

Documentation of FastApi Medusa

Table of Contents

1. Description
2. Normal Endpoints
 - 1. /generate-token
 - 2. /get_paths
 - 3. /import
 - 4. /export
 - 5. /copy
 - 6. /move
 - 7. /from_old_to_new
 - 8. /delete
3. Dynamic endpoints with environments variables
 - 1. /env
 - 2. Endpoints

Description

This contribution aims to extend the functionality of Medusa by implementing a microservice using FastAPI.

The goal is to: - Make Medusa interactive and user-friendly. - Provide an API interface with Swagger documentation. - Reduce reliance on terminal commands.

This endpoint generates a Vault token by authenticating with the provided AppRole credentials. It accepts the `role_id` and `secret_id` to authenticate with Vault, and then returns an access token.

Normal endpoints

1. /generate-token

This endpoint is responsible for generating a Vault token using AppRole authentication credentials. It requires the `role_id` and `secret_id` as inputs and communicates with Vault's authentication system to obtain a client token.

Goal

This token can then be used to execute various functionalities of Medusa.

HTTP Method

POST

URL

/generate-token

Parameters

Request Body (AuthRequest)

Parameter Name	Type	Description
role_id	str	Role ID used for authentication with Vault
secret_id	str	Secret associated with the role for Vault authentication

Query Parameters

Parameter Name	Type	Description
vault_url	str	Vault URL for authentication (e.g., <code>http://vault.example.com</code>)

Example Request

Terminal

```
curl -X 'POST' \
  'http://localhost:8000/generate-token?vault_url=http%3A%2F%2Fvault%3A8201' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "role_id": "184376ff-23ba-5c76-0f9a-23cb85777e98",
    "secret_id": "0b4d0ebf-e504-40d5-9f48-14df22983c50"
  }'

{
  "role_id": "my-role-id",
  "secret_id": "my-secret-id"
}
```

Swagger

POST /generate-token Generate Token

Generates a Vault token by authenticating with the AppRole credentials provided.

Parameters

Name	Description
vault_url ^{required}	Vault URL for authentication

string (query)

Request body ^{required} application/json

```
{  "role_id": "184376ff-23ba-5c76-bf9a-23c8b777e98",  "secret_id": "8b4d8e0f-c594-4a65-9f48-54d722883c9f"}
```

Server response

Code	Details
200	<p>Response body</p> <pre>{ "token": "hvs.CAESJGKvRfTc8YufZCXMJNDhArcz7L3btpczNMIV7ecFKq4Hk43c5CDE115W0a6Q9M7N007ME5rH6Ebm"</pre> <p>Response headers</p> <pre>content-length: 187 content-type: application/json date: Tue, 31 Dec 2024 13:49:38 GMT server: wsl.com</pre>

Result :

2. /get_paths

Description

This endpoint retrieves secret paths from Vault based on the provided Vault URL, authentication token, and root secret. It interacts with the Vault server to fetch the secret paths, processes them, and returns the available paths to the user. If an error occurs during the process, a 500 HTTP error is returned with a detailed message.

Goal

The goal of this endpoint is to provide the user with all available secret paths. The user can then easily copy and paste these paths for other methods to retrieve, import, copy, move, or delete secrets, instead of manually writing them.

HTTP Method

GET

URL

/get_paths

Query Parameters

3. /import

Description

This endpoint imports files to a specified Vault path using Medusa. The method accepts multiple files, and the files are temporarily saved to the local system before being uploaded to Vault. If no files are provided, a 400 error is returned. Once the files are successfully uploaded, a message is returned with the list of imported file paths.

Goal

The goal of this endpoint is to allow users to import one or multiple files to a specific path in Vault, specifying the Vault URL, token, and engine type (vault version). This makes it easier to upload secrets, configurations, or any file-based data to Vault without manual intervention.

HTTP Method

POST

URL

/import

Query Parameters

Parameter		
Name	Type	Description
address	str	The URL of the Vault server where secrets are stored (e.g., <code>http://vault.example.com</code>).
token	str	The authentication token required to access Vault (e.g., <code>s.abc123xyz</code>).
path	str	The secret path where the files will be imported in Vault (e.g., <code>secret/my-secrets</code>).
engine_type	str	Specify the Vault engine type (kv1 or kv2). Defaults to kv2.
files	List[UploadFile]	A list of files to be uploaded.

Example Request

Terminal

```
curl -X 'POST' \
  'http://localhost:8000/import?address=http%3A%2F%2Fvault%3A8201&token=00000000-0000-0000-0000-0000-0000-0000-0000-0000'
```

```
-H 'accept: application/json' \
-H 'Content-Type: multipart/form-data' \
-F 'files=@import-example-1.json;type=application/json'
```

Swagger

POST /import Import To Vault

Imports files to a specified Vault path using Medusa. The method accepts multiple files, and the files are temporarily saved to the local system before being uploaded to Vault.

Returns:

- A message confirming successful import along with the list of imported file paths

Parameters

Name	Description
address * required string (query)	Vault URL http://vault:8201
token * required string (query)	Vault Token 00000000-0000-0000-0000-000000000000
path * required string (query)	Secret path to import secret
engine_type string (query)	Specify the Vault engine type kv2

Request body multipart/form-data

files array

Chosir un fichier import-example-1.json

Add string item

Send empty value

Server response

Code	Details
200	<p>Response body</p> <pre>{ "message": "Files imported successfully", "files": ["/tmp/import-example-1.json"] }</pre> <p>Response headers</p> <pre>content-length: 80 content-type: application/json date: Tue, 31 Dec 2024 14:42:45 GMT server: awslcm</pre>

Responses

Code	Description	Links
200	Successful Response	No links

Result :

Error Handling

If an error occurs, the server will respond with a 500 Internal Server Error and a message detailing the error.

4. /export

This endpoint is responsible for exporting secrets from a Vault instance to a file. It allows users to specify the file format (YAML or JSON), Vault engine type (kv1 or kv2), and optionally provide a custom file name for the exported file. If no file name is provided, a default name is generated based on the secret path.

Goal

This functionality is useful for exporting secrets from Vault for further use or backup in a structured file format. As a result a downloadable file will be returned in the format specified (either YAML or JSON).

HTTP Method

POST

URL

/export

Parameters

Query Parameters

Parameter		
Name	Type	Description
address	str	Vault URL for accessing the instance (e.g., <code>http://vault.example.com</code>)
token	str	Vault Token for authentication with Vault
path	str	Secret path to export from Vault
file_name	str	(Optional) Custom file name for the exported file. If not provided, a default name is used based on the secret path.
output_format	str	(Optional) The format for the exported file. Can be <code>yaml</code> or <code>json</code> . Default is <code>yaml</code> .
engine_type	str	(Optional) Vault engine type, either <code>kv1</code> or <code>kv2</code> . Default is <code>kv2</code> .

Example Request

Terminal

```
curl -X 'POST' \
  'http://localhost:8000/export?address=http%3A%2F%2Fvault%3A8201&token=00000000-0000-0000-0000-0000-0000-0000-0000-0000' \
  -H 'accept: application/json' \
  -d '' \
  -H 'Content-Type: application/json'
```

Swagger

The image shows the Swagger UI for the `POST /export` endpoint, titled "Export From Vault". The description states: "Exports secrets from Vault to a file using Medusa. The method allows for optionally, choosing the file format (YAML or JSON), and specifying the Vault engine type (kv1 or kv2). The exported file will either use the provided filename or a default one generated from the secret path."

Parameters:

- `address`: Vault URL (required, string, query). Value: `http://vault:8201`
- `token`: Vault Token (required, string, query). Value: `hvs.CAESiQoTNTQxMTRhZjZlZjc5S00ZTz`
- `path`: Secret path to export (required, string, query). Value: `secret/A`
- `file_name`: Choose file name for the exported file (string, query). Value: `name_test`
- `output_format`: Choose the output file format (string, query). Value: `yaml`
- `engine_type`: Specify Vault engine type (string, query). Value: `kv2`

Returns:

- A file response with the exported secret file in the specified format.

Server response:

Code: 200

Response body: [Download file](#)

Response headers:

```
accept-ranges: bytes
content-disposition: attachment; filename="name_test.yaml"
content-length: 221
content-type: application/octet-stream
date: Tue, 31 Dec 2024 15:04:04 GMT
etag: "bc0220b07f400b3f18d470cf900a0e"
last-modified: Tue, 31 Dec 2024 15:04:04 GMT
server: uvicorn
```

Responses:

Code	Description	Links
200	Successful Response	No links

Below the Swagger UI, a file download notification is visible in the bottom right corner:

Téléchargements

name_test (1).yaml

[Ouvrir un fichier](#)

[Afficher plus](#)

Result :

Error Handling

If an error occurs, the server will respond with a 500 Internal Server Error and a message detailing the error.

5. /copy

This endpoint allows copying secrets from one Vault path (`source_path`) to another (`target_path`). The method communicates with Vault using the provided Vault URL and token. It uses the Medusa tool to perform the copy operation. The engine type (kv1 or kv2) for Vault can also be specified. Upon successful execution, a confirmation message with source and target paths is returned.

Goal

This functionality allows users to copy secrets from one path to another within the same Vault instance, making it easier to manage and migrate secrets.

HTTP Method

POST

URL

/copy

Parameters

Query Parameters

Parameter		
Name	Type	Description
address	str	Vault URL for accessing the instance (e.g., <code>http://vault.example.com</code>)
token	str	Vault Token for authentication with Vault
source_path	str	The source secret path in Vault to be copied
target_path	str	The target secret path in Vault where the secret will be copied
engine_type	str	(Optional) Vault engine type, either <code>kv1</code> or <code>kv2</code> . Default is <code>kv2</code> .

Example Request

Terminal

```
curl -X 'POST' \
  'http://localhost:8000/copy?address=http%3A%2F%2Fvault%3A8201&token=00000000-0000-0000-0000-00000000' \
  -H 'accept: application/json' \
  -d ''
```

Swagger

POST /copy Copy Secret

This endpoint allows copying secrets from one Vault path (source_path) to another (target_path). The method communicates with Vault using the provided Vault URL and token. It uses the Medusa tool to perform the copy operation. The engine type (kv1 or kv2) for Vault can also be specified. Upon successful execution, a confirmation message with source and target paths is returned.

Parameters:

- address: The URL of the Vault instance.
- token: The Vault token for authentication.
- source_path: The path to the secret in Vault to be copied.
- target_path: The destination path in Vault where the secret will be copied.
- engine_type: The Vault engine type (kv1 or kv2).

Returns:

- A success message indicating the paths of the copied secret.

Raises:

- HTTPException: If the copy operation fails or encounters an error.

Parameters

Name	Description
address * required	Vault URL
token * required	Vault Token
source_path * required	Source secret path in Vault
target_path * required	Target secret path in Vault
engine_type	Specify Vault engine type

Execute

Clear

Server response

Code	Details
200	<div><div>Response body</div><div><pre>{ "message": "Secrets from 'secret/A' copied successfully", "source_path": "secret/A", "target_path": "B/A" }</pre></div><div>Download</div></div> <div><div>Response headers</div><div><pre>content-length: 182 content-type: application/json date: Tue, 01 Dec 2020 17:21:21 GMT server: uvicorn</pre></div></div>

Responses

Code	Description	Links
200	Successful Response	No links

Secrets / secret / B

secret version 2

Secrets

Configuration

B/

Q Search

Create secret +

AL

...

1-1 of 1

< 1 >

Result :

6. /move

This endpoint allows moving secrets from one Vault path (**source_path**) to another (**target_path**). The method communicates with Vault using the provided Vault URL and token. It uses the Medusa tool to perform the move operation. After the secret is successfully moved to the target path, it is deleted from the source path. The engine type (kv1 or kv2) for Vault can also be specified. Upon successful execution, a confirmation message with source and target paths is returned.

10

Goal

This functionality allows users to move secrets from one path to another within the same Vault instance, enabling easier secret management and cleanup.

HTTP Method

POST

URL

/move

Parameters

Query Parameters

Parameter		
Name	Type	Description
address	str	Vault URL for accessing the instance (e.g., <code>http://vault.example.com</code>)
token	str	Vault Token for authentication with Vault
source_path	str	The source secret path in Vault to be moved
target_path	str	The target secret path in Vault where the secret will be moved
engine_type	str	(Optional) Vault engine type, either <code>kv1</code> or <code>kv2</code> . Default is <code>kv2</code> .

Example Request

Terminal

```
curl -X 'POST' \
  'http://localhost:8000/move?address=http%3A%2F%2Fvault%3A8201&token=00000000-0000-0000-0000-00000000' \
  -H 'accept: application/json' \
  -d ''
```

Swagger

POST

/move: Move Secret

This endpoint allows moving secrets from one Vault path (source_path) to another (target_path). The method communicates with Vault using the provided Vault URL and token. It uses the Medusa tool to perform the move operation. After the secret is successfully moved to the target path, it is deleted from the source path. The engine type (kv1 or kv2) for Vault can also be specified. Upon successful execution, a confirmation message with source and target paths is returned.

Parameters:

- address: The URL of the Vault instance.
- token: The Vault token for authentication.
- source_path: The path to the secret in Vault to be moved.
- target_path: The destination path in Vault where the secret will be moved.
- engine_type: The Vault engine type (kv1 or kv2).

Returns:

- A success message indicating the paths of the moved secret.

Raises:

- HTTPException: If the move operation fails or encounters an error.

Parameters

Cancel

Name	Description
address * required string (query)	Vault URL http://vault:8201
token * required string (query)	Vault Token 00000000-0000-0000-0000-000000000000
source_path * required string (query)	Source secret path in Vault secret/A
target_path * required string (query)	Target secret path in Vault Z/A
engine_type string (query)	Specify Vault engine type kv2

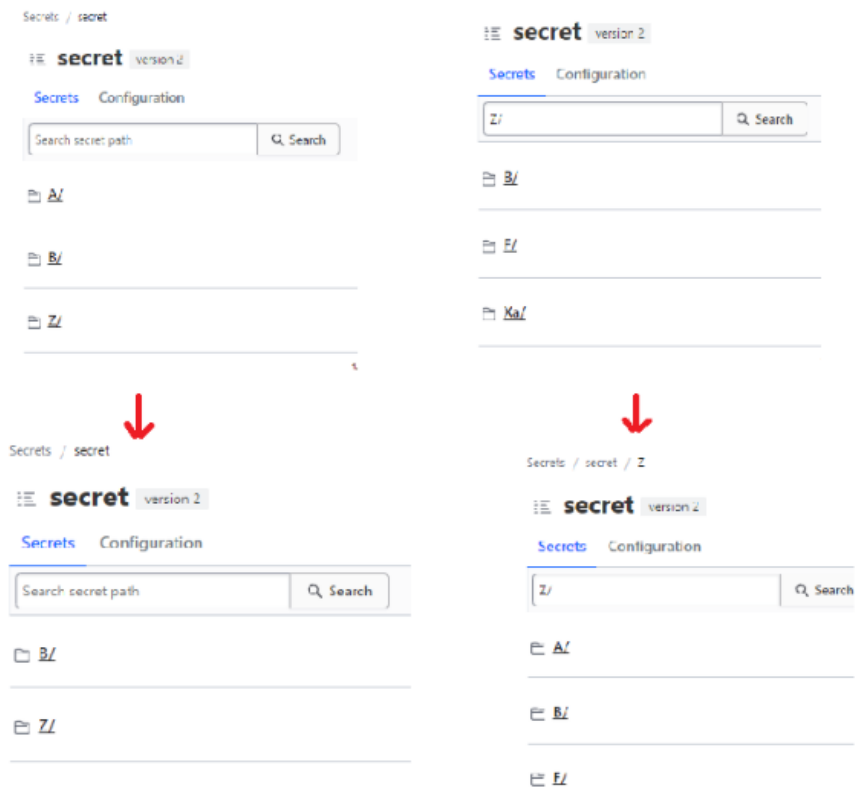
Execute

Clear

Result :

Server response		
Code	Details	
200	<div><div>Response body</div><div><pre>{ "message": "Secrets from 'secret/A' moved successfully", "source_path": "secret/A", "target_path": "Z/A" }</pre></div><div><div>Download</div></div></div> <div><div>Response headers</div><div><pre>content-length: 161 content-type: application/json date: Tue, 31 Dec 2024 17:47:19 GMT server: waitress</pre></div></div>	
Responses		
Code	Description	Links
200	Successful Response	No links

12



7. /from_old_to_new

Description

This endpoint facilitates the migration of secrets from an old Vault instance to a new Vault instance. It uses the Medusa tool to export secrets from the old Vault and imports them into the new Vault. The migration ensures secure transfer of secrets.

Goal

The purpose of this endpoint is to automate the migration of secrets between Vault instances, reducing manual effort and minimizing errors. Users can seamlessly move secrets from an outdated or deprecated Vault instance to a new one.

HTTP Method

POST

URL

/from_old_to_new

Query Parameters

Parameter		
Name	Type	Description
old_address	str	The URL of the old Vault instance (e.g., <code>http://old-vault.example.com</code>).
old_token	str	The authentication token for the old Vault instance.
old_path	str	The secret path in the old Vault instance to be migrated (e.g., <code>secret/my-old-secrets</code>).
old_engine_type	str	The engine type of the old Vault (<code>kv1</code> or <code>kv2</code>). Defaults to <code>kv2</code> .
new_address	str	The URL of the new Vault instance (e.g., <code>http://new-vault.example.com</code>).
new_token	str	The authentication token for the new Vault instance.
new_path	str	The secret path in the new Vault instance where secrets will be stored (e.g., <code>secret/my-new-secrets</code>).
engine_type	str	The engine type of the new Vault (<code>kv1</code> or <code>kv2</code>). Defaults to <code>kv2</code> .

Example Request

```
curl -X 'POST' \
  'http://localhost:8000/from_old_to_new?old_address=http%3A%2F%2Fvault%3A8201&old_token=00' \
  -H 'accept: application/json' \
  -d ''
```

Swagger

POST /from_old_to_new Migrate

Parameters

Cancel

Name	Description
old_address * required string (query)	Old Vault URL <input type="text" value="http://vault:8201"/>
old_token * required string (query)	Vault Token <input type="text" value="00000000-0000-0000-0000-000000000000"/>
old_path * required string (query)	Secret path <input type="text" value="secret"/>
old_engine_type string (query)	Choose file format <input type="text" value="kv2"/>
new_address * required string (query)	New Vault URL <input type="text" value="http://vault3:8203"/>
new_token * required string (query)	Vault Token <input type="text" value="00000000-0000-0000-0000-000000000000"/>
new_path * required string (query)	Secret path <input type="text" value="secret/test_moving"/>
engine_type string (query)	Choose file format <input type="text" value="kv2"/>
decrypt boolean (query)	Choose file format <input type="text" value="false"/>

Execute

Clear

Server response

Code	Details
200	<div><div>Response body</div><div><pre>{ "message": "Secrets from 'http://vault:8201' copied successfully to 'http://vault3:8203'" }</pre></div><div><div>Download</div></div></div> <div><div>Response headers</div><div><pre>content-length: 96 content-type: application/json date: Tue, 01 Dec 2020 18:29:26 GMT server: nginx</pre></div></div>

Responses

Code	Description	Links
200	Successful Response	No links

Result :

8. /delete

This endpoint deletes a secret from a HashiCorp Vault instance. It uses the Medusa CLI tool to perform the deletion of the specified secret at the given path in the Vault. The operation is automatically approved and executed with the provided Vault address and authentication token.

Goal

This functionality is designed to remove secrets from a Vault instance, ensuring sensitive data can be securely deleted when it is no longer needed.

HTTP Method

DELETE

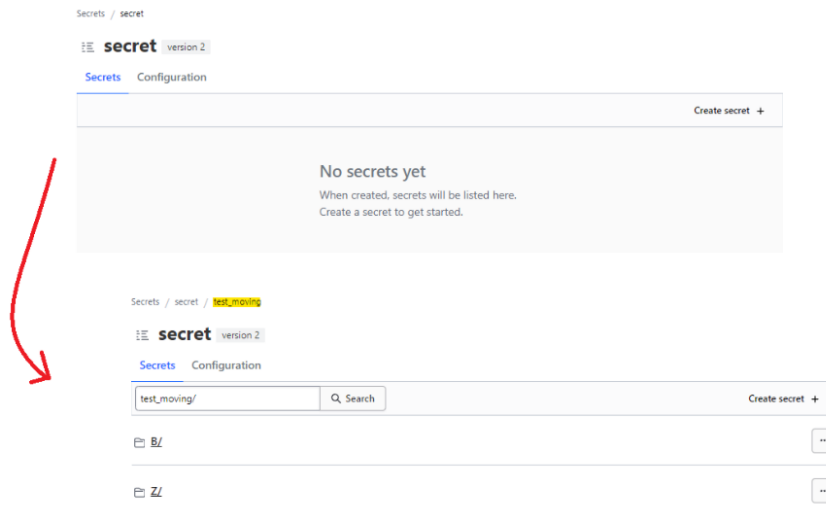


Figure 1: alt text

URL

/delete

Parameters

Query Parameters

Parameter		
Name	Type	Description
address	str	Vault URL for accessing the instance (e.g., <code>http://vault.example.com</code>)
token	str	Vault Token for authentication with Vault
secret_path	str	The path of the secret in Vault to be deleted

Example Request

Terminal

```
curl -X 'DELETE' \
'http://localhost:8000/delete?address=http%3A%2F%2Fvault%3A8201&token=00000000-0000-0000-0000-0000-0000-0000-0000-0000' \
-H 'accept: application/json'
```


Swagger

The image shows the Swagger UI for the `DELETE /delete` endpoint. The endpoint description states it deletes a secret from a HashiCorp Vault instance using the Medusa CLI tool. The parameters section lists three required query parameters: `address` (Vault URL, value: `http://vault:8201`), `token` (Vault Token, value: `00000000-0000-0000-0000-000000000000`), and `secret_path` (Secret path, value: `secret/B`). The server response section shows a 200 status code with a JSON body: `{ "message": "Secret deleted successfully", "secret_path": "secret/B" }`. Below the response, a table lists the response details: Code (200), Description (Successful Response), and Links (No links).

Result :

The diagram illustrates the transition from a static secret path to a dynamic one. On the left, a search bar labeled "secret" (version 2) has a "Secrets" tab selected, and the "Search secret path" field contains the static path `secret/B`. A red arrow points to the right, where the same search bar is shown, but the "Search secret path" field now contains the dynamic path `secret/B`, indicating the use of environment variables.

Dynamic Endpoints with Environment Variables

To make endpoints more dynamic and avoid repeatedly entering the URL, token, and engine type, you can now create these as environment variables using the `/env` endpoint.

`/env`

The `/env` endpoint takes the following parameters: - **Vault URL**: The address of your Vault instance. - **Token**: Your authentication token. - **Engine Type**: The type of engine you are working with.

POST

/advanced_export

Export From Vault

^

This endpoint exports secrets from a HashiCorp Vault instance to a file.

Parameters:

- file_name: The name of the file to save the secrets. Defaults to a name derived from the secret path.
- output_format: The format of the output file (yaml or json).
- secret_path: The path of the secret in Vault to export.
- engine_type: The type of engine (v1 or kv2) used in Vault.

Returns:

- The exported file as a downloadable response.

Raises:

- HTTPException: If the Medusa CLI export command fails or the file is not created.

Parameters

Cancel

Name	Description
file_name string (query)	Choose file name <input type="text" value="file_name"/>
output_format string (query)	Choose file format <div>yaml</div>
secret_path * required string (query)	Secret path <input type="text" value="secret_path"/>
engine_type string (query)	Choose file format <div>kv2</div>

Execute

POST

/advanced_import

Import From Vault

^

This endpoint imports secrets into a HashiCorp Vault instance from one or more files.

Parameters:

- path: The Vault secret path where the secrets will be imported.
- files: A list of files containing secrets to import.

The files are temporarily stored in [/tmp](#) and then imported into Vault using the Medusa CLI.

Returns:

- A success message with the list of imported file paths.

Raises:

- HTTPException: If no files are provided, the Medusa CLI import command fails, or other errors occur during the import process.

Parameters

Cancel

Name	Description
path * required string (query)	Secret path to import <input type="text" value="path"/>

Request body

multipart/form-data

files

Add string item

☒ Send empty value

Execute

- Import

POST

/advanced_copy

Copy From Vault

^

Parameters

Cancel

Name	Description
source_path * required string (query)	Source secret path <input type="text" value="source_path"/>
target_path * required string (query)	Target secret path <input type="text" value="target_path"/>

Execute

- Copy

POST /advanced_move Move From Vault

Parameters

Name	Description
source_path * required string (query)	Source secret path <input type="text" value="source_path"/>
target_path * required string (query)	Target secret path <input type="text" value="target_path"/>

Execute

DELETE /advanced_delete Delete From Vault

Parameters

Name	Description
secret_path * required string (query)	Secret path to delete <input type="text" value="secret_path"/>

Execute

- Move

- Delete

These advanced endpoints are built on top of the standard ones but are more dynamic, removing the need to repeatedly input parameters manually.

Docker Setup for Running Medusa and FastAPI Container

To run the container that includes both Medusa and FastAPI, follow these steps:

Note: The Medusa client is already built into the container, so you can directly use the CLI through the container's terminal.

1. Prerequisites:

Make sure you have Docker installed on your machine.

2. Build the Image

To build the Medusa container image, use the following command:

```
1- cd medusa/api
2- docker build -t medusa-api -f Dockerfile ../
```

3. Run the Medusa Container

To run the container, use the following command:

```
docker run -d -p 8000:8000 -p 8080:8080 --name medusa-app --network vault medusa-api
```

The Dockerfile ensures that Python, Go, and all the necessary dependencies are installed. It also builds the Medusa application and sets FastAPI as the entry point to make the application ready for use.

4. Accessing the Swagger UI

Once the container is running, you can access the Swagger UI for executing the endpoints at:

`http://localhost:8000/docs`