



© 2024 ANSYS, Inc. or affiliated companies
Unauthorized use, distribution, or duplication prohibited.

Allie Flowkit Python



ANSYS, Inc.
Southpointe
2600 Ansys Drive
Canonsburg, PA 15317
ansysinfo@ansys.com
<http://www.ansys.com>
(T) 724-746-3304
(F) 724-514-9494

Sep 30, 2024

ANSYS, Inc. and
ANSYS Europe,
Ltd. are UL
registered ISO
9001:2015
companies.

CONTENTS

1	Getting started	3
1.1	Installation	3
1.1.1	Quick start	3
2	User guide	5
3	API reference	7
3.1	The <code>src.allie.flowkit</code> library	7
3.1.1	Summary	7
3.1.2	Description	20
3.1.3	Module detail	21
4	Examples	23
5	Contribute	25
5.1	Clone the repository	25
5.2	Adhere to code style	25
5.3	Run the tests	26
5.4	Build the documentation	26
	Python Module Index	27
	Index	29

The Allie Flowkit Python is a Python service that exposes features from Allie Flowkit to Python users. This documentation provides information on how to install and use the Allie Flowkit Python.

The Allie Flowkit Python offers these main features:

Getting started Learn how to install the Allie Flowkit Python in user mode and quickly begin using it.

Getting started **User guide** Understand key concepts for implementing the Allie Flowkit Python in your workflow.

User guide **API reference** Understand how to use Python to interact programmatically with the Allie Flowkit Python.

The src.allie.flowkit library **Examples** Explore examples that show how to use the Allie Flowkit Python to perform many different types of operations.

Examples **Contribute** Learn how to contribute to the Allie Flowkit Python codebase or documentation.

Contribute

GETTING STARTED

This section describes how to install the Allie Flowkit Python in user mode and quickly begin using it. If you are interested in contributing to the Allie Flowkit Python, see [Contribute](#) for information on installing in developer mode.

1.1 Installation

To use `pip` to install the Allie Flowkit Python, run this command:

```
pip install allie-flowkit-python
```

Alternatively, to install the latest version from this library's [GitHub repository](#), run these commands:

```
git clone https://github.com/ansys/allie-flowkit-python
cd allie-flowkit-python
pip install .
```

1.1.1 Quick start

The following examples show how to use the Allie Flowkit Python.

```
allie-flowkit-python --host 0.0.0.0 --port 50052 --workers 1
```


USER GUIDE

This section explains key concepts for implementing the Allie Flowkit Python in your workflow. You can use the Allie Flowkit Python in your examples as well as integrate this library into your own code.

API REFERENCE

This section describes Allie Flowkit Python endpoints, their capabilities, and how to interact with them programmatically.

3.1 The `src.allie.flowkit` library

3.1.1 Summary

Subpackages

<code>config</code>	Configuration package for the application.
<code>endpoints</code>	Endpoints package responsible for defining the endpoints.
<code>models</code>	Models package used to define the data models.
<code>utils</code>	Utils module.

Submodules

<code>__main__</code>	Main module for the FlowKit service.
<code>fastapi_utils</code>	Utils module for FastAPI related operations.
<code>flowkit_service</code>	Module for the Allie Flowkit service.

Attributes

<code>__version__</code>

The config package

Summary

Description

Configuration package for the application.

The endpoints package

Summary

Submodules

<i>splitter</i>	Module for splitting text into chunks.
-----------------	--

The splitter.py module

Summary

Functions

<i>split_ppt</i>	Endpoint for splitting text in a PowerPoint document into chunks.
<i>split_py</i>	Endpoint for splitting Python code into chunks.
<i>split_pdf</i>	Endpoint for splitting text in a PDF document into chunks.
<i>process_ppt</i>	Process a PowerPoint document to split text into chunks.
<i>process_python_code</i>	Process Python code to split text into chunks.
<i>process_pdf</i>	Process a PDF document to split text into chunks.
<i>validate_request</i>	Validate the splitter request and API key.

Attributes

<i>router</i>

Constants

<i>TOKEN_TO_CHARACTER_MULTIPLIER</i>

Description

Module for splitting text into chunks.

Module detail

async `splitter.split_ppt`(*request*: allie.flowkit.models.splitter.SplitterRequest, *api_key*: *str* = Header(...)) → *allie.flowkit.models.splitter.SplitterResponse*

Endpoint for splitting text in a PowerPoint document into chunks.

Parameters

request

[SplitterRequest] An object containing 'document_content' in Base64, 'chunk_size', and 'chunk_overlap'

api_key

[str] The API key for authentication.

async `splitter.split_py`(*request*: allie.flowkit.models.splitter.SplitterRequest, *api_key*: *str* = Header(...)) → *allie.flowkit.models.splitter.SplitterResponse*

Endpoint for splitting Python code into chunks.

Parameters

request

[SplitterRequest] An object containing 'document_content' in Base64, 'chunk_size', and 'chunk_overlap'

api_key

[str] The API key for authentication.

Returns

SplitterResponse

An object containing a list of text chunks.

async `splitter.split_pdf`(*request*: allie.flowkit.models.splitter.SplitterRequest, *api_key*: *str* = Header(...)) → *allie.flowkit.models.splitter.SplitterResponse*

Endpoint for splitting text in a PDF document into chunks.

Parameters

request

[SplitterRequest] An object containing 'document_content' in Base64, 'chunk_size', and 'chunk_overlap'.

api_key

[str] The API key for authentication.

Returns

SplitterResponse

An object containing a list of text chunks.

`splitter.process_ppt`(*request*: allie.flowkit.models.splitter.SplitterRequest) → *allie.flowkit.models.splitter.SplitterResponse*

Process a PowerPoint document to split text into chunks.

Parameters

request

[SplitterRequest] An object containing 'document_content' in Base64, 'chunk_size', and 'chunk_overlap'

Returns

SplitterResponse

An object containing a list of text chunks.

`splitter.process_python_code(request: allie.flowkit.models.splitter.SplitterRequest) → allie.flowkit.models.splitter.SplitterResponse`

Process Python code to split text into chunks.

Parameters

request

[SplitterRequest] An object containing 'document_content' in Base64, 'chunk_size', and 'chunk_overlap'

Returns

SplitterResponse

An object containing a list of text chunks.

`splitter.process_pdf(request: allie.flowkit.models.splitter.SplitterRequest) → allie.flowkit.models.splitter.SplitterResponse`

Process a PDF document to split text into chunks.

Parameters

request

[SplitterRequest] An object containing 'document_content' in Base64, 'chunk_size', and 'chunk_overlap'

Returns

SplitterResponse

An object containing a list of text chunks.

`splitter.validate_request(request: allie.flowkit.models.splitter.SplitterRequest, api_key: str)`

Validate the splitter request and API key.

Parameters

request

[SplitterRequest] An object containing 'document_content' in Base64, 'chunk_size', and 'chunk_overlap'

api_key

[str] The API key for authentication.

Raises

HTTPException

If the API key is invalid or if any of the request parameters are invalid.

`splitter.TOKEN_TO_CHARACTER_MULTIPLIER = 4`

`splitter.router`

Description

Endpoints package responsible for defining the endpoints.

The models package

Summary

Submodules

<i>functions</i>	Module for defining the models used in the endpoints.
<i>splitter</i>	Model for the splitter endpoint.

The functions.py module

Summary

Classes

<i>ParameterInfo</i>	Parameter information model.
<i>EndpointInfo</i>	Endpoint information model.

Enums

<i>FunctionCategory</i>	Enum for function categories.
-------------------------	-------------------------------

ParameterInfo

```
class src.allie.flowkit.models.functions.ParameterInfo(/, **data: Any)
```

Bases: pydantic.BaseModel

Parameter information model.

Parameters

BaseModel

[pydantic.BaseModel] The base model for the parameter information

Overview

Attributes

<i>name</i>
<i>type</i>

Import detail

```
from src.allie.flowkit.models.functions import ParameterInfo
```

Attribute detail

ParameterInfo.**name**: `str`

ParameterInfo.**type**: `str`

EndpointInfo

class src.allie.flowkit.models.functions.**EndpointInfo**(/, ***data: Any*)

Bases: pydantic.BaseModel

Endpoint information model.

Parameters

BaseModel

[pydantic.BaseModel] The base model for the endpoint information

Overview

Attributes

<i>name</i>
<i>path</i>
<i>category</i>
<i>display_name</i>
<i>description</i>
<i>inputs</i>
<i>outputs</i>
<i>definitions</i>

Import detail

```
from src.allie.flowkit.models.functions import EndpointInfo
```

Attribute detail

```
EndpointInfo.name: str
EndpointInfo.path: str
EndpointInfo.category: str
EndpointInfo.display_name: str
EndpointInfo.description: str
EndpointInfo.inputs: list[ParameterInfo]
EndpointInfo.outputs: list[ParameterInfo]
EndpointInfo.definitions: dict[str, Any]
```

FunctionCategory

```
class src.allie.flowkit.models.functions.FunctionCategory
    Bases: enum.Enum
    Enum for function categories.
```

Overview

Attributes

<i>DATA_EXTRACTION</i>
<i>GENERIC</i>
<i>KNOWLEDGE_DB</i>
<i>LLM_HANDLER</i>
<i>ANSYS_GPT</i>

Import detail

```
from src.allie.flowkit.models.functions import FunctionCategory
```


Attribute detail

FunctionCategory.DATA_EXTRACTION = 'data_extraction'

FunctionCategory.GENERIC = 'generic'

FunctionCategory.KNOWLEDGE_DB = 'knowledge_db'

FunctionCategory.LLM_HANDLER = 'llm_handler'

FunctionCategory.ANSYS_GPT = 'ansys_gpt'

Description

Module for defining the models used in the endpoints.

The `splitter.py` module

Summary

Classes

<code>SplitterRequest</code>	Request model for the splitter endpoint.
<code>SplitterResponse</code>	Response model for the splitter endpoint.

SplitterRequest

class src.allie.flowkit.models.splitter.**SplitterRequest**(/, ****data**: Any)

Bases: pydantic.BaseModel

Request model for the splitter endpoint.

Parameters

BaseModel

[pydantic.BaseModel] The base model for the request.

Overview

Attributes

<code>document_content</code>
<code>chunk_size</code>
<code>chunk_overlap</code>

Import detail

```
from src.allie.flowkit.models.splitter import SplitterRequest
```

Attribute detail

SplitterRequest.**document_content**: bytes

SplitterRequest.**chunk_size**: int

SplitterRequest.**chunk_overlap**: int

SplitterResponse

```
class src.allie.flowkit.models.splitter.SplitterResponse(/, **data: Any)
```

Bases: pydantic.BaseModel

Response model for the splitter endpoint.

Parameters

BaseModel

[pydantic.BaseModel] The base model for the response.

Overview

Attributes

chunks

Import detail

```
from src.allie.flowkit.models.splitter import SplitterResponse
```

Attribute detail

SplitterResponse.**chunks**: list[str]

Description

Model for the splitter endpoint.

Description

Models package used to define the data models.

The utils package

Summary

Submodules

<i>decorators</i>	Decorators module for function definitions.
-------------------	---

The decorators.py module

Summary

Functions

<i>category</i>	Decorator to add a category to the function.
<i>display_name</i>	Decorator to add a display name to the function.

Description

Decorators module for function definitions.

Module detail

`decorators.category`(*value: str*)

Decorator to add a category to the function.

`decorators.display_name`(*value: str*)

Decorator to add a display name to the function.

Description

Utils module.

The `__main__.py` module

Summary

Functions

<code>parse_cli_args</code>	Parse the command line arguments.
<code>substitute_empty_values</code>	Substitute the empty values with configuration values.
<code>main</code>	Run entrypoint for the FlowKit service.

Description

Main module for the FlowKit service.

Module detail

`__main__.parse_cli_args()`

Parse the command line arguments.

`__main__.substitute_empty_values(args)`

Substitute the empty values with configuration values.

`__main__.main()`

Run entrypoint for the FlowKit service.

The `fastapi_utils.py` module

Summary

Functions

<code>extract_field_type</code>	Extract the field type from a given schema field information.
<code>extract_fields_from_schema</code>	Extract fields and their types from a schema.
<code>get_parameters_info</code>	Get parameter information from function parameters.
<code>get_return_type_info</code>	Get return type information from the function's return type.
<code>extract_definitions_from_schema</code>	Extract definitions from a schema.
<code>get_definitions_from_params</code>	Get definitions from function parameters.
<code>get_definitions_from_return_type</code>	Get definitions from the function's return type.
<code>extract_endpoint_info</code>	Extract endpoint information from the given routes.

Description

Utils module for FastAPI related operations.

Module detail

`fastapi_utils.extract_field_type(field_info: dict)`

Extract the field type from a given schema field information.

Parameters

field_info

[dict] The field information from the schema.

Returns

str

The extracted field type.

`fastapi_utils.extract_fields_from_schema(schema: dict)`

Extract fields and their types from a schema.

Parameters

schema

[dict] The schema dictionary.

Returns

list

A list of ParameterInfo objects representing the fields.

`fastapi_utils.get_parameters_info(params: dict)`

Get parameter information from function parameters.

Parameters

params

[dict] A dictionary of function parameters.

Returns

list

A list of ParameterInfo objects representing the parameters.

`fastapi_utils.get_return_type_info(return_type: type[pydantic.BaseModel])`

Get return type information from the function's return type.

Parameters

return_type

[type[BaseModel]] The return type of the function.

Returns

list

A list of ParameterInfo objects representing the return type fields.

`fastapi_utils.extract_definitions_from_schema(schema: dict) → dict[str, Any]`

Extract definitions from a schema.

Parameters

schema`[dict]` The schema dictionary.**Returns****dict**

A dictionary of definitions.

`fastapi_utils.get_definitions_from_params(params: dict) → dict[str, Any]`

Get definitions from function parameters.

Parameters**params**`[dict]` A dictionary of function parameters.**Returns****dict**

A dictionary of definitions extracted from the parameters.

`fastapi_utils.get_definitions_from_return_type(return_type: type[pydantic.BaseModel]) → dict[str, Any]`

Get definitions from the function's return type.

Parameters**return_type**`[type[BaseModel]]` The return type of the function.**Returns****dict**

A dictionary of definitions extracted from the return type.

`fastapi_utils.extract_endpoint_info(function_map: dict[str, Any], routes: list[fastapi.routing.APIRoute]) → list[allie.flowkit.models.functions.EndpointInfo]`

Extract endpoint information from the given routes.

Parameters**function_map**`[dict[str, Any]]` A dictionary mapping function names to their implementations.**routes**`[list[APIRoute]]` A list of APIRoute objects representing the API routes.**Returns****list**

A list of EndpointInfo objects representing the endpoints.

The flowkit_service.py module

Summary

Functions

<code>list_functions</code>	List all available functions and their endpoints.
-----------------------------	---

Attributes

<code>flowkit_service</code>
<code>function_map</code>

Description

Module for the Allie Flowkit service.

Module detail

`async flowkit_service.list_functions(api_key: str = Header(...)) → list[allie.flowkit.models.functions.EndpointInfo]`

List all available functions and their endpoints.

Parameters

api_key
[str] The API key for authentication.

Returns

`list[EndpointInfo]`
A list of EndpointInfo objects representing the endpoints.

`flowkit_service.flowkit_service`

`flowkit_service.function_map`

3.1.2 Description

App package responsible for creating the FastAPI app.

3.1.3 Module detail

`flowkit.__version__`

EXAMPLES

This section show how to use the Allie Flowkit Python to perform many different types of operations.

CONTRIBUTE

Overall guidance on contributing to a PyAnsys library appears in the [Contributing](#) topic in the *PyAnsys developer's guide*. Ensure that you are thoroughly familiar with this guide before attempting to contribute to the Allie Flowkit Python.

The following contribution information is specific to the Allie Flowkit Python.

5.1 Clone the repository

To clone and install the latest *Allie Flowkit Python* release in development mode, run these commands:

```
git clone https://github.com/ansys/allie-flowkit-python/  
cd allie-flowkit-python  
python -m pip install --upgrade pip  
pip install -e .
```

5.2 Adhere to code style

Allie Flowkit Python follows the PEP8 standard as outlined in PEP 8 in the PyAnsys Developer's Guide and implements style checking using pre-commit.

To ensure your code meets minimum code styling standards, run these commands:

```
pip install pre-commit  
pre-commit run --all-files
```

You can also install this as a pre-commit hook by running this command:

```
pre-commit install
```

5.3 Run the tests

Prior to running the tests, you must run this command to install the test dependencies:

```
pip install -e .[tests]
```

To run the tests, navigate to the root directory of the repository and run this command:

```
pytest
```

5.4 Build the documentation

Prior to building the documentation, you must run this command to install the documentation dependencies:

```
pip install -e .[doc]
```

To build the documentation, run the following commands:

```
cd doc  
  
# On linux  
make html  
  
# On windows  
./make.bat html
```

The documentation is built in the *docs/_build/html* directory.

PYTHON MODULE INDEX

S

- src.allie.flowkit, 7
- src.allie.flowkit.__main__, 17
- src.allie.flowkit.config, 8
- src.allie.flowkit.endpoints, 8
- src.allie.flowkit.endpoints.splitter, 8
- src.allie.flowkit.fastapi_utils, 17
- src.allie.flowkit.flowkit_service, 20
- src.allie.flowkit.models, 11
- src.allie.flowkit.models.functions, 11
- src.allie.flowkit.models.splitter, 14
- src.allie.flowkit.utils, 16
- src.allie.flowkit.utils.decorators, 16

Symbols

`__version__` (in module *flowkit*), 21

A

`ANSYS_GPT` (in module *FunctionCategory*), 14

C

`category` (in module *EndpointInfo*), 13
`category()` (in module *decorators*), 16
`chunk_overlap` (in module *SplitterRequest*), 15
`chunk_size` (in module *SplitterRequest*), 15
`chunks` (in module *SplitterResponse*), 15

D

`DATA_EXTRACTION` (in module *FunctionCategory*), 14
`definitions` (in module *EndpointInfo*), 13
`description` (in module *EndpointInfo*), 13
`display_name` (in module *EndpointInfo*), 13
`display_name()` (in module *decorators*), 16
`document_content` (in module *SplitterRequest*), 15

E

`extract_definitions_from_schema()` (in module *fastapi_utils*), 18
`extract_endpoint_info()` (in module *fastapi_utils*), 19
`extract_field_type()` (in module *fastapi_utils*), 18
`extract_fields_from_schema()` (in module *fastapi_utils*), 18

F

`flowkit_service` (in module *flowkit_service*), 20
`function_map` (in module *flowkit_service*), 20

G

`GENERIC` (in module *FunctionCategory*), 14
`get_definitions_from_params()` (in module *fastapi_utils*), 19
`get_definitions_from_return_type()` (in module *fastapi_utils*), 19
`get_parameters_info()` (in module *fastapi_utils*), 18

`get_return_type_info()` (in module *fastapi_utils*), 18

I

`inputs` (in module *EndpointInfo*), 13

K

`KNOWLEDGE_DB` (in module *FunctionCategory*), 14

L

`list_functions()` (in module *flowkit_service*), 20
`LLM_HANDLER` (in module *FunctionCategory*), 14

M

`main()` (in module *__main__*), 17
 module
 `src.allie.flowkit`, 7
 `src.allie.flowkit.__main__`, 17
 `src.allie.flowkit.config`, 8
 `src.allie.flowkit.endpoints`, 8
 `src.allie.flowkit.endpoints.splitter`, 8
 `src.allie.flowkit.fastapi_utils`, 17
 `src.allie.flowkit.flowkit_service`, 20
 `src.allie.flowkit.models`, 11
 `src.allie.flowkit.models.functions`, 11
 `src.allie.flowkit.models.splitter`, 14
 `src.allie.flowkit.utils`, 16
 `src.allie.flowkit.utils.decorators`, 16

N

`name` (in module *EndpointInfo*), 13
`name` (in module *ParameterInfo*), 12

O

`outputs` (in module *EndpointInfo*), 13

P

`parse_cli_args()` (in module *__main__*), 17
`path` (in module *EndpointInfo*), 13
`process_pdf()` (in module *splitter*), 10
`process_ppt()` (in module *splitter*), 9
`process_python_code()` (in module *splitter*), 10

R

router (*in module splitter*), 10

S

split_pdf() (*in module splitter*), 9

split_ppt() (*in module splitter*), 9

split_py() (*in module splitter*), 9

src.allie.flowkit
module, 7

src.allie.flowkit.__main__
module, 17

src.allie.flowkit.config
module, 8

src.allie.flowkit.endpoints
module, 8

src.allie.flowkit.endpoints.splitter
module, 8

src.allie.flowkit.fastapi_utils
module, 17

src.allie.flowkit.flowkit_service
module, 20

src.allie.flowkit.models
module, 11

src.allie.flowkit.models.functions
module, 11

src.allie.flowkit.models.functions.EndpointInfo
(*built-in class*), 12

src.allie.flowkit.models.functions.FunctionCategory
(*built-in class*), 13

src.allie.flowkit.models.functions.ParameterInfo
(*built-in class*), 11

src.allie.flowkit.models.splitter
module, 14

src.allie.flowkit.models.splitter.SplitterRequest
(*built-in class*), 14

src.allie.flowkit.models.splitter.SplitterResponse
(*built-in class*), 15

src.allie.flowkit.utils
module, 16

src.allie.flowkit.utils.decorators
module, 16

substitute_empty_values() (*in module __main__*),
17

T

TOKEN_TO_CHARACTER_MULTIPLIER (*in module splitter*), 10

type (*in module ParameterInfo*), 12

V

validate_request() (*in module splitter*), 10