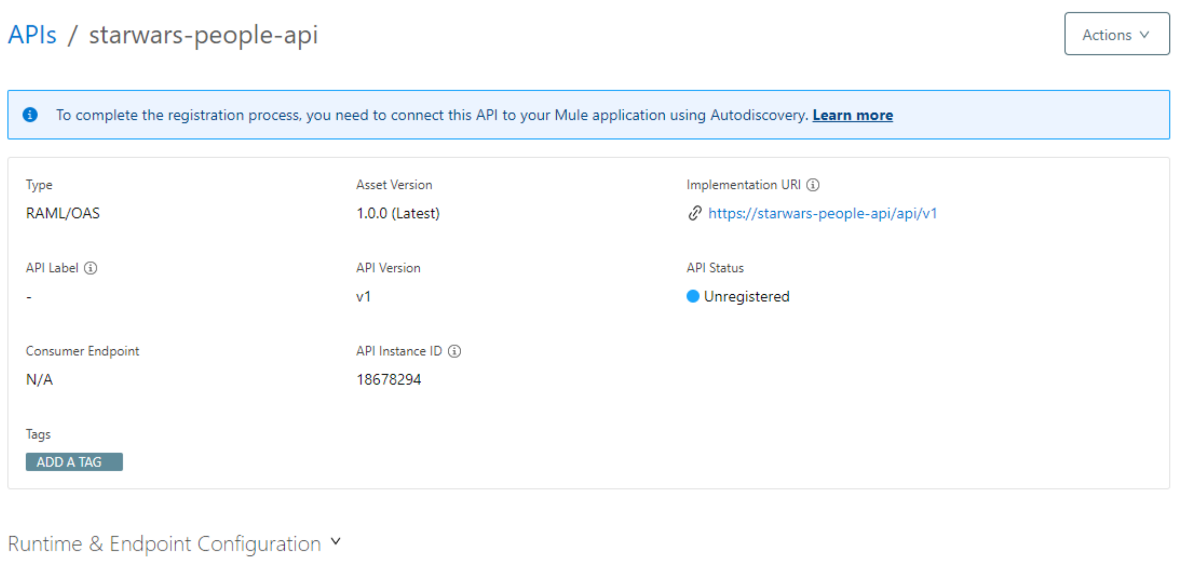


Add API

[✓ starwars-people-api] — **Published to Exchange:** May 5, 2023 — View in Exchange

( ) tipos-servicios — **Published to Exchange:** March 1, 2023 — View in Exchange

Selected API	**starwars-people-api** View in Exchange
Asset type	RAML/OAS
API version	v1 (Latest)
Asset version	1.0.0 (Latest)


APIs / starwars-people-api

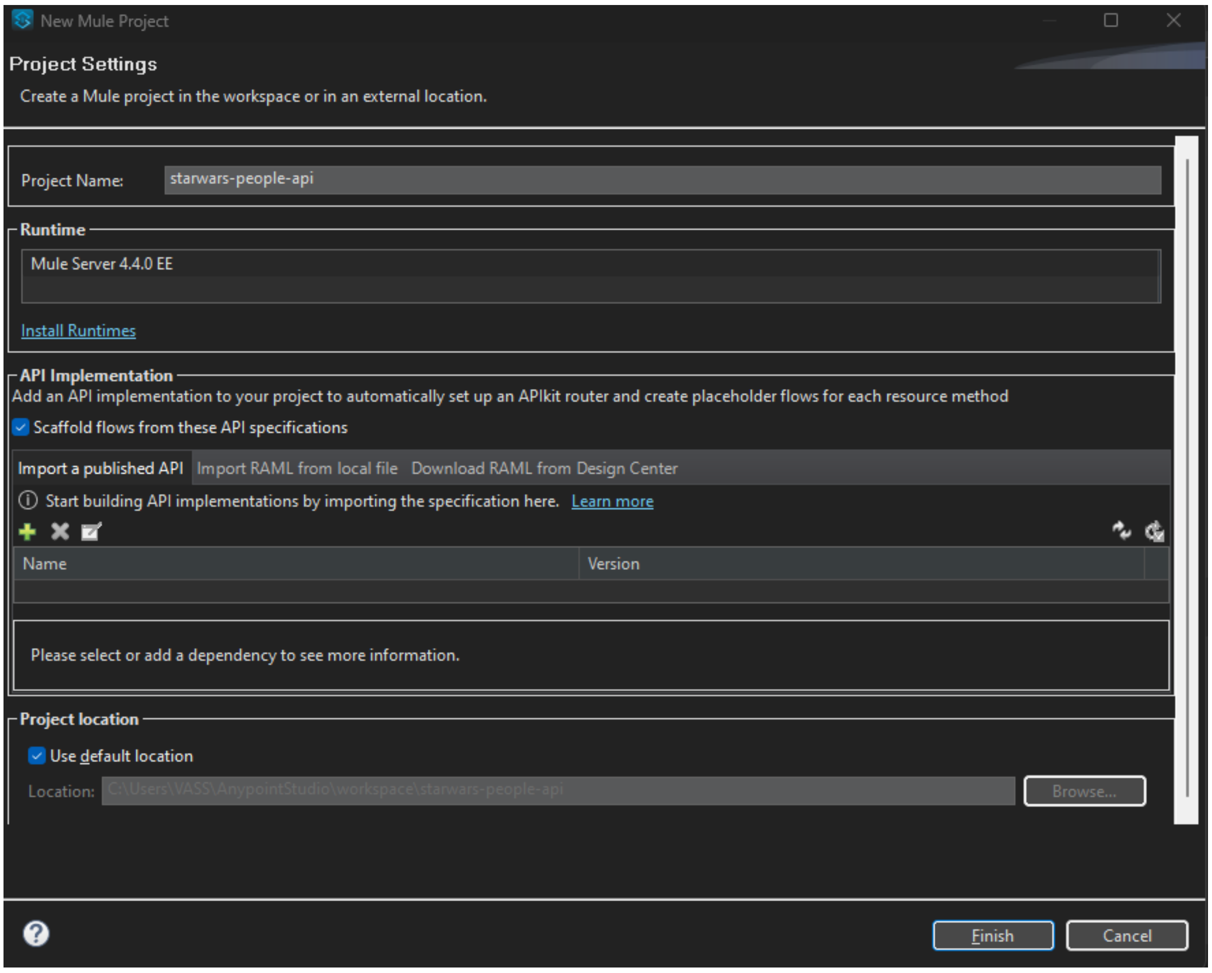
To complete the registration process, you need to connect this API to your Mule application using Autodiscovery. Learn more

Type: RAML/OAS
Asset Version: 1.0.0 (Latest)
Implementation URI: https://starwars-people-api/api/v1
API Label: -
API Version: v1
API Status: Unregistered
Consumer Endpoint: N/A
API Instance ID: 18678294
Tags

Anypoint Studio

At anypoint studio we start with the creation of the implementation.

Create a new project and give it a name.


New Mule Project

Project Settings

Create a Mule project in the workspace or in an external location.

Project Name: starwars-people-api

Runtime: Mule Server 4.4.0 EE

API Implementation

Add an API implementation to your project to automatically set up an APIkit router and create placeholder flows for each resource method

✓ Scaffold flows from these API specifications

Start building API implementations by importing the specification here. Learn more

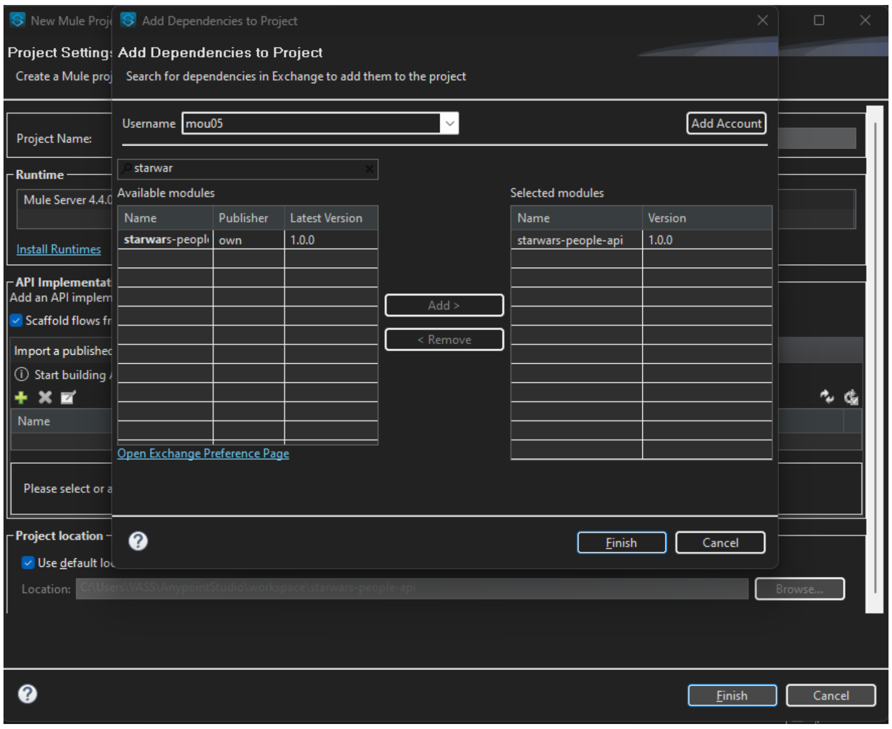
Please select or add a dependency to see more information.

Project location

✓ Use default location

Location: C:\Users\VASS\AnypointStudio\workspace\starwars-people-api

Then, we add the specification from exchange.



Add Dependencies to Project

Search for dependencies in Exchange to add them to the project

Username: mou05

starwar

Available modules

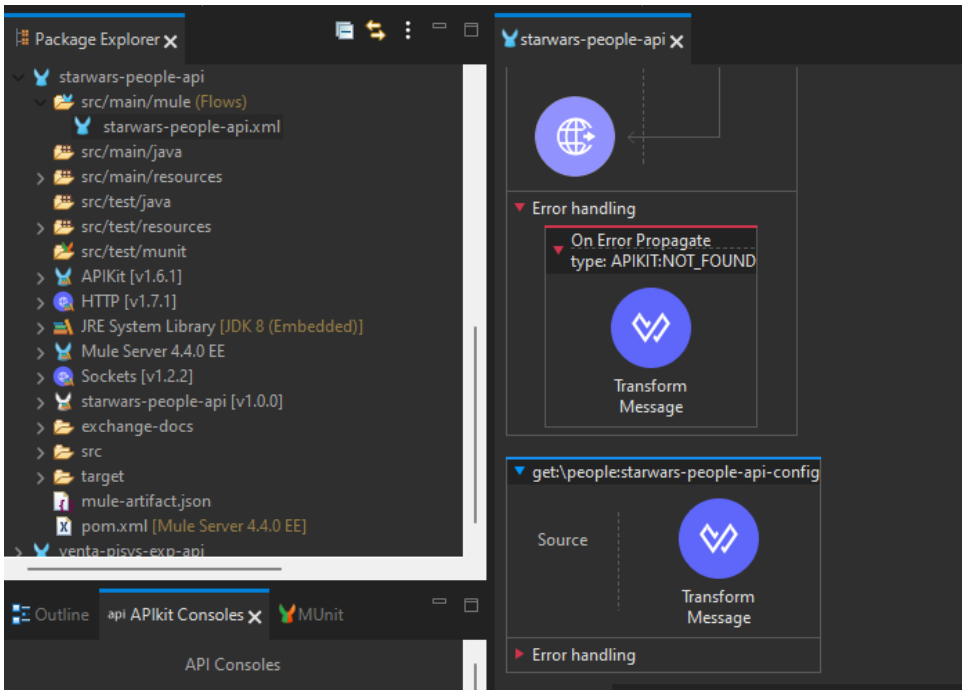
Name	Publisher	Latest Version
starwars-peopl	own	1.0.0

Selected modules

Name	Version
starwars-people-api	1.0.0

Open Exchange Preference Page

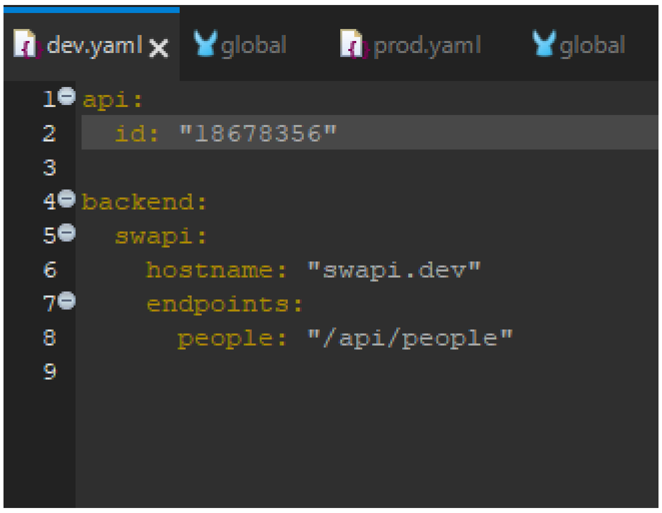
Once the specification is downloaded, we will see the flows created automatically.



```
Package Explorer
starwars-people-api
  src/main/mule (Flows)
    starwars-people-api.xml
  src/main/java
  src/main/resources
  src/test/java
  src/test/resources
  src/test/munit
  APIKit [v1.6.1]
  HTTP [v1.7.1]
  JRE System Library [JDK 8 (Embedded)]
  Mule Server 4.4.0 EE
  Sockets [v1.2.2]
  starwars-people-api [v1.0.0]
  exchange-docs
  src
  target
  mule-artifact.json
  pom.xml [Mule Server 4.4.0 EE]
venta-pisys-exp-api
```

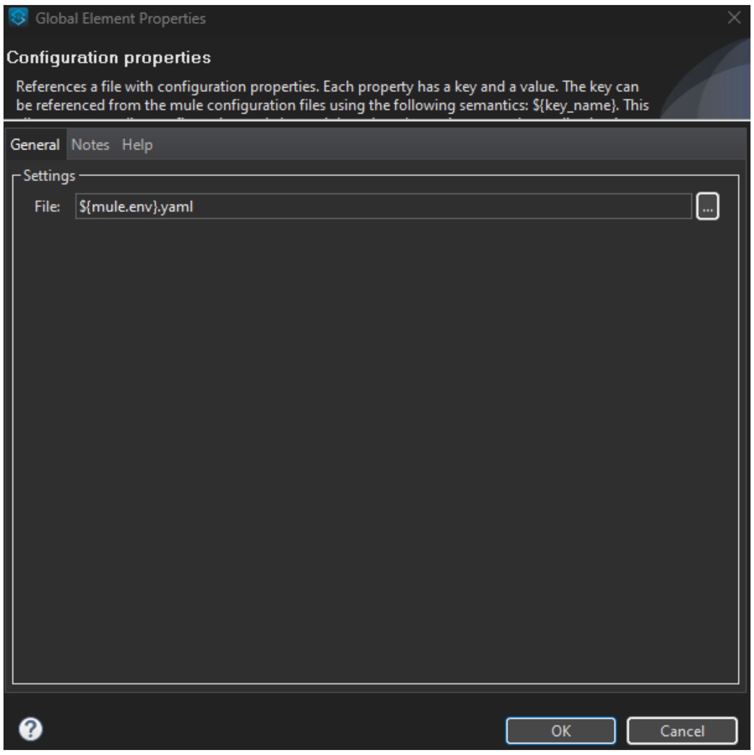
Then, configuration files are created for each development environment, this helps to better manage credentials and consumed services.

Also, in the development environment, the API ID created earlier is added.



```yaml
api:
  id: "18678356"

backend:
  swapi:
    hostname: "swapi.dev"
    endpoints:
      people: "/api/people"

```

A properties configuration component is created to link the configuration files.

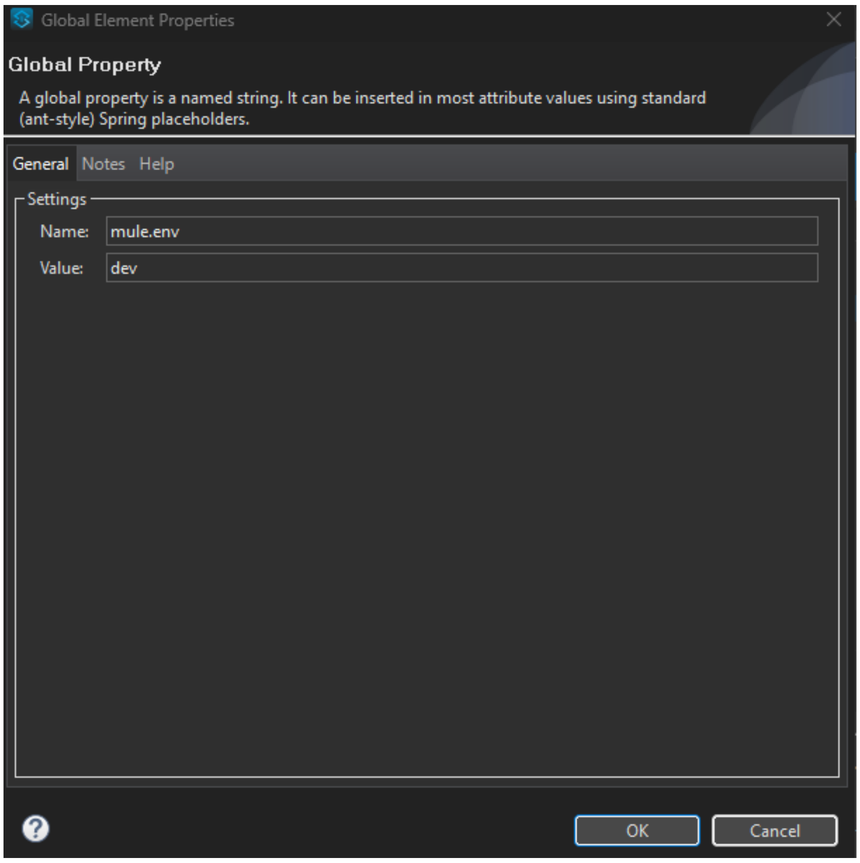


Configuration properties

References a file with configuration properties. Each property has a key and a value. The key can be referenced from the mule configuration files using the following semantics: ${key_name}. This

File: ${mule.env}.yaml

Also, a global component is created for the configuration file that is used locally.


Global Element Properties

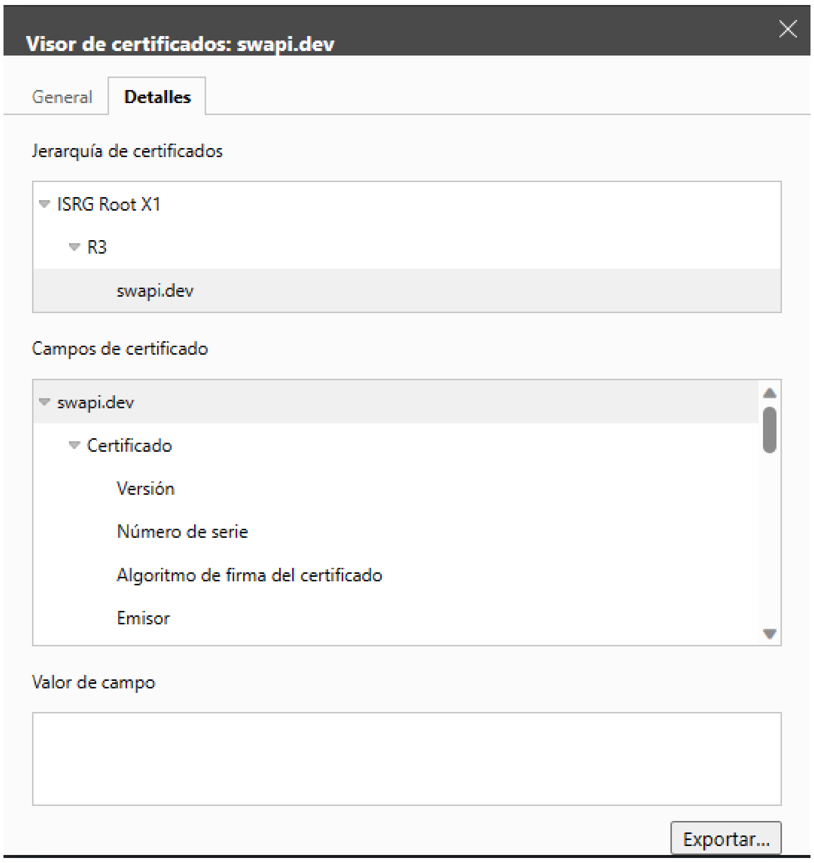
Global Property

A global property is a named string. It can be inserted in most attribute values using standard (ant-style) Spring placeholders.

Name: mule.env

Value: dev

Now, the API used for the exercise (swapi) has a secure protocol (HTTPS) so you have to download its trusted certificate to create a jks file.


Visor de certificados: swapi.dev

Jerarquía de certificados

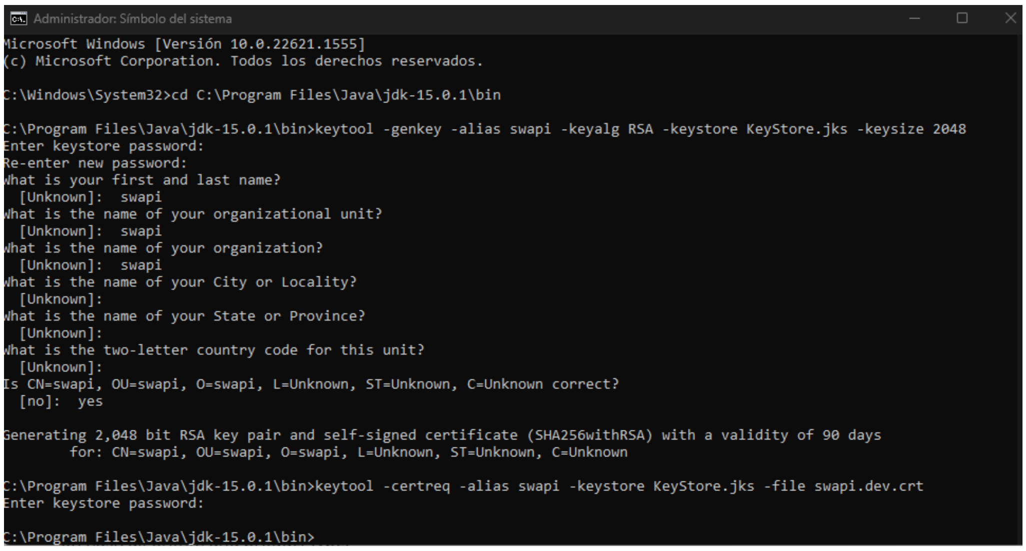
- ISRG Root X1
  - R3
    - swapi.dev

Campos de certificado

- swapi.dev
  - Certificado
    - Versión
    - Número de serie
    - Algoritmo de firma del certificado
    - Emisor

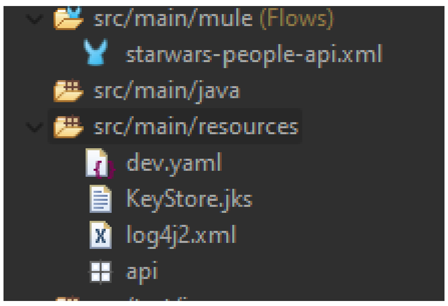
Valor de campo

The jks file is created from the keytool command, it is configured and the trusted certificate of the api to be consumed is stored here.

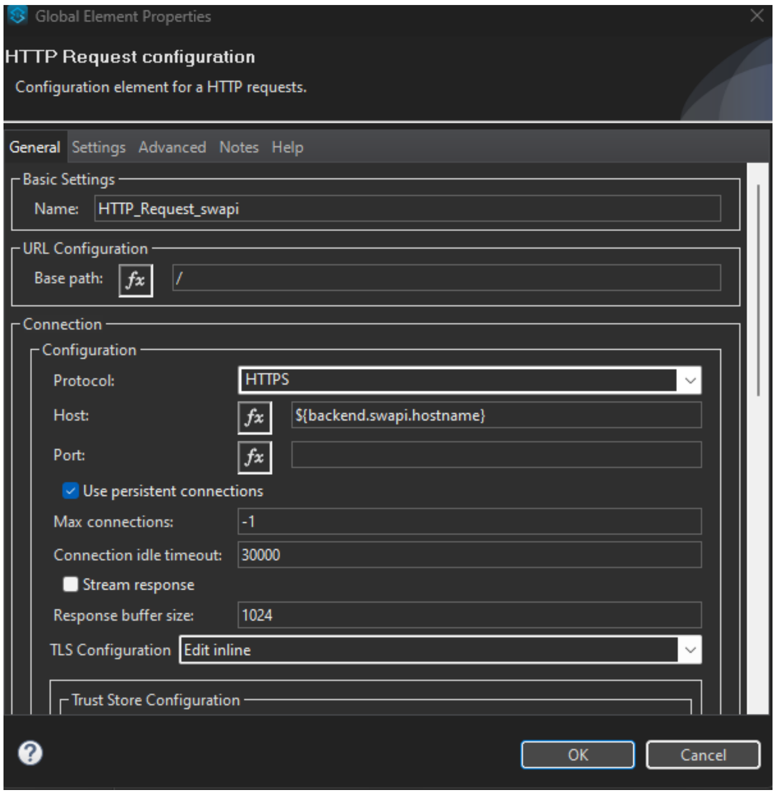


```
Microsoft Windows [Versión 10.0.22621.1555]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Windows\System32>cd C:\Program Files\Java\jdk-15.0.1\bin

C:\Program Files\Java\jdk-15.0.1\bin>keytool -genkey -alias swapi -keyalg RSA -keystore KeyStore.jks -keysize 2048
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  swapi
What is the name of your organizational unit?
  [Unknown]:  swapi
What is the name of your organization?
  [Unknown]:  swapi
What is the name of your City or Locality?
  [Unknown]:
What is the name of your State or Province?
  [Unknown]:
What is the two-letter country code for this unit?
  [Unknown]:
Is CN=swapi, OU=swapi, O=swapi, L=Unknown, ST=Unknown, C=Unknown correct?
  [no]:  yes

Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 90 days
        for: CN=swapi, OU=swapi, O=swapi, L=Unknown, ST=Unknown, C=Unknown

C:\Program Files\Java\jdk-15.0.1\bin>keytool -certreq -alias swapi -keystore KeyStore.jks -file swapi.dev.crt
Enter keystore password:

C:\Program Files\Java\jdk-15.0.1\bin>
```

As the jks file is created, it is saved in the API resources.



Now, an http request component is created, it is configured with the respective parameters.


Global Element Properties

HTTP Request configuration

Configuration element for a HTTP requests.

General Settings Advanced Notes Help

Basic Settings

Name: HTTP_Request_swapi

URL Configuration

Base path: /

Connection

Configuration

Protocol: HTTPS

Host: ${backend.swapi.hostname}

Port:

[✓] Use persistent connections

Max connections: -1

Connection idle timeout: 30000

[ ] Stream response

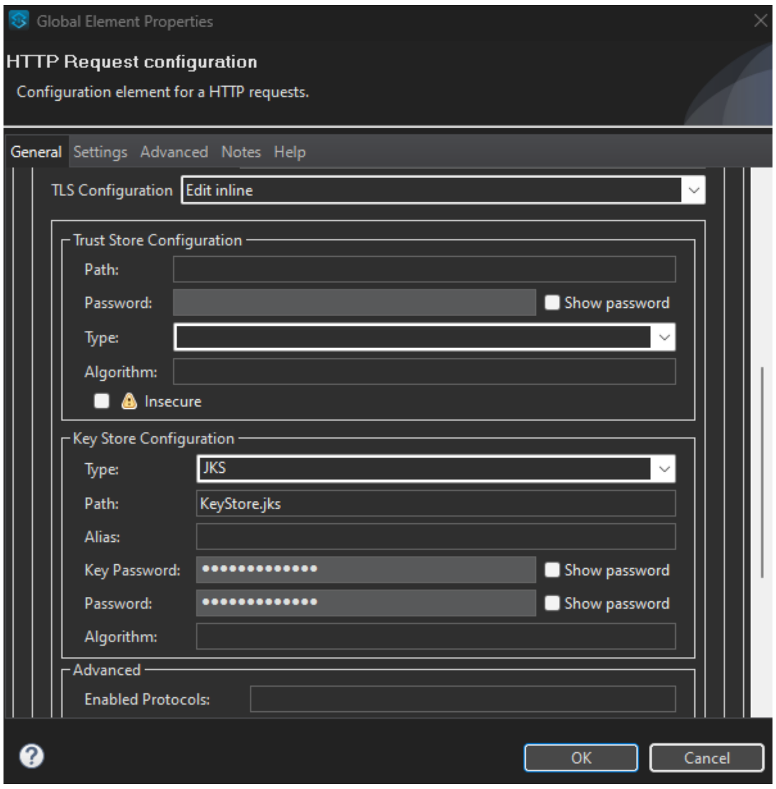
Response buffer size: 1024

TLS Configuration: Edit inline

Trust Store Configuration

OK Cancel

This is also how the keystore is configured, the password is assigned when the jks file is created.


Global Element Properties

HTTP Request configuration

Configuration element for a HTTP requests.

General Settings Advanced Notes Help

TLS Configuration: Edit inline

Trust Store Configuration

Path:

Password: [ ] Show password

Type:

Algorithm:

[ ] Insecure

Key Store Configuration

Type: JKS

Path: KeyStore.jks

Alias:

Key Password: ••••••••••••• [ ] Show password

Password: ••••••••••••• [ ] Show password

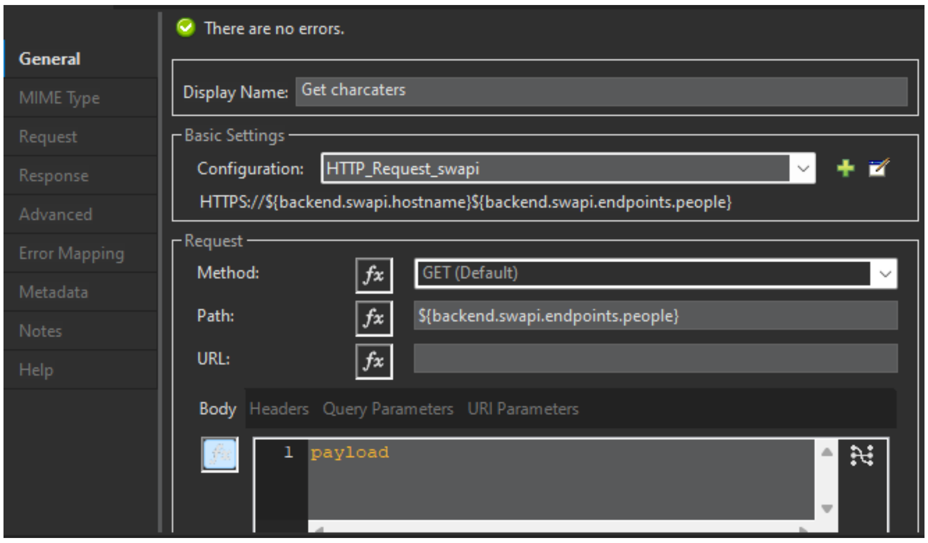
Algorithm:

Advanced

Enabled Protocols:

OK Cancel

Now, we finish configuring the http request, choose the correct method and map the path to consume.


There are no errors.

Display Name: Get charcaters

Basic Settings

Configuration: HTTP_Request_swapi

HTTPS://${backend.swapi.hostname}${backend.swapi.endpoints.people}

Request

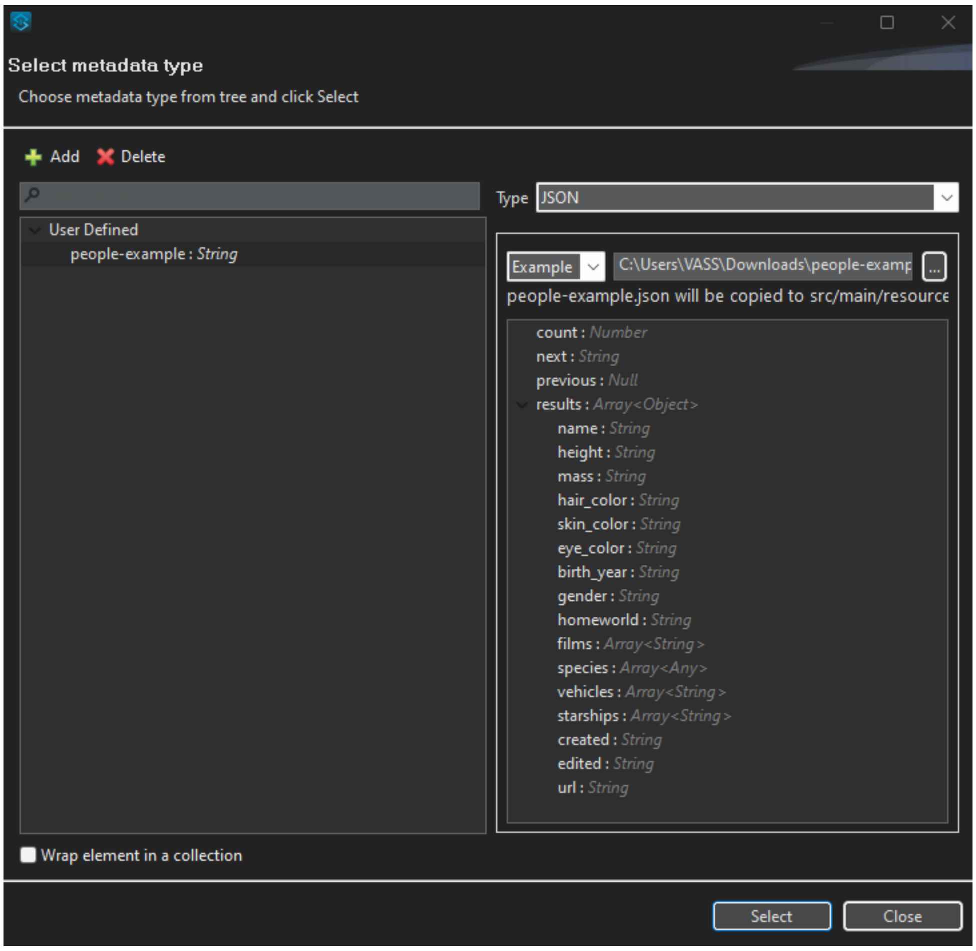
Method: GET (Default)

Path: ${backend.swapi.endpoints.people}

URL:

```
payload
```

to map the data correctly, a json file is created with an example of how the swapi responds and is stored in a metadata type.


Select metadata type

Choose metadata type from tree and click Select

User Defined
people-example : String

Type: JSON

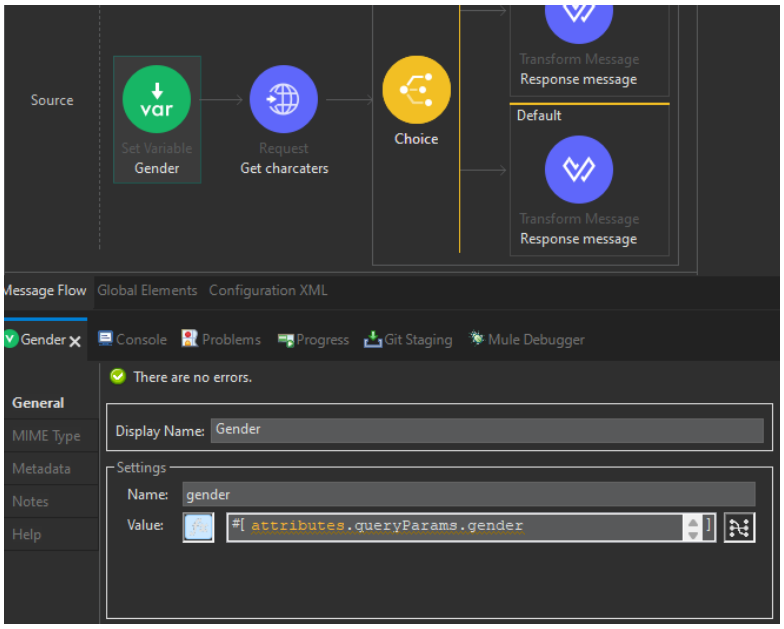
Example: C:\Users\VASS\Downloads\people-examp

people-example.json will be copied to src/main/resource

```
count : Number
next : String
previous : Null
results : Array<Object>
    name : String
    height : String
    mass : String
    hair_color : String
    skin_color : String
    eye_color : String
    birth_year : String
    gender : String
    homeworld : String
    films : Array<String>
    species : Array<Any>
    vehicles : Array<String>
    starships : Array<String>
    created : String
    edited : String
    url : String
```

Wrap element in a collection

As the activity asks to be able to filter through a query param, a variable was added to store the value.



There are no errors.

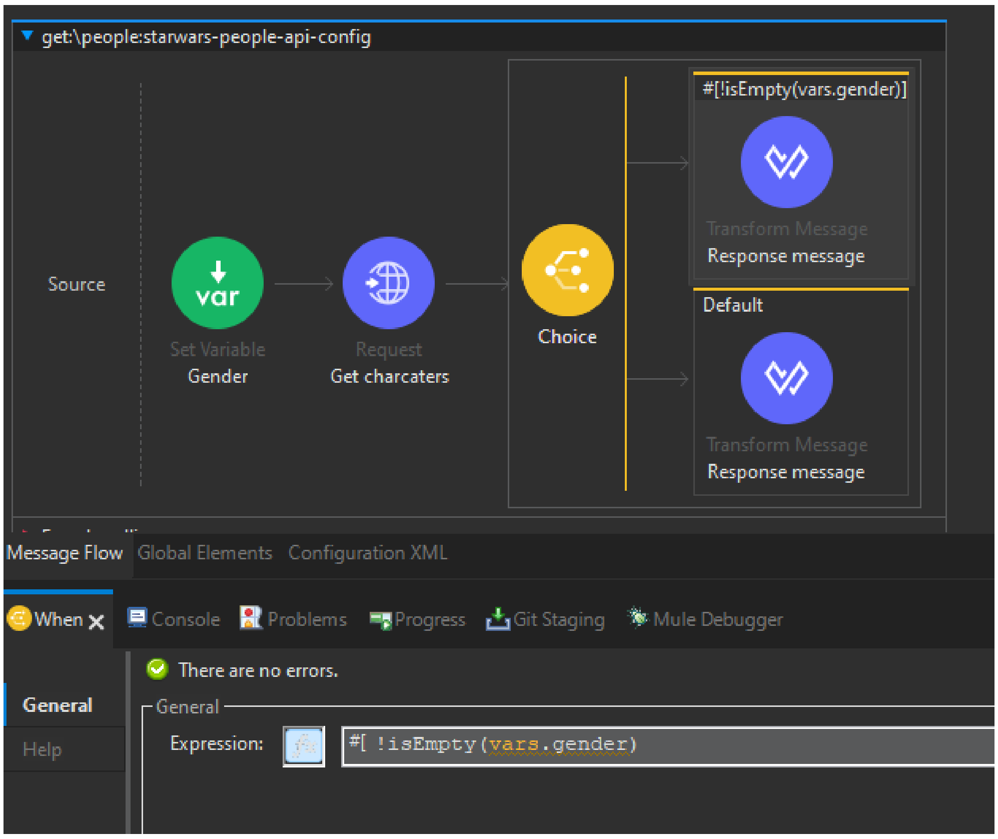
Display Name: Gender

Settings

Name: gender

Value: `#[ attributes.queryParams.gender`

Now, a choice was used with the conditional that if the query param has a value, it will go to the first answer and filter by gender.

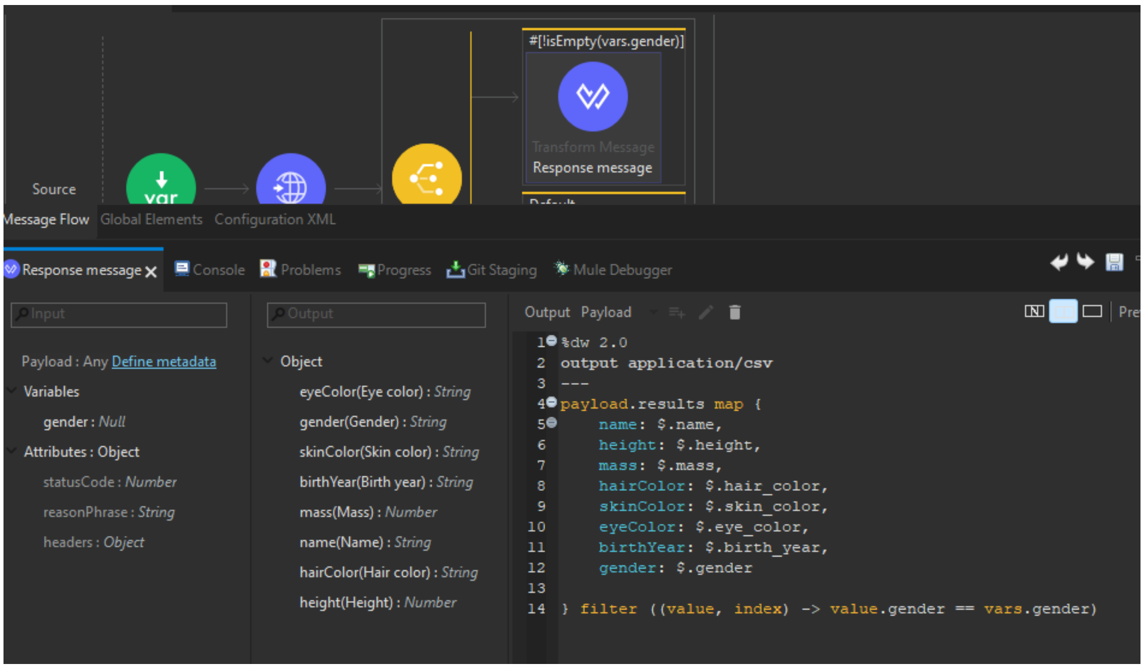


There are no errors.

General

Expression: `#[ !isEmpty(vars.gender)`

As the response has to be of type csv, it is specified that the output will be of this type specially.

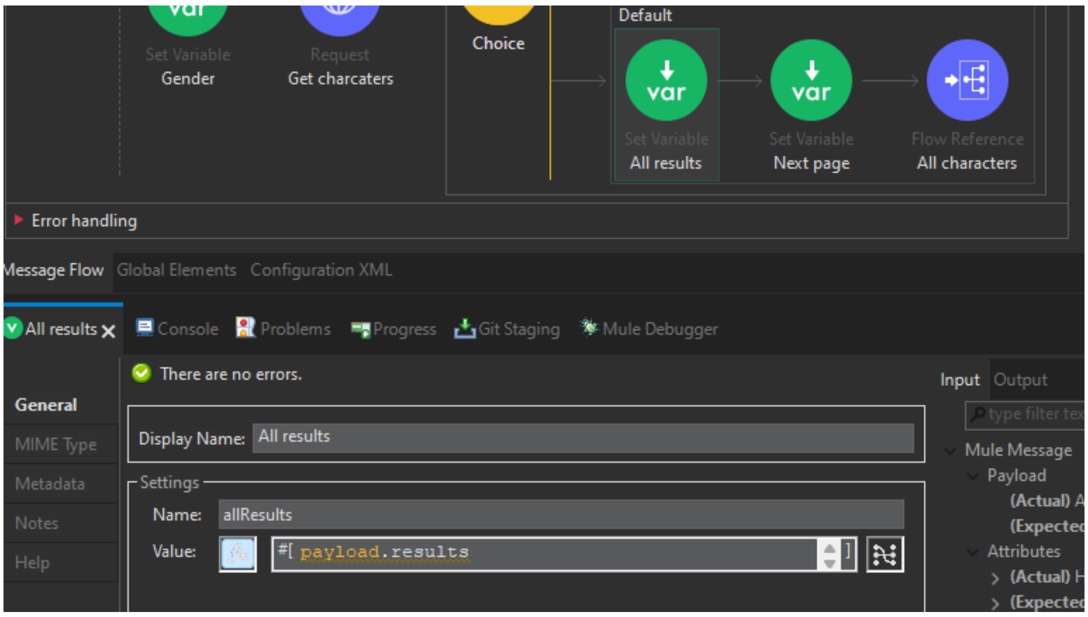


Payload : Any Define metadata
Variables
gender : Null
Attributes : Object
statusCode : Number
reasonPhrase : String
headers : Object

Object
eyeColor(Eye color) : String
gender(Gender) : String
skinColor(Skin color) : String
birthYear(Birth year) : String
mass(Mass) : Number
name(Name) : String
hairColor(Hair color) : String
height(Height) : Number

```
%dw 2.0
output application/csv
---
payload.results map {
    name: $.name,
    height: $.height,
    mass: $.mass,
    hairColor: $.hair_color,
    skinColor: $.skin_color,
    eyeColor: $.eye_color,
    birthYear: $.birth_year,
    gender: $.gender

} filter ((value, index) -> value.gender == vars.gender)
```

And if the query param is empty, it will go to the default answer, which is the collection of all characters.
To obtain all the characters, it will be done recursively. A variable is created to store the first result of the request.

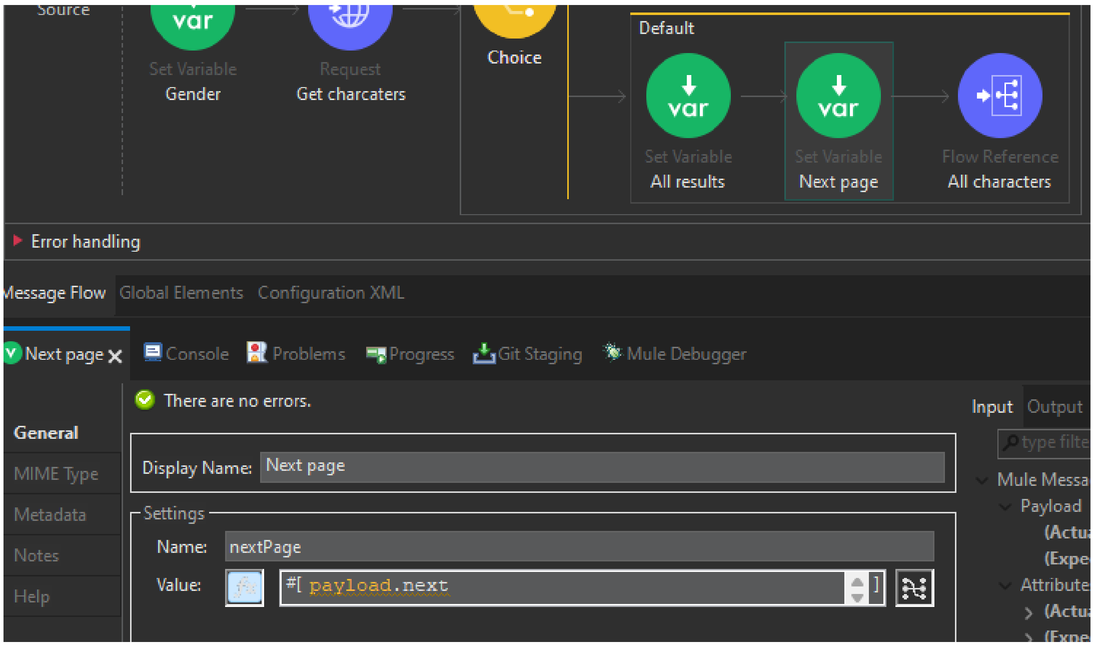


There are no errors.

Display Name: All results

Settings
Name: allResults
Value: #[payload.results]

Then, the url of the following page will be saved.



There are no errors.

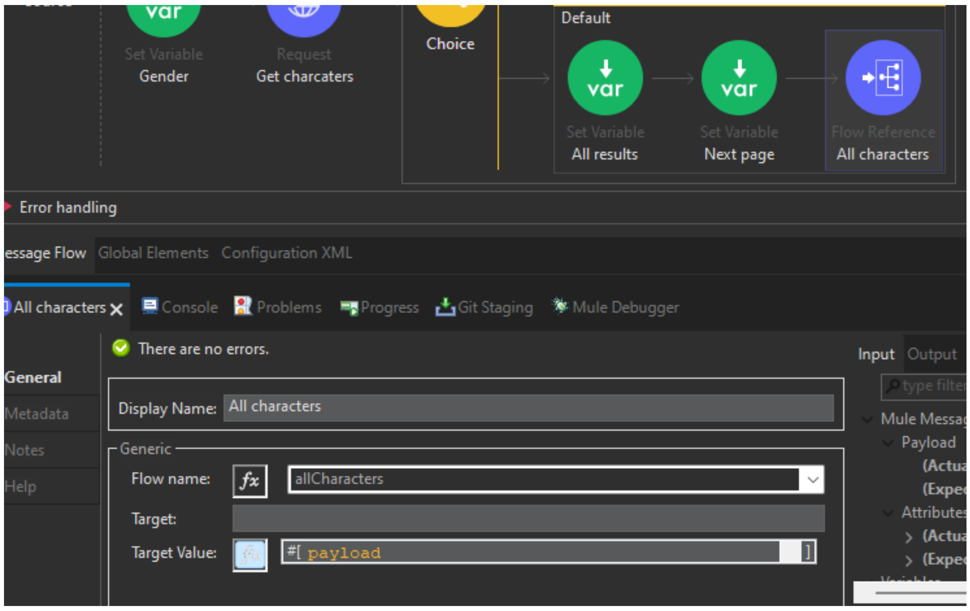
Display Name: Next page

Settings

Name: nextPage

Value: #[payload.next]

A subflow will be created to call the following pages recursively.



There are no errors.

Display Name: All characters

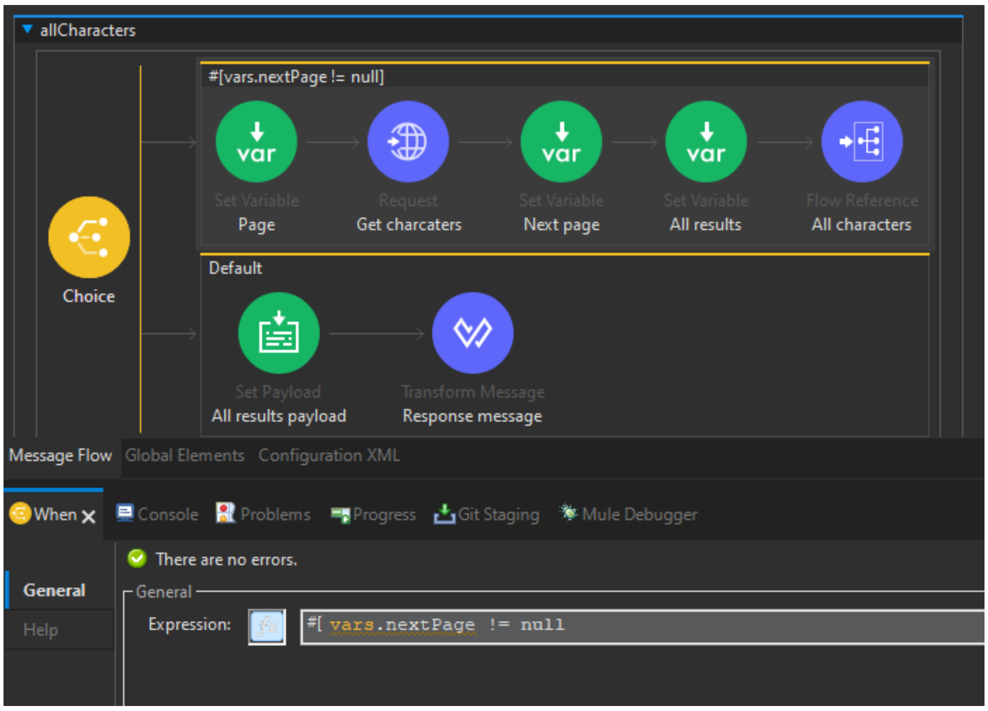
Generic

Flow name: allCharacters

Target:

Target Value: #[payload]

In the subflow, a choice is created where the condition is that the variable where the next page is stored is different from null.

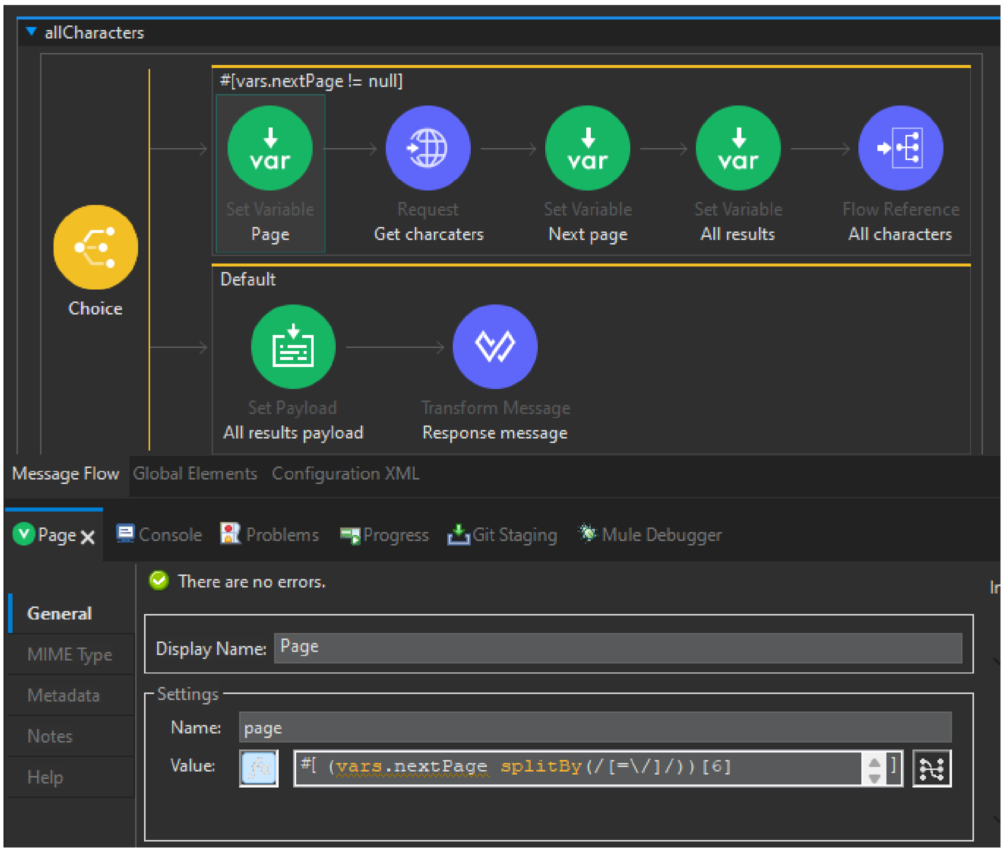


There are no errors.

General

Expression: `#[ vars.nextPage != null`

In the following variable is stored the next page number, which is obtained by separating the query param from the previously stored url.



There are no errors.

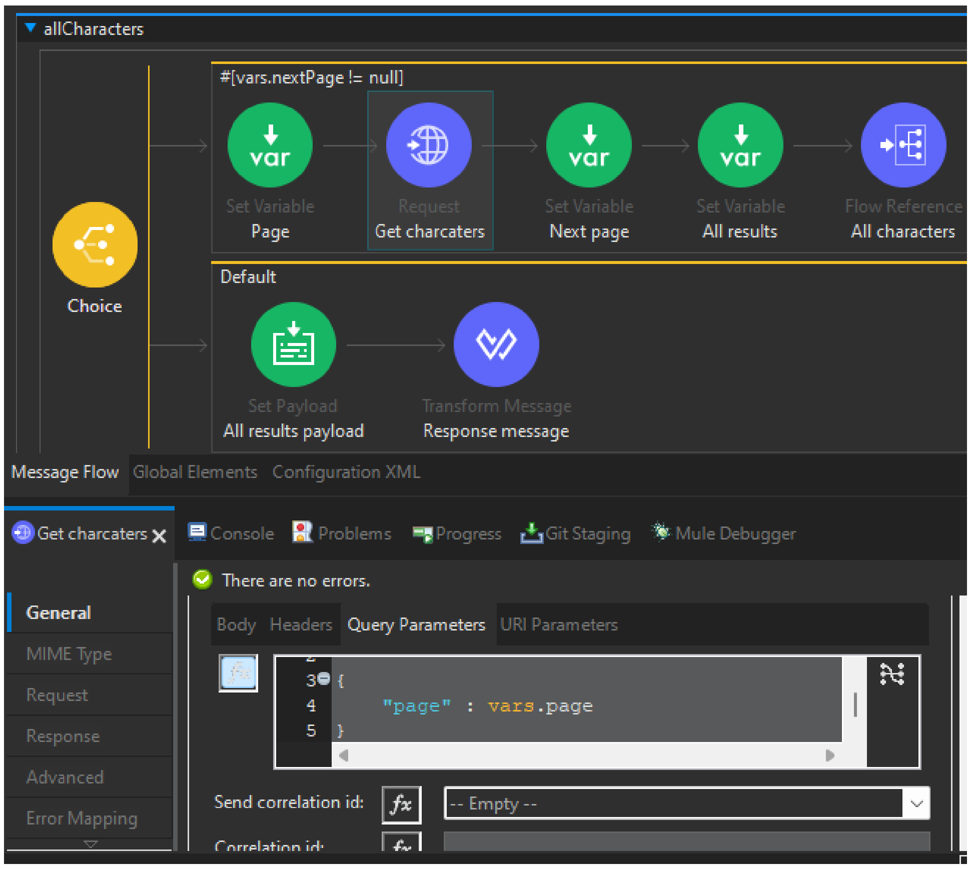
Display Name: Page

Settings

Name: page

Value: `#[ (vars.nextPage splitBy(/[=\/]/))[6]`

The swapi call is made again, now defining the query param with the page number obtained previously.



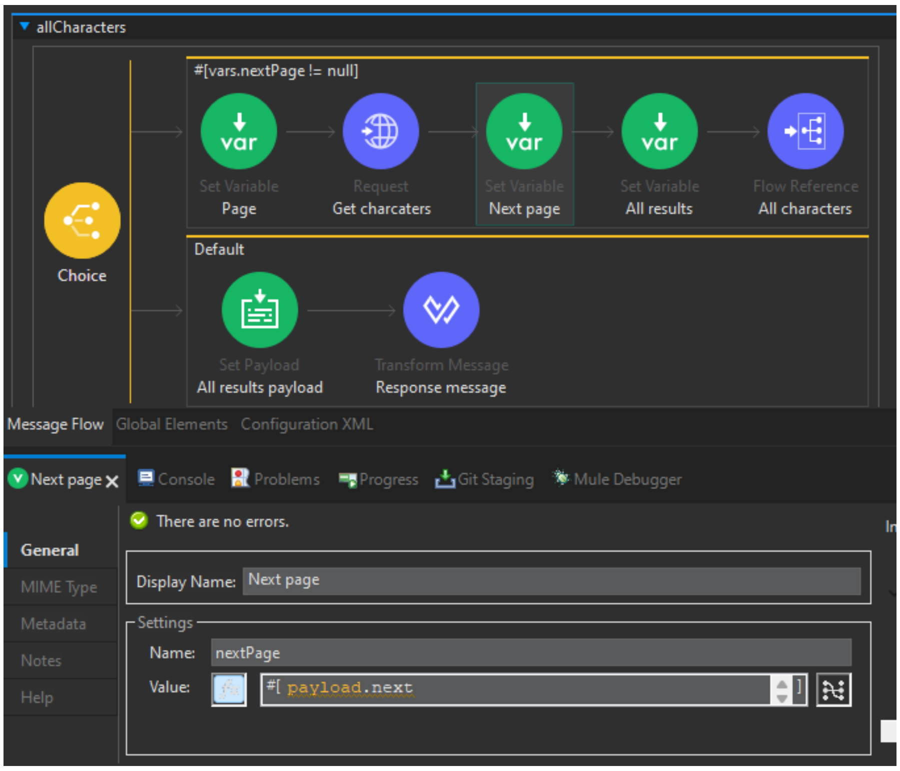
There are no errors.

```
{
    "page" : vars.page
}
```

Send correlation id: -- Empty --

Correlation id:

The url of the following page is saved again.



There are no errors.

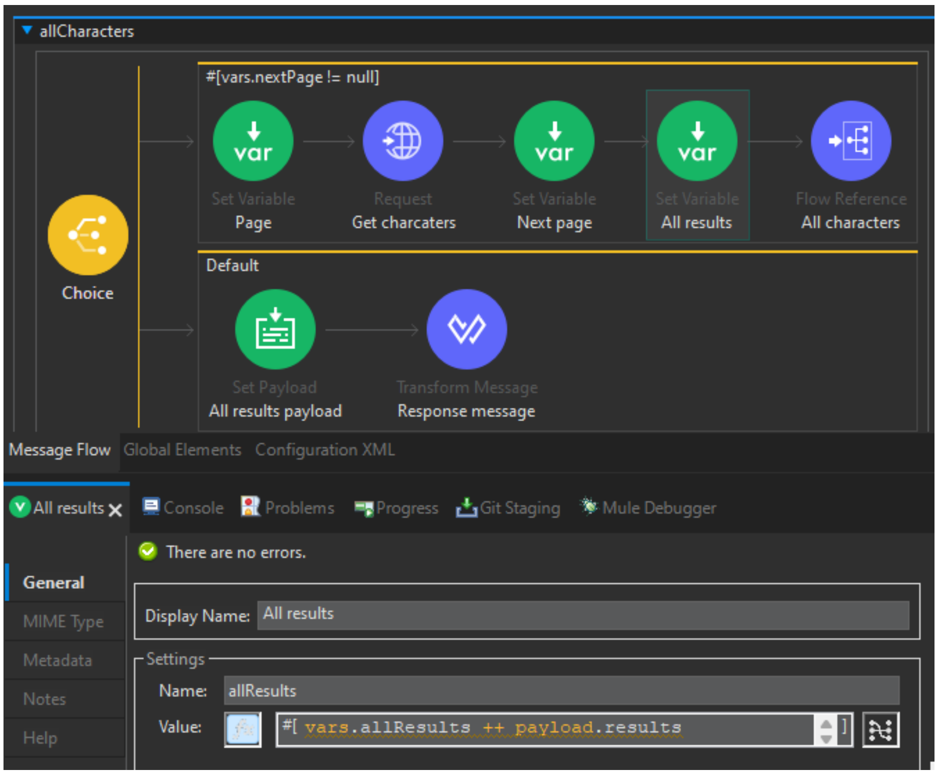
Display Name: Next page

Settings

Name: nextPage

Value: #[payload.next]

The following variable stores the answers of all previous calls made.



There are no errors.

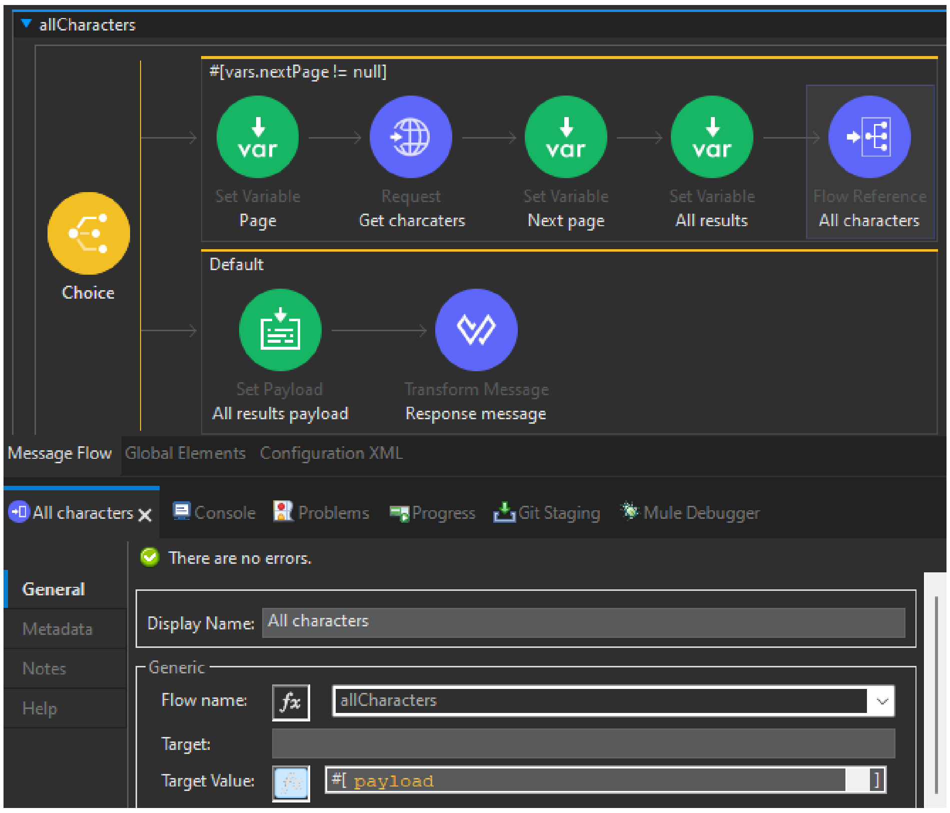
Display Name: All results

Settings

Name: allResults

Value: #[vars.allResults ++ payload.results]

In the flow reference, the call is made to the same subflow, this will be done until the choice condition is no longer fulfilled.



There are no errors.

Display Name: All characters

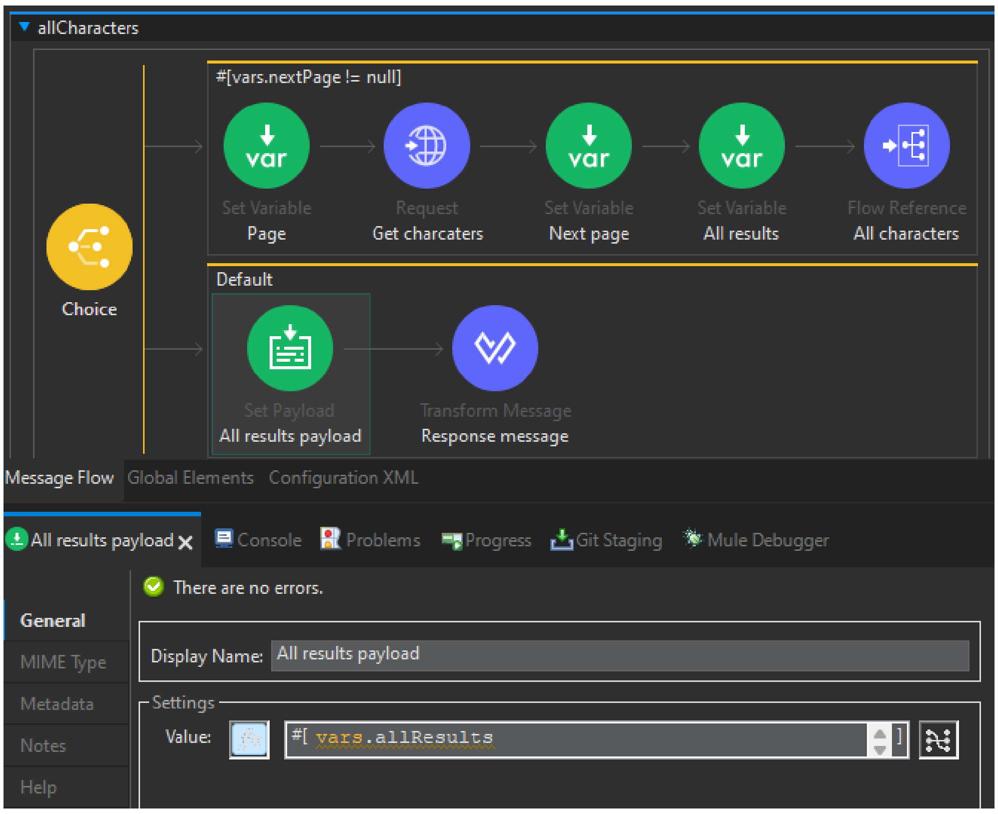
Generic

Flow name: allCharacters

Target:

Target Value: #[payload]

In the default option, a set payload will be set to store all the records retrieved from the previously made requests.



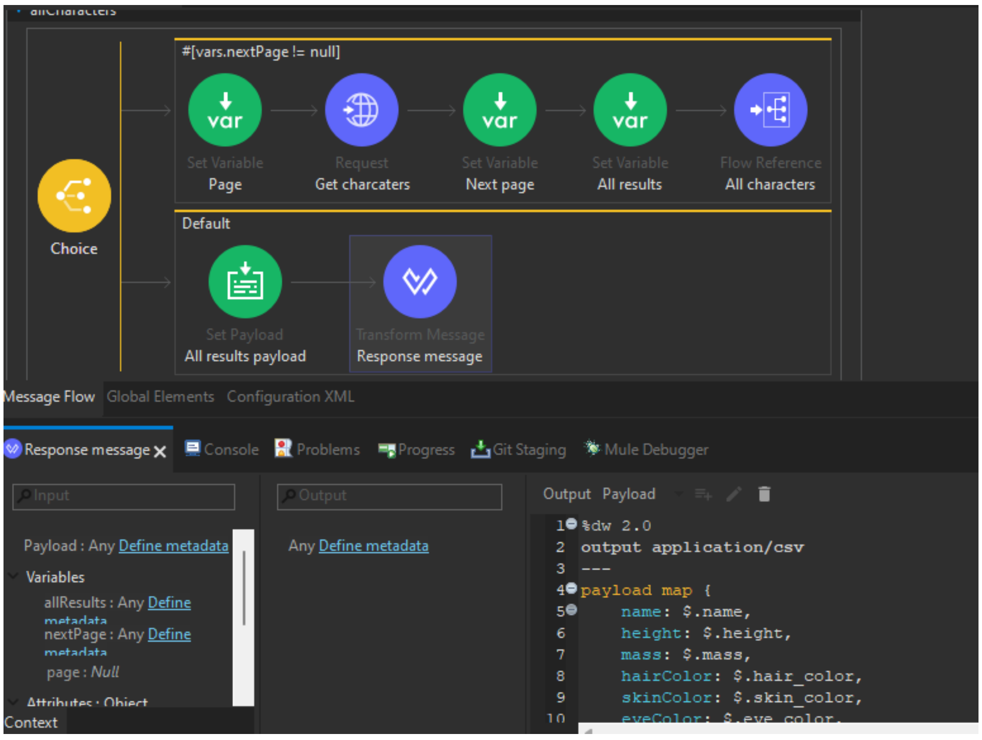
There are no errors.

Display Name: All results payload

Settings

Value: #[vars.allResults]

Finally, a transform message is set to display the final response in csv format.



Payload : Any Define metadata

Variables

allResults : Any Define metadata

nextPage : Any Define metadata

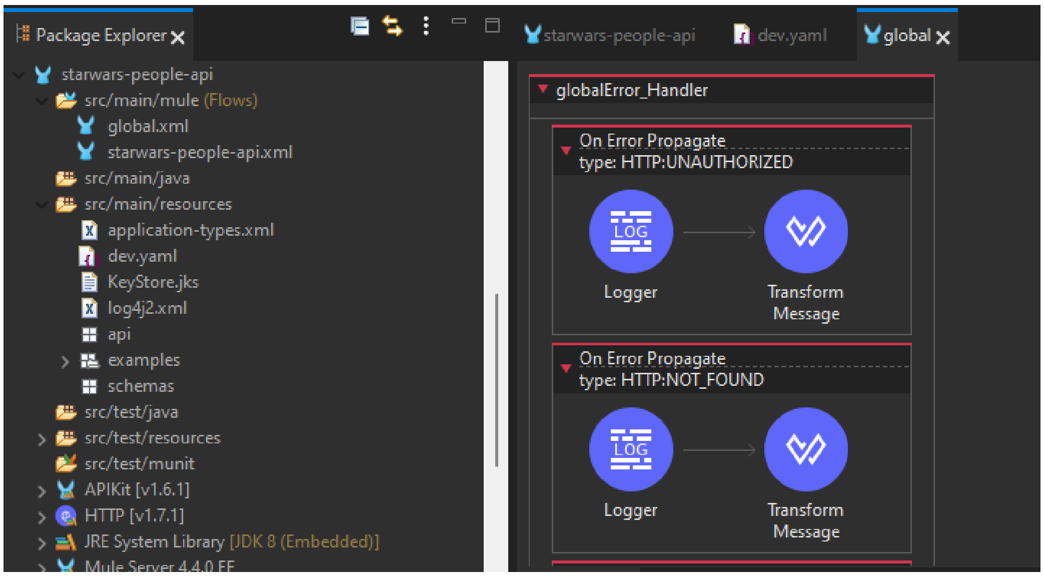
page : Null

Attributes : Object

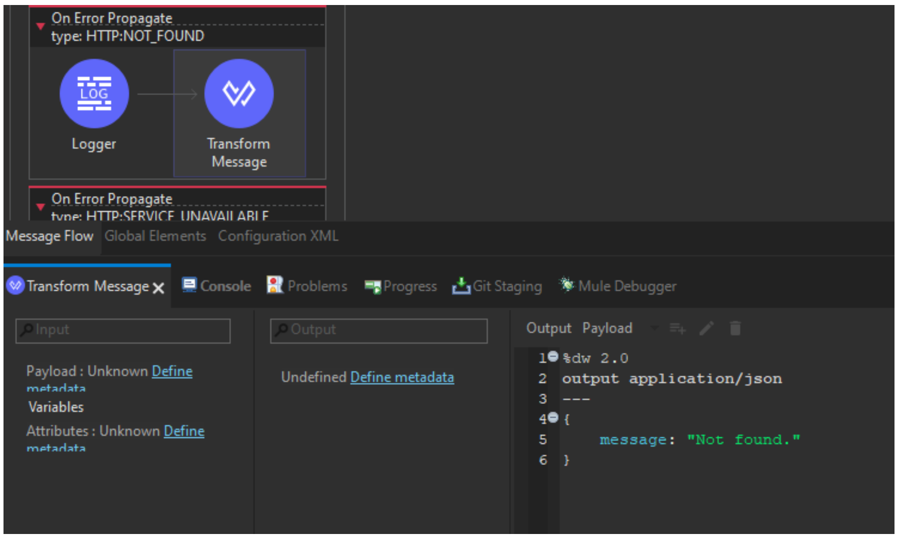
Any Define metadata

```
%dw 2.0
output application/csv
---
payload map {
    name: $.name,
    height: $.height,
    mass: $.mass,
    hairColor: $.hair_color,
    skinColor: $.skin_color,
    eyeColor: $.eye_color,
```

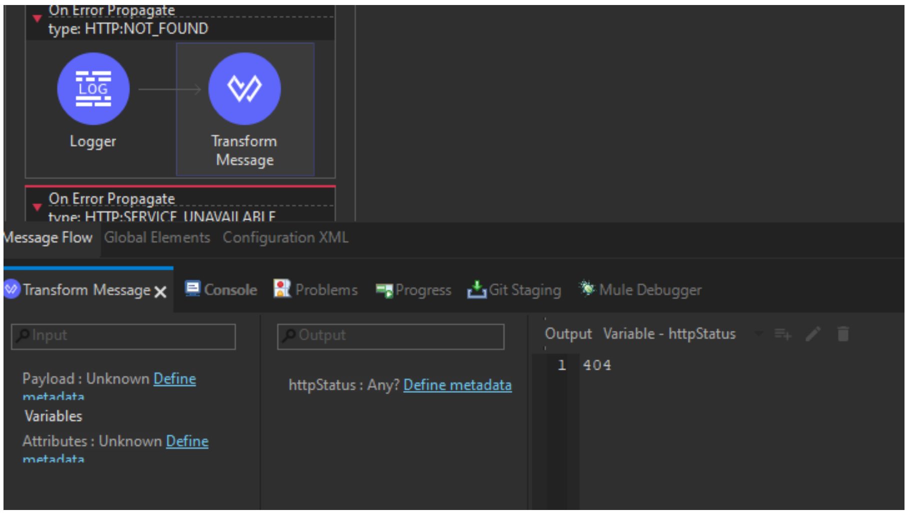
In order to control errors, a global file is created, mapping different common errors.



Each error has its own response message, and inside the transmor message, a variable is created to be the error code.

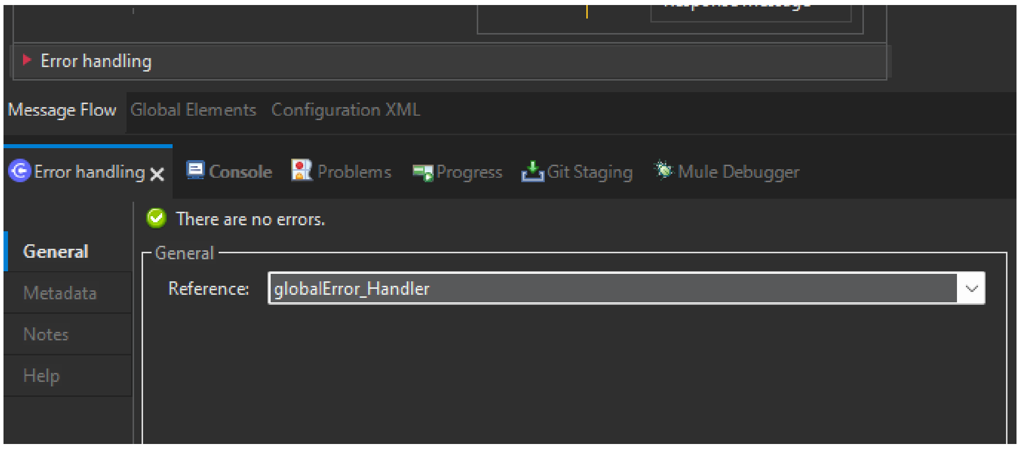


```
%dw 2.0
output application/json
---
{
    message: "Not found."
}
```



```
404
```

And so that the flow can direct the errors to the global file, it is referenced in the error handling of the flow.



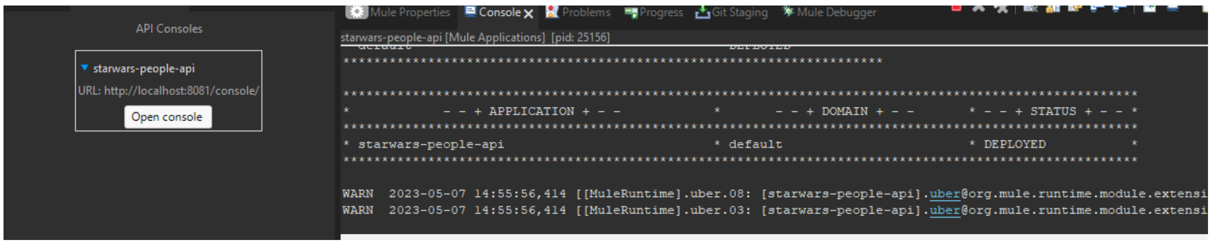
There are no errors.

General

Reference: globalError_Handler

Tests

Now, it's time to test, run the application until it is deployed correctly.



starwars-people-api

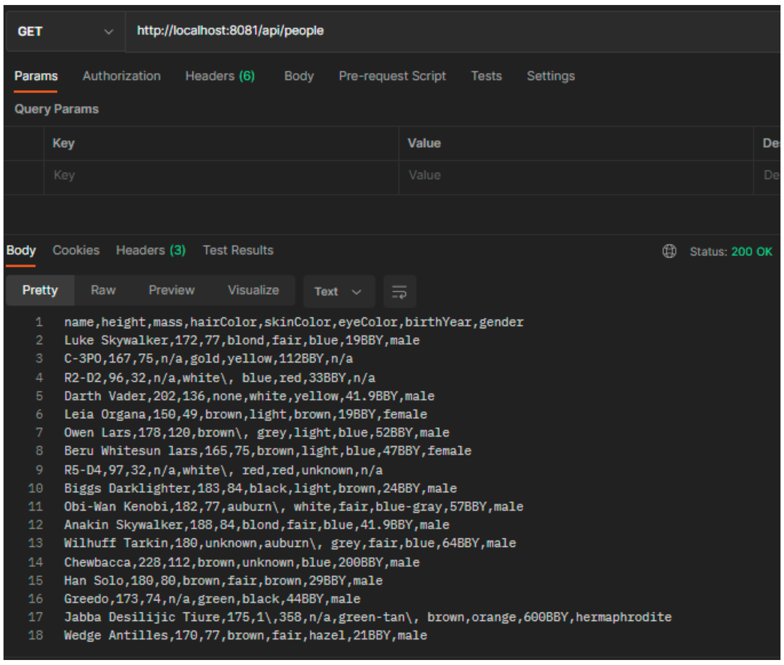
URL: http://localhost:8081/console/

starwars-people-api [Mule Applications] [pid: 25156]

```
**********************************************************************

**********************************************************************************************************
*            - - + APPLICATION + - -            *       - - + DOMAIN + - -       * - - + STATUS + - - *
**********************************************************************************************************
* starwars-people-api                           * default                        * DEPLOYED           *
**********************************************************************************************************

WARN  2023-05-07 14:55:56,414 [[MuleRuntime].uber.08: [starwars-people-api].uber@org.mule.runtime.module.extensi
WARN  2023-05-07 14:55:56,414 [[MuleRuntime].uber.03: [starwars-people-api].uber@org.mule.runtime.module.extensi
```

In postman a request is created to test that the application returns the expected response.

This is the response from all characters in csv.

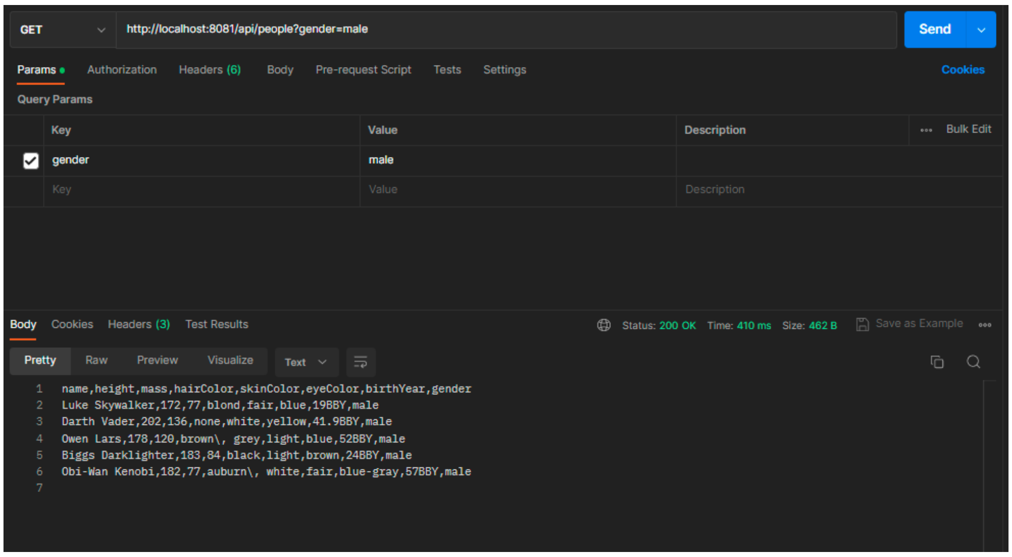


GET http://localhost:8081/api/people

Status: 200 OK

```
name,height,mass,hairColor,skinColor,eyeColor,birthYear,gender
Luke Skywalker,172,77,blond,fair,blue,19BBY,male
C-3PO,167,75,n/a,gold,yellow,112BBY,n/a
R2-D2,96,32,n/a,white\, blue,red,33BBY,n/a
Darth Vader,202,136,none,white,yellow,41.9BBY,male
Leia Organa,150,49,brown,light,brown,19BBY,female
Owen Lars,178,120,brown\, grey,light,blue,52BBY,male
Beru Whitesun lars,165,75,brown,light,blue,47BBY,female
R5-D4,97,32,n/a,white\, red,red,unknown,n/a
Biggs Darklighter,183,84,black,light,brown,24BBY,male
Obi-Wan Kenobi,182,77,auburn\, white,fair,blue-gray,57BBY,male
Anakin Skywalker,188,84,blond,fair,blue,41.9BBY,male
Wilhuff Tarkin,180,unknown,auburn\, grey,fair,blue,64BBY,male
Chewbacca,228,112,brown,unknown,blue,200BBY,male
Han Solo,180,80,brown,fair,brown,29BBY,male
Greedo,173,74,n/a,green,black,44BBY,male
Jabba Desilijic Tiure,175,1\,358,n/a,green-tan\, brown,orange,600BBY,hermaphrodite
Wedge Antilles,170,77,brown,fair,hazel,21BBY,male
```

And this is the answer if filtered by gender.



http://localhost:8081/api/people?gender=male

Query Params

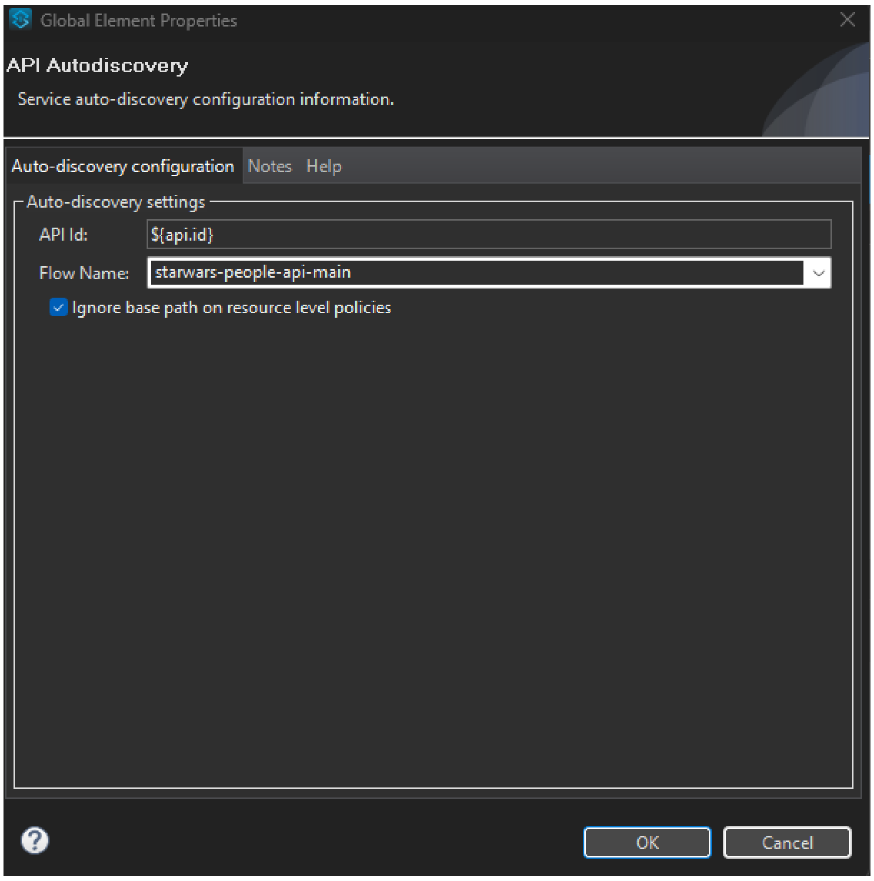
Key	Value	Description
gender	male	

Status: 200 OK Time: 410 ms Size: 462 B

```
name,height,mass,hairColor,skinColor,eyeColor,birthYear,gender
Luke Skywalker,172,77,blond,fair,blue,19BBY,male
Darth Vader,202,136,none,white,yellow,41.9BBY,male
Owen Lars,178,120,brown\, grey,light,blue,52BBY,male
Biggs Darklighter,183,84,black,light,brown,24BBY,male
Obi-Wan Kenobi,182,77,auburn\, white,fair,blue-gray,57BBY,male
```

Cloudhub

Now, to be able to deploy in cloudhub, it is necessary to add an API Autodiscovery, which will be the one that will link to the API ID.



API Autodiscovery

Service auto-discovery configuration information.

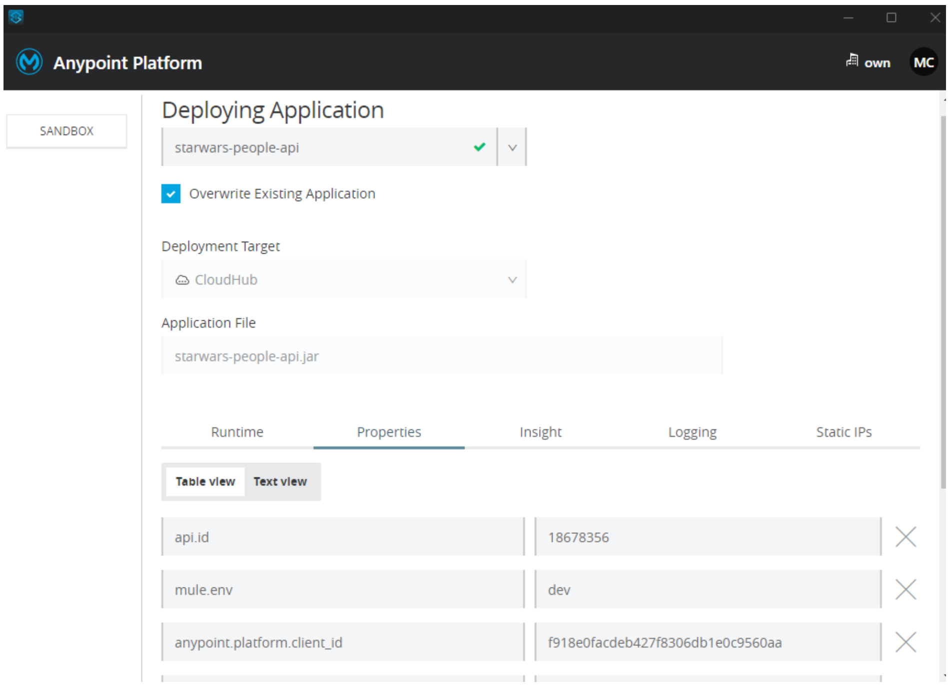
Auto-discovery settings

API Id: ${api.id}

Flow Name: starwars-people-api-main

✓ Ignore base path on resource level policies

From Anypoint Studio there is an option to deploy, here you have to set certain properties such as the api id, the environment and the credentials of the environment or business group of the Anypoint Platform.



Deploying Application

starwars-people-api

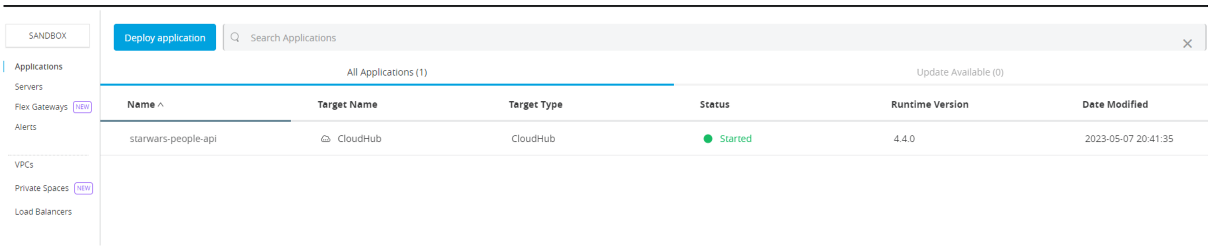
[✓] Overwrite Existing Application

Deployment Target: CloudHub

Application File: starwars-people-api.jar

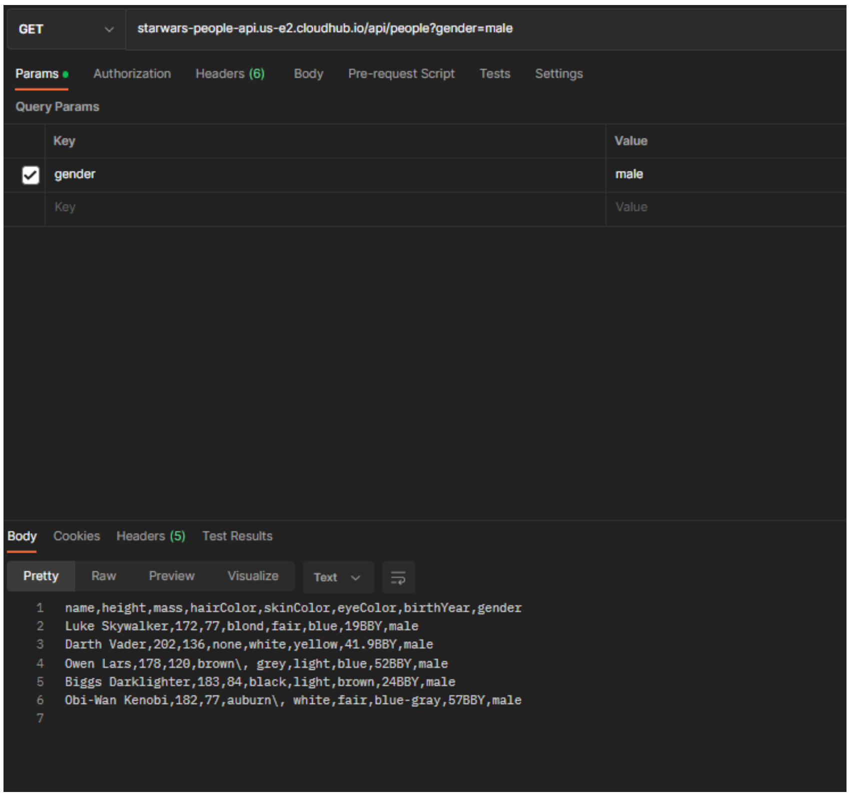
api.id	18678356
mule.env	dev
anypoint.platform.client_id	f918e0facdeb427f8306db1e0c9560aa

Once successfully deployed, the application appears in runtime manager.



Name ^	Target Name	Target Type	Status	Runtime Version	Date Modified
starwars-people-api	CloudHub	CloudHub	● Started	4.4.0	2023-05-07 20:41:35

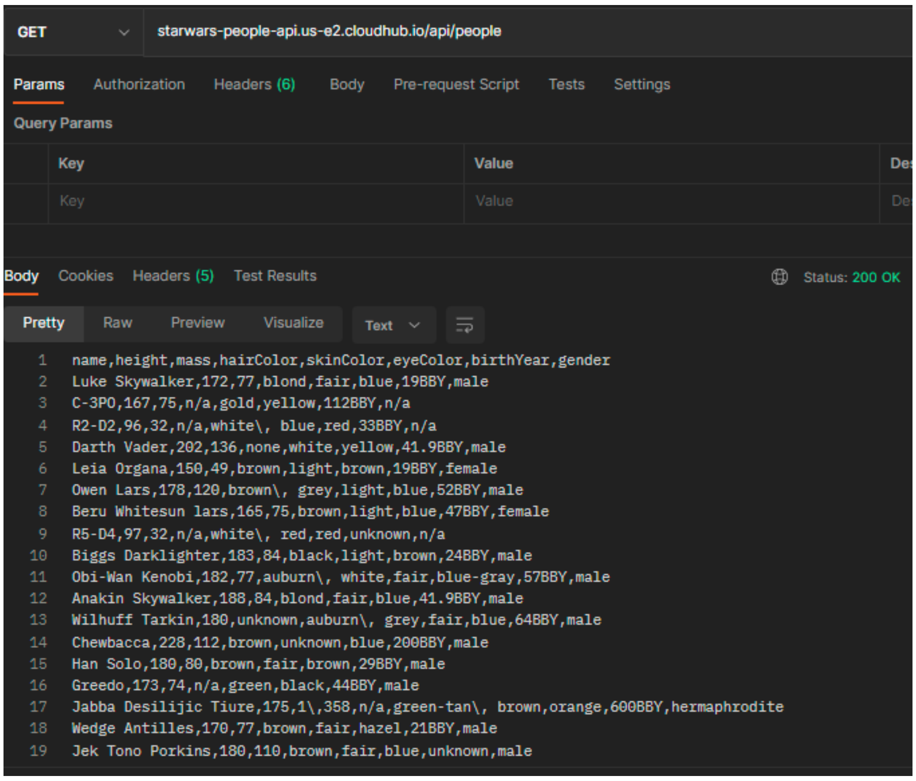
Now, you test in postman with the url of the cloudhub.



starwars-people-api.us-e2.cloudhub.io/api/people?gender=male

Key	Value
gender	male

```
name,height,mass,hairColor,skinColor,eyeColor,birthYear,gender
Luke Skywalker,172,77,blond,fair,blue,19BBY,male
Darth Vader,202,136,none,white,yellow,41.9BBY,male
Owen Lars,178,120,brown\, grey,light,blue,52BBY,male
Biggs Darklighter,183,84,black,light,brown,24BBY,male
Obi-Wan Kenobi,182,77,auburn\, white,fair,blue-gray,57BBY,male
```



starwars-people-api.us-e2.cloudhub.io/api/people

```
name,height,mass,hairColor,skinColor,eyeColor,birthYear,gender
Luke Skywalker,172,77,blond,fair,blue,19BBY,male
C-3PO,167,75,n/a,gold,yellow,112BBY,n/a
R2-D2,96,32,n/a,white\, blue,red,33BBY,n/a
Darth Vader,202,136,none,white,yellow,41.9BBY,male
Leia Organa,150,49,brown,light,brown,19BBY,female
Owen Lars,178,120,brown\, grey,light,blue,52BBY,male
Beru Whitesun lars,165,75,brown,light,blue,47BBY,female
R5-D4,97,32,n/a,white\, red,red,unknown,n/a
Biggs Darklighter,183,84,black,light,brown,24BBY,male
Obi-Wan Kenobi,182,77,auburn\, white,fair,blue-gray,57BBY,male
Anakin Skywalker,188,84,blond,fair,blue,41.9BBY,male
Wilhuff Tarkin,180,unknown,auburn\, grey,fair,blue,64BBY,male
Chewbacca,228,112,brown,unknown,blue,200BBY,male
Han Solo,180,80,brown,fair,brown,29BBY,male
Greedo,173,74,n/a,green,black,44BBY,male
Jabba Desilijic Tiure,175,1\,358,n/a,green-tan\, brown,orange,600BBY,hermaphrodite
Wedge Antilles,170,77,brown,fair,hazel,21BBY,male
Jek Tono Porkins,180,110,brown,fair,blue,unknown,male
```