MuleSoft

Super StarWars Process API

Super Organization
v1.0, 2024-05-16

Version	Date	Author	Description
1.0	2024/05/16	Kevin Jonathan Medina Resendis	Initial version

Table of contents

Introduction

This document defines the technical components to implement the proposed API/Integration solution.

Audience

This document's audience primarily includes architects, consultants, developers, and testers engaged in architecting, designing, developing, and testing API-based solutions.

Glossary

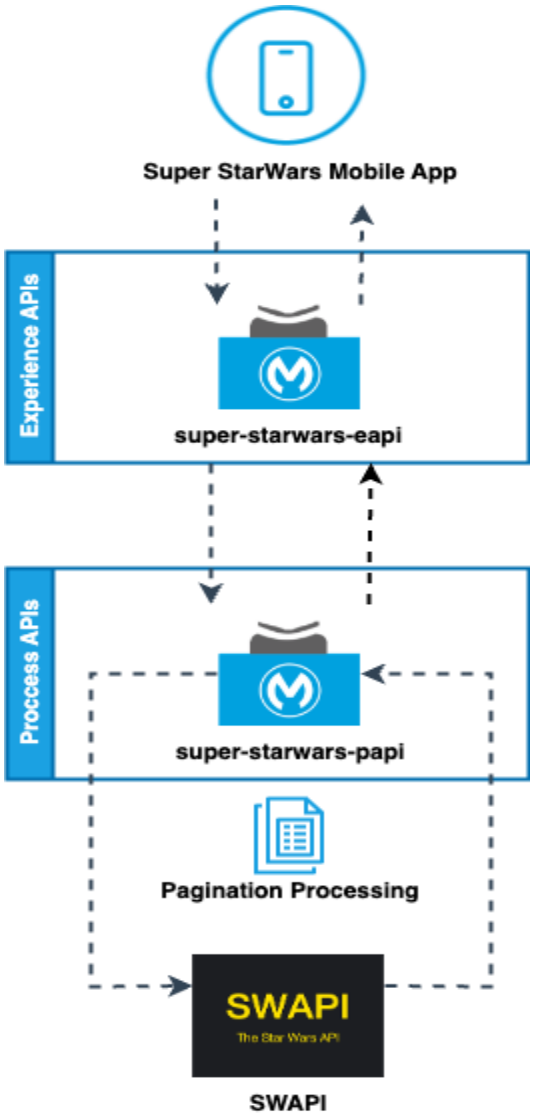
The general terms and acronyms referenced in this document

References

<TBD>

Use-case overview

Super StarWars Process API provides the functionality to retrieve the data related to the characters from the movie franchise by processing SWAPI API Pagination Results and formatting to CSV Format.



Assumptions

This section covers any assumptions that have been made in the process of designing the solution.

- SWAPI API is a public API and no authentication process is needed.

Prerequisites

This section covers any prerequisites that are required as part of the solution.

- NA

Business Requirements

This section captures high-level requirements as part of the solution.

- Business needs all the characters information in CSV Format to be used and compared by the Super StarWars Mobile Application users.
- Business needs a filter that can be applied to the characters information based on the character gender.

Technical Design

This section captures the technical design details of the solution.

Overview

Mulesoft Super StarWars Experience API will invoke super-starwars-papi to check starwars characters information and share it with consumers of Super StarWars Mobile Application.

Logical View

The sequence diagram is too big to be shown in the document and will be added to a folder and be part of the documentation.

Components/Mule Applications

Mule Applications			
S.No	Application Name	Type	Description
1	super-starwars-papi	API	Super StarWars Experience API
2	SWAPI(StarWars API)	API	Public StarWars API

Non-Functional Requirements

This section captures the non-functional requirements of all APIs related to the use-case(s)/solution.

Security

Authentication		
S.No	Application Name	Authentication Type/Policy
1	super-starwars-papi	Client ID Enforcement Policy
2	SWAPI	No Authentication

Common NFR

NA

HTTP/Web service

Properties for API

Reconnection Frequency	2000 milliseconds
Reconnection Max Attempts	2
HTTP Timeout	15000 milliseconds

Super StarWars Process API

Mule Application Details

API Overview		
S.No	Property	Details
1	Application Technical Name	super-starwars-papi
2	Application Business Name	Super StarWars PAPI
3	Aliases	Super StarWars PAPI
4	Mule Application Type	API
5	API Type	Process
6	Business Domain	Mobile App
7	Purpose of API	The Super Star Wars API will retrieve the data related to the characters from the movie franchise.
8	API Owner	Super Organization Dev Team
9	Version(s)	v1

10	Initial Release Date (optional)	May 17th, 2024
11	app.url.name	TBD
12	Source Code Repo	TBD

Health-check/Ping end-point	
Endpoint	GET {BASE_URI}/api/v1/healthcheck

Success	Return a **200** upon receipt of a request on the /healthcheck end-point with the next body:
	{
	"status": "OK",
	"dependencies":[
	{
	"name": "SWAPI",
	"status": "OK"
	}
	]
	}

HTTP Status Code	Possible HTTP status codes for the response include: • 200 Request accepted • 401 Unauthorized – for errors in an API authentication

	• 500 Internal Server Error – for unexpected failures

Resources

Resources			
S.No	Resource Name	Supported Methods	Description
1	/healthcheck	GET	This Resource will retrieve application status. Media Type: application/json
2	/characters	GET	This Resource will retrieve StarWars characters information consuming SWAPI API Pagination Results, this information could be filtered based on gender. Media Type: application/csv

GET: /characters

Retrieve characters information

RAML Specification Details

Input Request			
Name	Type	Mandatory	Comments
gender	Query Param	No	Possible Values: • male • female • unknown • n/a(for robots)

hypermediaSearch	Query Param	No	Possible Values: yes, no.

Response			
Name	Type	Mandatory	Comments
name	String	Yes	
height	String	Yes	
mass	String	Yes	
hair_color	String	Yes	
skin_color	String	Yes	
eye_color	String	Yes	
birth_year	String	Yes	
gender	String	Yes	
homeworld	String	No	*This applies only if the hypermediaSearch query parameter is sent.*
films	String	No	Films will be put together in a single string separated by a comma. *This applies only if the hypermediaSearch query parameter is sent.* *Ex. movie1, movie2, ...*
species	String	No	Species will be put together in a single string separated by a comma. *This*

			applies only if the hypermediaSearch query parameter is sent. *Ex. specie1, specie2, ...*
vehicles	String	No	Vehicles will be put together in a single string separated by a comma. *This applies only if the hypermediaSearch query parameter is sent.* *Ex. vehicle1, vehicle2, ...*
starships	String	No	Starships will be put together in a single string separated by a comma. *This applies only if the hypermediaSearch query parameter is sent.* *Ex. starship1, starship2, ...*
created	String	No	Date Format: "YYYY-MM-DDTHH:mm:ss.ssssssZ" *This applies only if the hypermediaSearch query parameter is sent.*
edited	String	No	Date Format: "YYYY-MM-DDTHH:mm:ss.ssssssZ" *This applies only if the hypermediaSearch query parameter is sent.*
url	String	No	*This applies only if the hypermediaSearch query parameter is sent.*

API Response Body Structure

HTTP Code	/ Format	Structure
200 OK	*application/csv*	**Basic Search Response:** name,height,mass,hair_color,skin_color,eye_color,birth_year,gender C-3PO,167,75,n/a,gold,yellow,112BBY,n/a R2-D2,96,32,n/a,white\, blue,red,33BBY,n/a R5-D4,97,32,n/a,white\, red,red,unknown,n/a **Hypermedia Search Response:** name,height,mass,hair_color,skin_color,eye_color,birth_year,gender,homeworld,films,species,vehicles,starships,created,edited,url C-3PO,167,75,n/a,gold,yellow,112BBY,n/a,Tatooine,A New Hope\, The Empire Strikes Back\, Return of the Jedi\, The Phantom Menace\, Attack of the Clones\, Revenge of the Sith,Droid,,,2014-12-10T15:10:51.357000Z,2014-12-20T21:17:50.309000Z,https://swapi.dev/api/people/2/

400 Bad Request	*application/json*	{
		"status": "ERROR",
		"code": "400",
		"message": "Bad Request",
		"context": "Detailed Description Error"
		}

401 Unauthorized	*application/json*	“Unauthorized or invalid client application credentials”

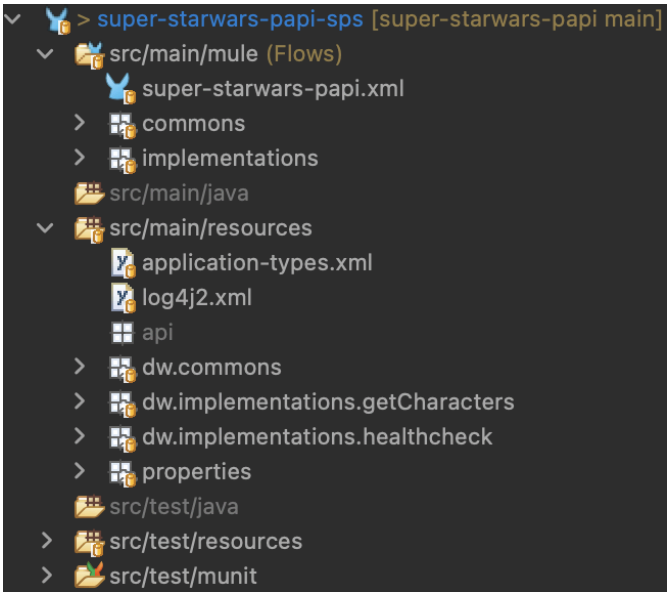
404 Not Found	*application/ json*	{
		"status": "ERROR",
		"code": "404",
		"message": "Not Found",
		"context": "Detailed Description Error"
		}

500 Internal Server Error	*application/ json*	{
		"status": "ERROR",
		"code": "500",
		"message": "Internal Server Error",
		"context": "Detailed Description Error"
		}

Project Structure:

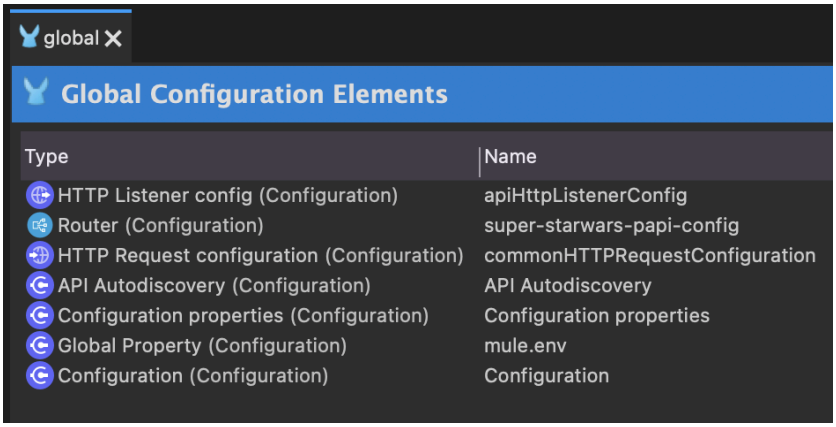
Prepare the Project by following the next structure:

- super-starwars-papi.xml(main file)
- Common folder
- Implementation folder
- Under src/resources add:
    - Properties folder
    - Dw folder



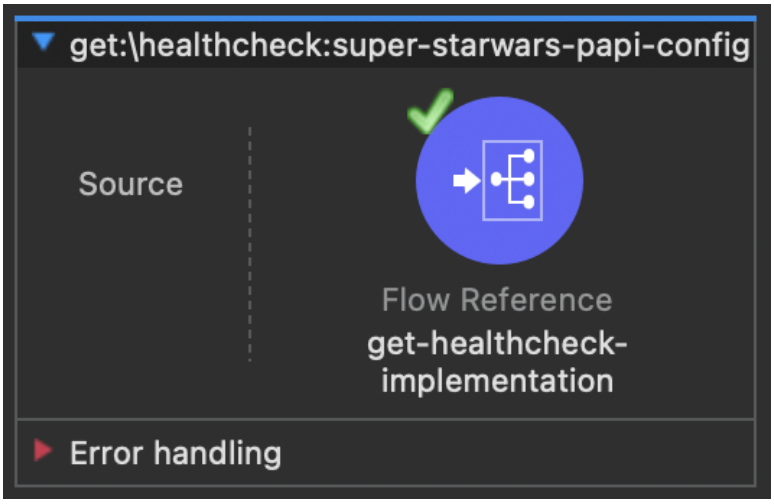
Create a new file **global.xml** to add all the API global configurations:

- OPTIONAL, put the file in the common folder(create a folder if it doesn´t exist).
- Move the **HTTP Listener Configuration** from **super-starwars-papi.xml**l file to **global.xml.**
- Move the **APIKit Configuration** from **super-starwars-papi.xml** file to **global.xml.**



Flow Process get healthcheck:

Add a Flow-Reference to the **get:\healthcheck:super-starwars-papi-config** Flow and configure it to call the **get-healthcheck-flow.**



Create a new file **get-healthcheck-implementation.xml** in the implementation folder(create a folder if it doesn´t exist). This file will contain the following flows:

- Get-healthcheck-flow

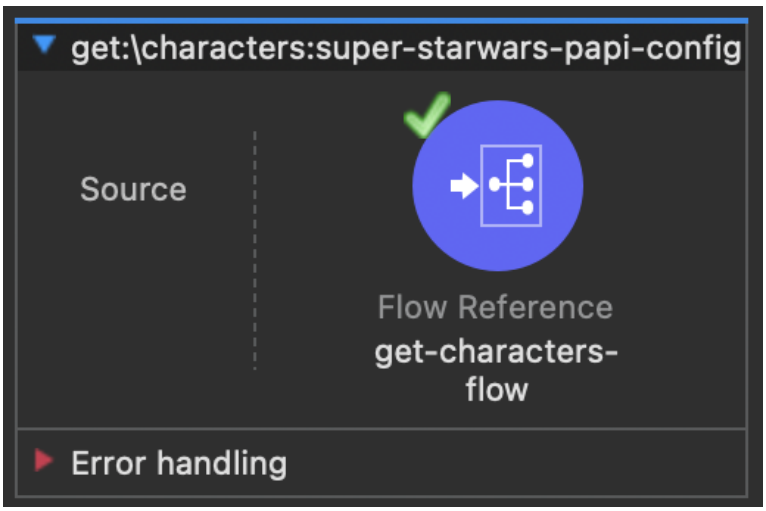
Inside **get-healthcheck-flow:**

- Add a Transform Message and set the **httpRequest** variable.
- Add a **Try Scope:**
    - Inside Try:
        - add a Flow-Reference and configure it to call common flow **common-http-request-flow**. This flow reference will use the **httpRequest** variable.
        - Add a Transform Message and set the **healthcheck** response for OK scenario.
    - Inside Catch:

- .Add On-Error-Continue, inside the error scope:
    - Add a Transform Message and set the **heathcheck** response for DOWN scenario

Flow Process get characters:

Add a Flow-Reference to the **get:\characters:super-starwars-papi-config** Flow and configure it to call the **get-characters-flow.**



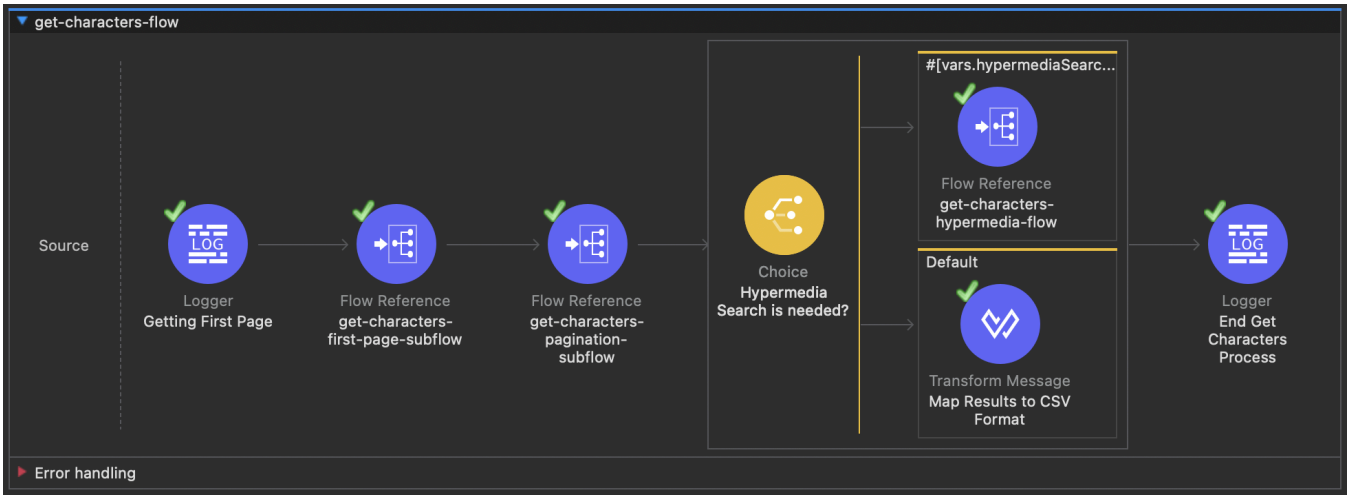
Create a new file **get-characters-implementation.xml** in the implementation folder(create a folder if it doesn´t exist). This file will contain the following flows:

- Get-character-flow
- Get-characters-first-page-subflow
- Get-characters-pagination-subflow

Inside **get-characters-flow**:

- Add a logger and print a message to indicate the process is getting started.
- Add a Flow-Reference and configure it to call get-characters-first-page-subflow.

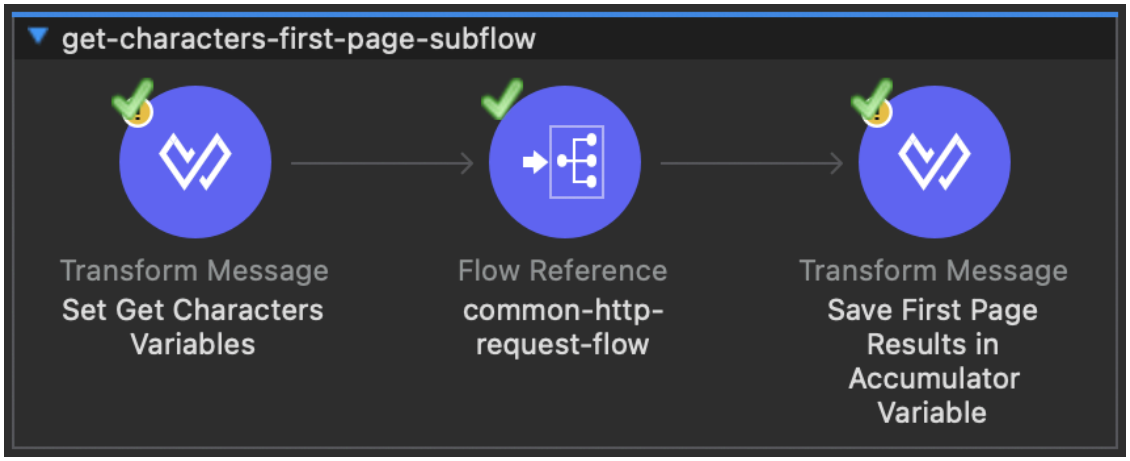
- Add a Flow-Reference and configure it to call get-characters-pagination-subflow.
- Add a Choice-Router and configure it to validate if the query parameter hypermediaSearch is present:
    - If the parameter is present, add a Flow-Reference and configure it to call get-characters-hypermedia-flow. This flow reference will use pagination accumulator results and perform a deep search to retrieve hypermedia data for each one of the results.
    - If the parameter is absent, add a Transform Message and map the pagination results as the payload response to CSV format. This is a ***Basic Search*** process. The request process ends here.
- Add a logger and print a message to indicate the process ended.



Inside **get-characters-first-page-subflow**:

- Add a Transform Message and set 4 set variables:
    - **hypermediaSearch**: stores a query parameter indicating if a hypermedia search is needed.

    - **gender**: stores a query parameter that will help us filter the results based on the character's gender if required.
    - **paginationAccumulator**: create an empty array, this variable will store all the results from all the calls to SWAPI API /people.
    - **httpRequest**: stores a common JSON object to send a request to the common-http-request component.
- Add a Flow-Reference and configure it to call common flow **common-http-request-flow**. This flow reference will use the httpRequest variable created in previous step.
- Add a Transform Message and update the **paginationAccumulator** variable with the HTTP response from SWAPI API, results are from page-1 iteration. Results will be filtered if the gender variable is present.

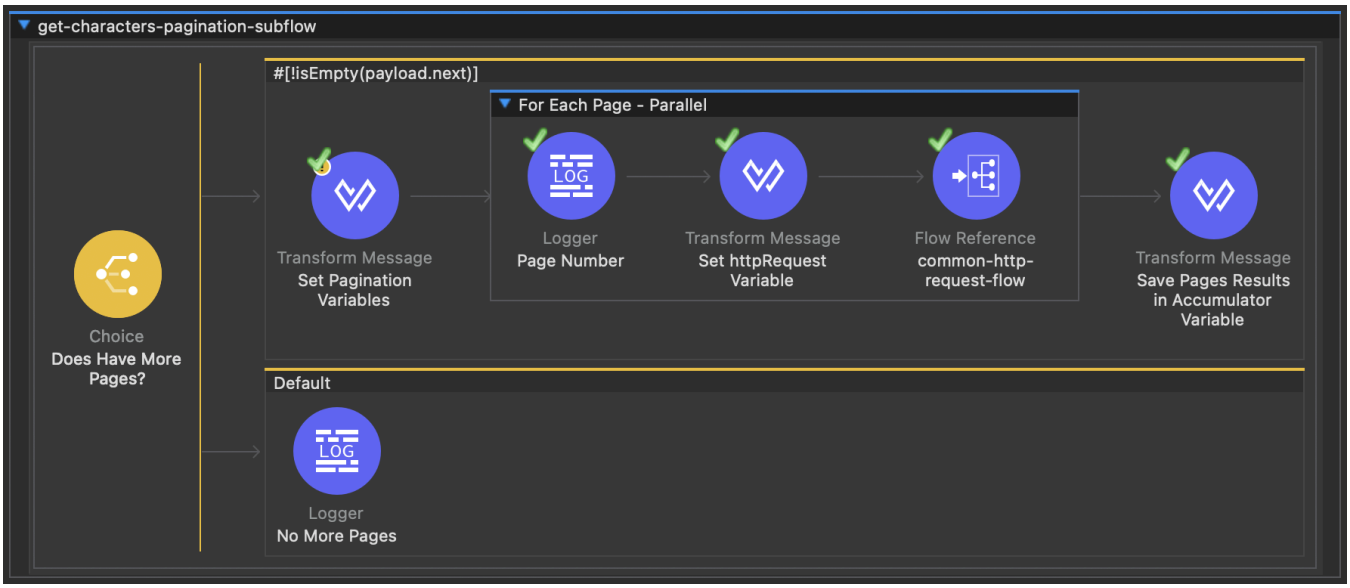


Inside **get-characters-pagination-subflow**:

- Add a Choice-Router to validate if a pagination process is needed:
    - If there are **more mages** to process:
        - Add a Transform Message and set 1 variable:
            - **Pages**: stores array that will save the number of pages to be processed.

- Add a Parallel For Each to iterate the pages saved in the pages variable, inside the for each scope we will follow the next steps:
    - Add a logger to print the current page in iteration.
    - Add a Transform Message and update the **httpRequest** variable by adding the "page" query parameter to the common JSON object, the "page" value will be the current page in iteration.
    - Add a Flow-Reference and configure it to call common flow **common-http-request-flow**. This flow reference will use the httpRequest variable updated in the previous step.
    - Add a Transform Message and update **paginationAccumulator** variable with the Parallel For Each result(accumulated HTTP Responses from SWAPI API pages calls). Results will be filtered if the gender variable is present.

- The **default** route: add a logger to print a message to indicate no more pages were found.



Create a new file **get-characters-implementation.xml** in the implementation folder(create a folder if it doesn´t exist). This file will contain the following flows:

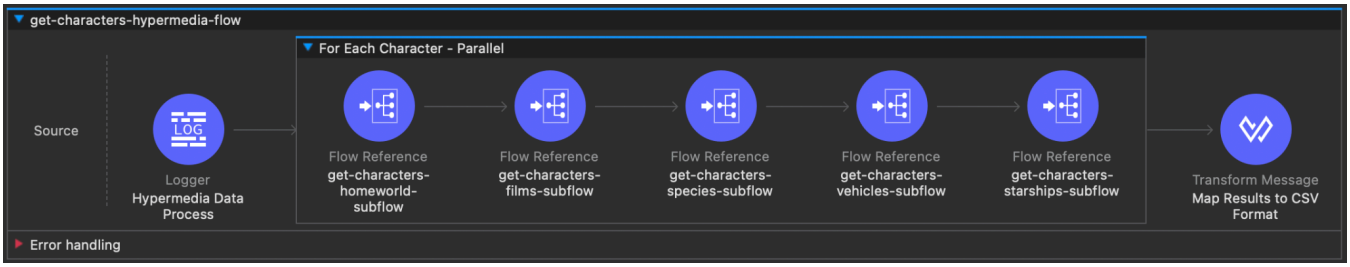
- Get-characters-hypermedia-flow
- Get-characters-homeworld-subflow
- Get-characters-films-subflow
- Get-characters-species-subflow
- Get-characters-vehicles-subflow
- Get-characters-starships-subflow

Inside **get-characters-hypermedia-flow:**

- Add a logger and print a message to indicate the hypermedia process is getting started.
- Add a Parallel For Each to iterate the results saved in the **paginationAccumulator** variable, inside the for each scope we will follow the next steps:
    - Add Flow-Reference and configure it to call **get-characters-homerworld-subflow.** This flow reference will use the current **payload.homeworld** of the current character payload in iteration
    - Add Flow-Reference and configure it to call **get-characters-films-subflow.** This flow reference will use the current **payload.films** of the current character payload in iteration.
    - Add Flow-Reference and configure it to call **get-characters-species-subflow.** This flow reference will use the current **payload.species** of the current character payload in iteration.

- Add Flow-Reference and configure it to call **get-characters-vehicles-subflow.** This flow reference will use the current **payload.vehicles** of the current character payload in iteration.
- Add Flow-Reference and configure it to call **get-characters-starships-subflow.** This flow reference will use the current **payload.starships** of the current character payload in iteration.

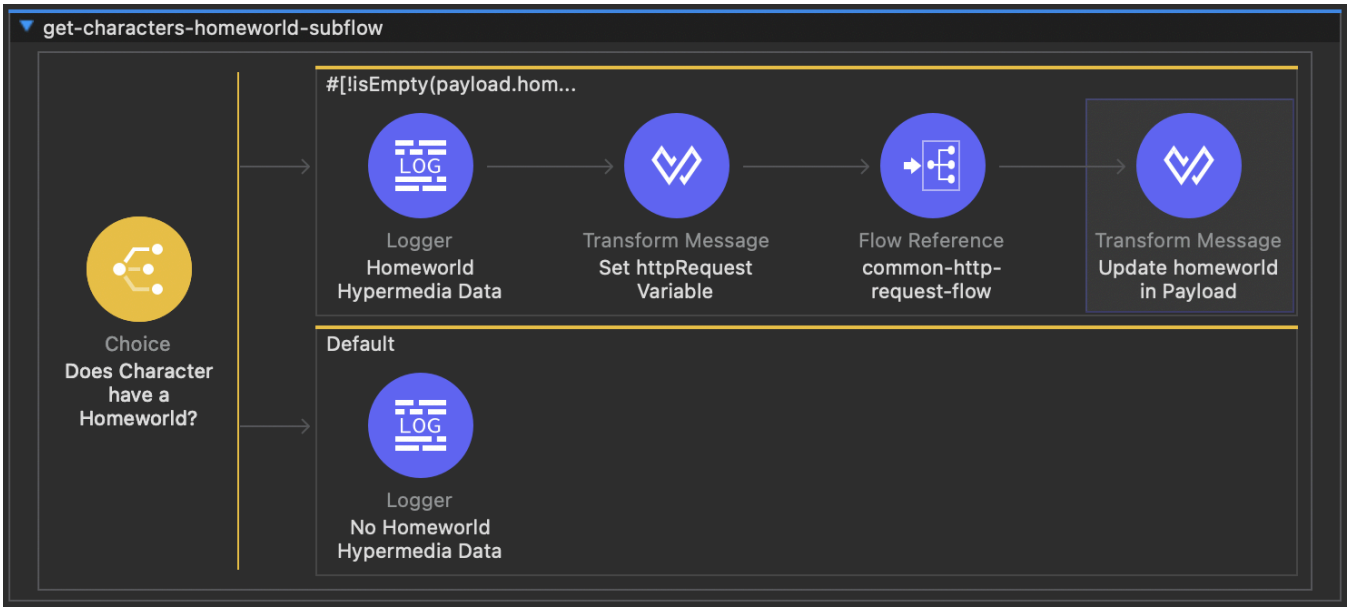
- Add a Transform Message and map the For each result as the **payload response** to CSV Format. This is a ***Hypermedia Search*** process. The request process ends here.



Inside **get-characters-homeworld-subflow:**

- Add a **Choice-Router** to validate if we need to search for the character's homeworld:
  - If **payload.homeworld** has a URL:
    - Add a logger to print a message to indicate homeworld URL is present and will be called.
    - Add a Transform Message and use the **httpRequest** variable to set the URL homeworld path we need to use. This will replace the previous path defined.
    - Add a Flow-Reference and configure it to call common flow **common-http-request-flow**. This flow reference will use the httpRequest variable updated in the previous step. Add a Target Variable to the flow reference called **characterPlanet** to save only the name needed from the response.

        - Add a Transform Message and use it to replace the current **payload** in iteration by changing the homeworld URL with the name from the **characterPlanet** variable.
    - The **default** route: add a logger to print a message to indicate homeworld is not present.



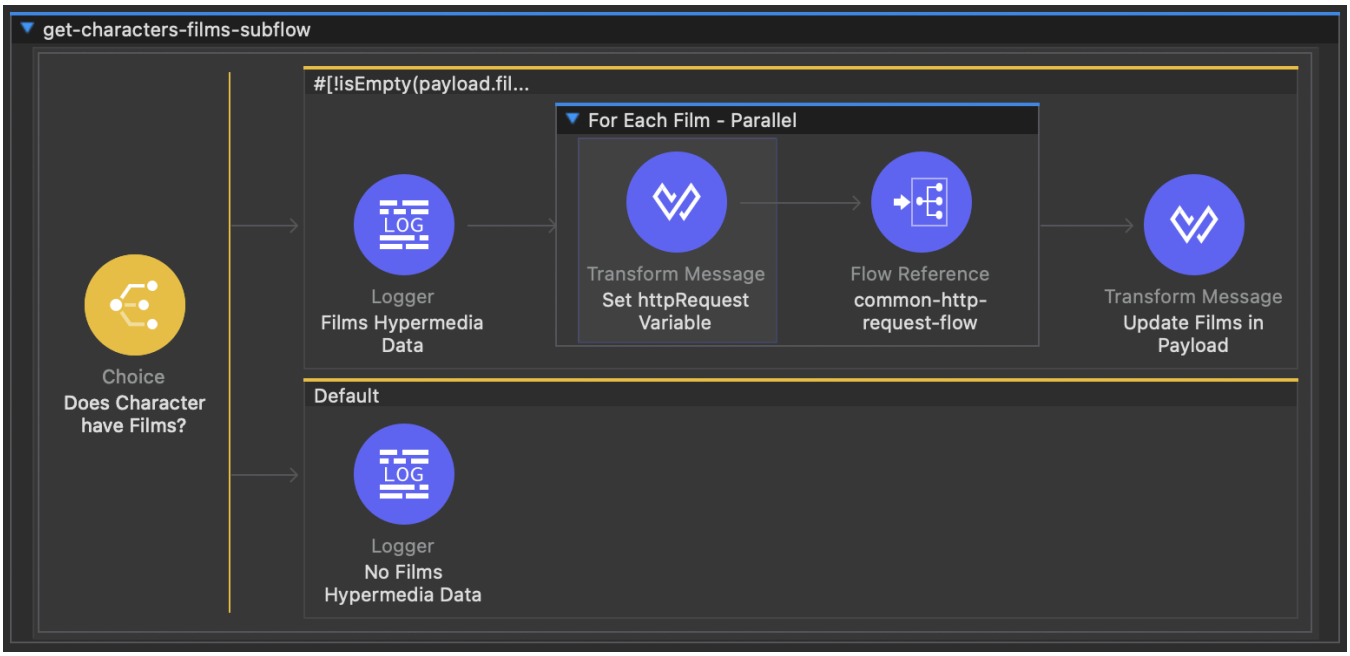
Inside **get-characters-films-subflow:**

- Add a **Choice-Router** to validate if we need to search for the character films:
    - If **payload.films** URLs array is not empty:
        - Add a logger to print a message to indicate film URLs are present and will be called.
        - Add a **Parallel For Each** to iterate **payload.films** and set a Target variable called **characterFilms** to store for each result, inside the scope we will follow the next steps:

- Add a Transform Message and use the **httpRequest** variable to set the URL film path we need to use. This will replace the previous path defined.
- Add a Flow-Reference and configure it to call common flow **common-http-request-flow**. This flow reference will use the httpRequest variable updated in the previous step.

- Add a Transform Message and use it to replace the current **payload** in iteration by changing the film URLs array with the film names from the **characterFilms** variable.

- The **default** route: add a logger to print a message to indicate films array is empty.

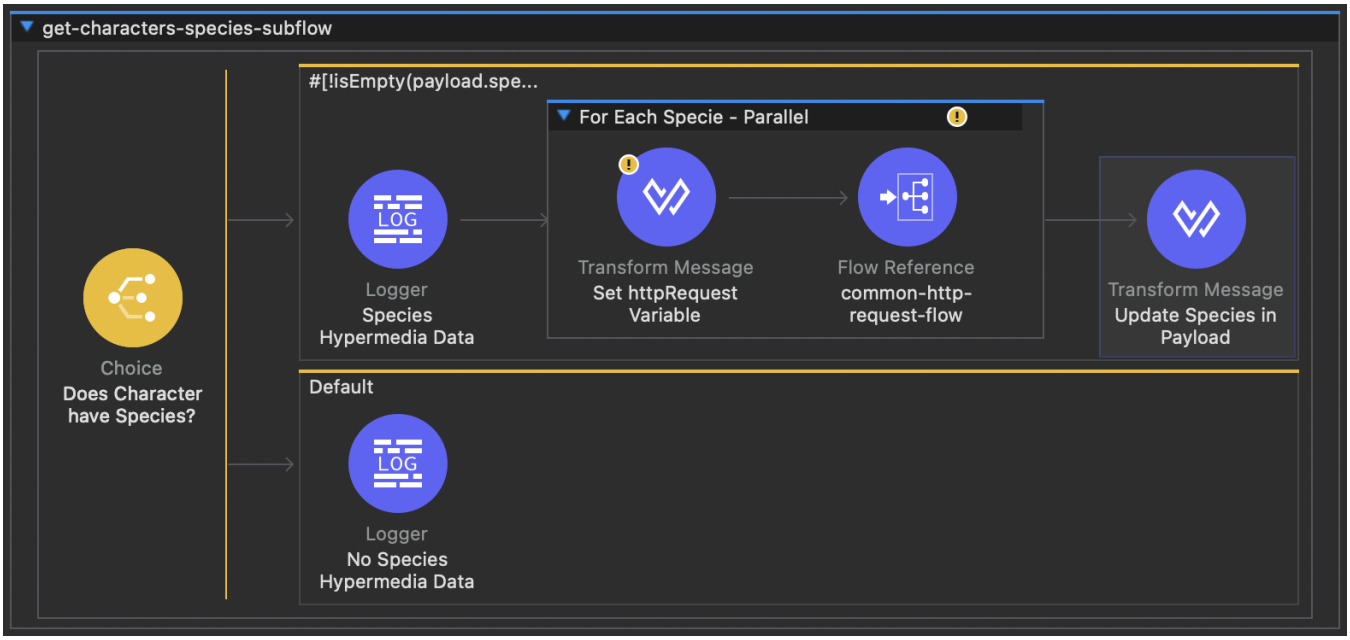


Inside **get-characters-species-subflow:**

- Add a **Choice-Router** to validate if we need to search for the character species:
  - If **payload.species** URLs array is not empty:

- Add a logger to print a message to indicate specie URLs are present and will be called.
- Add a **Parallel For Each** to iterate **payload.species** and set a Target variable called **characterSpecies** to store for each result, inside the scope we will follow the next steps:
    - Add a Transform Message and use the **httpRequest** variable to set the URL species path we need to use. This will replace the previous path defined.
    - Add a Flow-Reference and configure it to call common flow **common-http-request-flow**. This flow reference will use the httpRequest variable updated in the previous step.
- Add a Transform Message and use it to replace the current **payload** in iteration by changing the specie URLs array with the specie names from the **characterSpecies** variable.

- The **default** route: add a logger to print a message to indicate species array is empty.



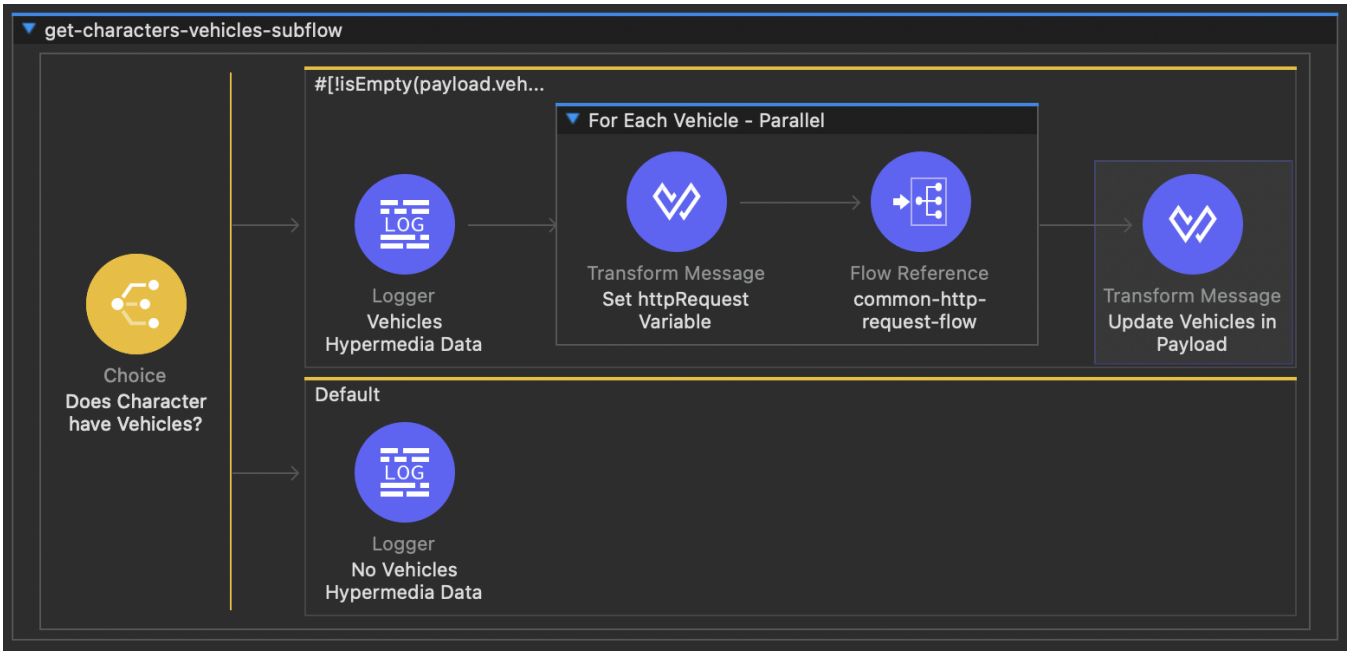
Inside **get-characters-vehicles-subflow:**

- Add a **Choice-Router** to validate if we need to search for the character vehicles:
    - If **payload.vehicles** URLs array is not empty:
        - Add a logger to print a message to indicate vehicle URLs are present and will be called.
        - Add a **Parallel For Each** to iterate **payload.vehicles** and set a Target variable called **characterVehicles** to store for each result, inside the scope we will follow the next steps:
            - Add a Transform Message and use the **httpRequest** variable to set the URL vehicle path we need to use. This will replace the previous path defined.

- Add a Flow-Reference and configure it to call common flow **common-http-request-flow.** This flow reference will use the httpRequest variable updated in the previous step.

    - Add a Transform Message and use it to replace the current **payload** in iteration by changing the vehicle URLs array with the vehicle names from the **characterVehicles** variable.

- The **default** route: add a logger to print a message to indicate the vehicle array is empty.

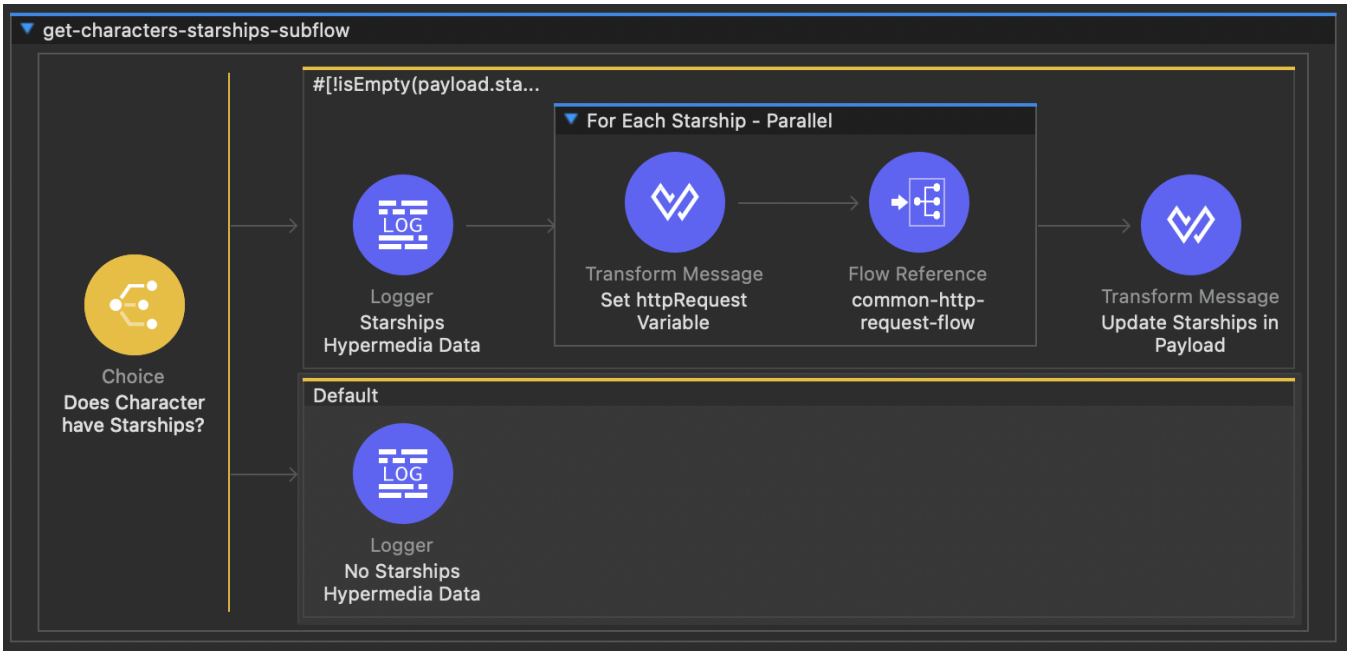


Inside **get-characters-starships-subflow:**

- Add a **Choice-Router** to validate if we need to search for the character starships:
    - If **payload.starships** URLs array is not empty:
        - Add a logger to print a message to indicate starship URLs are present and will be called.

- Add a **Parallel For Each** to iterate **payload.starships** and set a Target variable called **characterStarships** to store for each result, inside the scope we will follow the next steps:
    - Add a Transform Message and use the **httpRequest** variable to set the URL starship path we need to use. This will replace the previous path defined.
    - Add a Flow-Reference and configure it to call common flow **common-http-request-flow**. This flow reference will use the httpRequest variable updated in the previous step.
- Add a Transform Message and use it to replace the current **payload** in iteration by changing the starship URLs array with the starship names from the **characterStarships** variable.

- The **default** route: add a logger to print a message to indicate the starships array is empty.



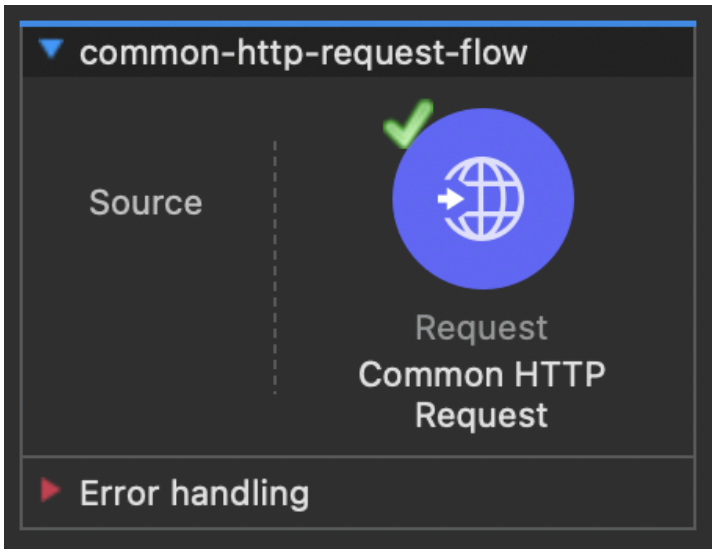
Common HTTP Request:

Create a new file **common-http-request.xml** in the commons folder(create a folder if doesn´t exist)This file will contain the following flows:

- Common-http-request-flow

Inside **common-http-request-flow:**

- Add **HTTP Request** and configure it to work dynamically using the common **httpRequest** variable.
- Create a common **HTTP Request configuration**, you can move this configuration inside the **global.xml** file.



Common Error process:

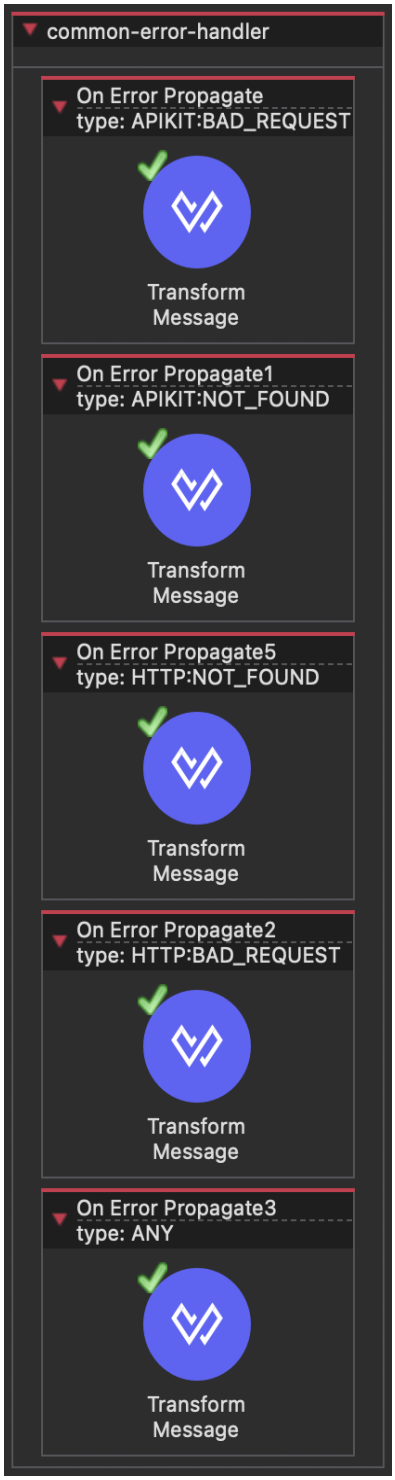
Create a **common-error-handler.xml** in the commons folder(create a folder if doesn´t exist). This file will contain the following flows:

- Common-error-handler(Error-Handler Component)

Inside **common-error-handler:**

- Set handlers for APIKit errors 400, 404:
    - For each Error Scenario, add a **On-Error-Propagate** to the **Error-handler.**

    - Add a Transform Message and set the **payload** error response.
- Set handlers for HTTP errors 400,404, and 500:
    - For each Error Scenario, add a **On-Error-Propagate** to the **Error-handler.**
    - Add a Transform Message and set the **payload** error response.


common-error-handler
On Error Propagate
type: APIKIT:BAD_REQUEST
Transform Message
On Error Propagate1
type: APIKIT:NOT_FOUND
Transform Message
On Error Propagate5
type: HTTP:NOT_FOUND
Transform Message
On Error Propagate2
type: HTTP:BAD_REQUEST
Transform Message
On Error Propagate3
type: ANY
Transform Message

SWAPI Response Details:

Response Example


```
{
 "count": 82,
 "next": "https://swapi.dev/api/people/?page=2",
 "previous": null,
 "results": [
  {
    "name": "Luke Skywalker",
    "height": "172",
    "mass": "77",
    "hair_color": "blond",
    "skin_color": "fair",
    "eye_color": "blue",
    "birth_year": "19BBY",
    "gender": "male",
    "homeworld": "https://swapi.dev/api/planets/1/",
    "films": [
      "https://swapi.dev/api/films/1/"
    ],
    "species": [
      "https://swapi.dev/api/films/1/"
    ],
    "vehicles": [
```

```
    "https://swapi.dev/api/vehicles/1/"
   ],
   "starships": [
    "https://swapi.dev/api/starships/1/"
   ],
   "created": "2014-12-09T13:50:51.644000Z",
   "edited": "2014-12-20T21:17:56.891000Z",
   "url": "https://swapi.dev/api/people/1/"
  }
 ]
}
```


Response Details Mapping - SWAPI to Mule API

Target System Field	Data Type	Mule Field	Data Type	Defaulting/ Transformation Logic
count	Integer	NA	NA	Used to get number pages
next	String	NA	NA	Used to validate pagination
previous	String	NA	NA	NA
results	Array	Payload	Array	Characters results to process.
name	String	name	String	Format to CSV
height	String	height	String	Format to CSV
mass	String	mass	String	Format to CSV
hair_color	String	hair_color	String	Format to CSV
skin_color	String	skin_color	String	Format to CSV
eye_color	String	eye_color	String	Format to CSV
birth_year	String	birth_year	String	Format to CSV

gender	String	gender	String	Format to CSV
homeworld	String	homeworld	String	Get Hypermedia and Format to CSV
films	String	films	String	Get Hypermedia and Format to CSV
species	String	species	String	Get Hypermedia and Format to CSV
vehicles	String	vehicles	String	Get Hypermedia and Format to CSV
starships	String	starships	String	Get Hypermedia and Format to CSV
created	String	created	String	Format to CSV
edited	String	edited	String	Format to CSV
url	String	url	String	Format to CSV