



© 2025 ANSYS, Inc. or affiliated companies  
Unauthorized use, distribution, or duplication prohibited.

## Aali Flowkit Python

---



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

Jul 25, 2025

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 `src.aali.flowkit` library . . . . . 7
    - 3.1.1 Summary . . . . . 7
    - 3.1.2 Description . . . . . 23
    - 3.1.3 Module detail . . . . . 24
- 4 Examples 25**
- 5 Contribute 27**
  - 5.1 Clone the repository . . . . . 27
  - 5.2 Adhere to code style . . . . . 27
  - 5.3 Run the tests . . . . . 28
  - 5.4 Build the documentation . . . . . 28
- Python Module Index 29**
- Index 31**



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

The Aali Flowkit Python offers these main features:

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

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

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

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

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

*Contribute*



## GETTING STARTED

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

### 1.1 Installation

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

```
pip install aali-flowkit-python
```

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

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

#### 1.1.1 Quick start

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

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



## **USER GUIDE**

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





## API REFERENCE

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

### 3.1 The `src.aali.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 Aali Flowkit service.

##### Attributes

<code>__version__</code>
--------------------------

## The config package

### Summary

### Description

Configuration package for the application.

## The endpoints package

### Summary

### Submodules

<code>mechscriptbot</code>	Module for triggering the MechanicalScriptingBot application.
<code>splitter</code>	Module for splitting text into chunks.

## The mechscriptbot.py module

### Summary

### Functions

<code>triggermechscriptbot</code>	Endpoint for triggering the MechanicalScriptingBot application.
-----------------------------------	---

### Attributes

<code>router</code>
---------------------

### Description

Module for triggering the MechanicalScriptingBot application.

### Module detail

```
async mechscriptbot.triggermechscriptbot(request:
aali.flowkit.models.mechscriptbot.MechScriptBotRequest,
api_key: str = Header(...)) →
aali.flowkit.models.mechscriptbot.MechScriptBotResponse
```

Endpoint for triggering the MechanicalScriptingBot application.

#### Parameters

**request**

[MechScriptBotRequest] An object containing the input query and other relevant parameters for the MechanicalScriptingBot application.

**api\_key**

[str] The API key for authentication.

**Returns****MechScriptBotResponse**

An object containing the output and other relevant metadata for the MechanicalScriptingBot application.

mechscriptbot.**router**

**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**

*router*

**Constants**

*TOKEN\_TO\_CHARACTER\_MULTIPLIER*

**Description**

Module for splitting text into chunks.

## Module detail

**async** `splitter.split_ppt(request: aali.flowkit.models.splitter.SplitterRequest, api_key: str = Header(...))` → *aali.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: aali.flowkit.models.splitter.SplitterRequest, api_key: str = Header(...))` → *aali.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: aali.flowkit.models.splitter.SplitterRequest, api_key: str = Header(...))` → *aali.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: aali.flowkit.models.splitter.SplitterRequest)` → *aali.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: aali.flowkit.models.splitter.SplitterRequest) →`  
*aali.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: aali.flowkit.models.splitter.SplitterRequest) →`  
*aali.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: aali.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>mechscriptbot</i>	Model for the MechanicalScriptingBot endpoint.
<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.aali.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.aali.flowkit.models.functions import ParameterInfo
```

## Attribute detail

`ParameterInfo.name`: `str`

`ParameterInfo.type`: `str`

## EndpointInfo

**class** `src.aali.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.aali.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.aali.flowkit.models.functions.FunctionCategory(*args, **kws)
    Bases: enum.Enum
    Enum for function categories.
```

## Overview

### Attributes

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

## Import detail

```
from src.aali.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 mechsriptbot.py module

## Summary

## Classes

<i>MechScriptBotRequest</i>	Request model for the MechanicalScriptingBot endpoint.
<i>MechScriptBotResponse</i>	Response model for the MechanicalScriptingBot endpoint.

## MechScriptBotRequest

**class** src.aali.flowkit.models.mechsriptbot.**MechScriptBotRequest**(/, \*\*data: Any)

Bases: pydantic.BaseModel

Request model for the MechanicalScriptingBot endpoint.

### Parameters

#### BaseModel

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

## Overview

## Attributes

<i>question</i>
<i>mech_script_bot_url</i>
<i>full_human_memory</i>
<i>full_ai_memory</i>
<i>full_variables</i>
<i>full_mechanical_objects</i>

## Import detail

```
from src.aali.flowkit.models.mechscriptbot import MechScriptBotRequest
```

## Attribute detail

```
MechScriptBotRequest.question: str
MechScriptBotRequest.mech_script_bot_url: str
MechScriptBotRequest.full_human_memory: list[str]
MechScriptBotRequest.full_ai_memory: list[str]
MechScriptBotRequest.full_variables: list[str]
MechScriptBotRequest.full_mechanical_objects: list[str]
```

## MechScriptBotResponse

```
class src.aali.flowkit.models.mechscriptbot.MechScriptBotResponse(/, **data: Any)
    Bases: pydantic.BaseModel
    Response model for the MechanicalScriptingBot endpoint.
```

### Parameters

#### BaseModel

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

## Overview

## Attributes

<i>output</i>
<i>updated_human_memory</i>
<i>updated_ai_memory</i>
<i>updated_variables</i>
<i>updated_mechanical_objects</i>

## Import detail

```
from src.aali.flowkit.models.mechscriptbot import MechScriptBotResponse
```

## Attribute detail

```
MechScriptBotResponse.output: str
MechScriptBotResponse.updated_human_memory: list[str]
MechScriptBotResponse.updated_ai_memory: list[str]
MechScriptBotResponse.updated_variables: list[str]
MechScriptBotResponse.updated_mechanical_objects: list[str]
```

## Description

Model for the MechanicalScriptingBot endpoint.

## The splitter.py module

## Summary

## Classes

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

## SplitterRequest

```
class src.aali.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

```
document_content
chunk_size
chunk_overlap
```

### Import detail

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

### Attribute detail

`SplitterRequest.document_content:` `bytes`

`SplitterRequest.chunk_size:` `int`

`SplitterRequest.chunk_overlap:` `int`

### SplitterResponse

```
class src.aali.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.aali.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>handle_legacy_port_config</code>	Handle legacy port configuration.
<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__.handle_legacy_port_config()`  
Handle legacy port configuration.

`__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[aali.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 Aali Flowkit service.

### Module detail

`async flowkit_service.list_functions(api_key: str = Header(...)) → list[aali.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 Aali 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 Aali Flowkit Python.

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

### 5.1 Clone the repository

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

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

### 5.2 Adhere to code style

*Aali 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.aali.flowkit`, 7
- `src.aali.flowkit.__main__`, 20
- `src.aali.flowkit.config`, 8
- `src.aali.flowkit.endpoints`, 8
- `src.aali.flowkit.endpoints.mechscriptbot`, 8
- `src.aali.flowkit.endpoints.splitter`, 9
- `src.aali.flowkit.fastapi_utils`, 20
- `src.aali.flowkit.flowkit_service`, 23
- `src.aali.flowkit.models`, 12
- `src.aali.flowkit.models.functions`, 12
- `src.aali.flowkit.models.mechscriptbot`, 15
- `src.aali.flowkit.models.splitter`, 17
- `src.aali.flowkit.utils`, 19
- `src.aali.flowkit.utils.decorators`, 19





## Symbols

`__version__` (in module *flowkit*), 24

## A

`ANSYS_GPT` (in module *FunctionCategory*), 15

## C

`category` (in module *EndpointInfo*), 14  
`category()` (in module *decorators*), 19  
`chunk_overlap` (in module *SplitterRequest*), 18  
`chunk_size` (in module *SplitterRequest*), 18  
`chunks` (in module *SplitterResponse*), 18

## D

`DATA_EXTRACTION` (in module *FunctionCategory*), 15  
`definitions` (in module *EndpointInfo*), 14  
`description` (in module *EndpointInfo*), 14  
`display_name` (in module *EndpointInfo*), 14  
`display_name()` (in module *decorators*), 19  
`document_content` (in module *SplitterRequest*), 18

## E

`extract_definitions_from_schema()` (in module *fastapi\_utils*), 21  
`extract_endpoint_info()` (in module *fastapi\_utils*), 22  
`extract_field_type()` (in module *fastapi\_utils*), 21  
`extract_fields_from_schema()` (in module *fastapi\_utils*), 21

## F

`flowkit_service` (in module *flowkit\_service*), 23  
`full_ai_memory` (in module *MechScriptBotRequest*), 16  
`full_human_memory` (in module *MechScriptBotRequest*), 16  
`full_mechanical_objects` (in module *MechScriptBotRequest*), 16  
`full_variables` (in module *MechScriptBotRequest*), 16  
`function_map` (in module *flowkit\_service*), 23

## G

`GENERIC` (in module *FunctionCategory*), 15

`get_definitions_from_params()` (in module *fastapi\_utils*), 22  
`get_definitions_from_return_type()` (in module *fastapi\_utils*), 22  
`get_parameters_info()` (in module *fastapi\_utils*), 21  
`get_return_type_info()` (in module *fastapi\_utils*), 21

## H

`handle_legacy_port_config()` (in module *\_\_main\_\_*), 20

## I

`inputs` (in module *EndpointInfo*), 14

## K

`KNOWLEDGE_DB` (in module *FunctionCategory*), 15

## L

`list_functions()` (in module *flowkit\_service*), 23  
`LLM_HANDLER` (in module *FunctionCategory*), 15

## M

`main()` (in module *\_\_main\_\_*), 20  
`mech_script_bot_url` (in module *MechScriptBotRequest*), 16

## module

`src.aali.flowkit`, 7  
`src.aali.flowkit.__main__`, 20  
`src.aali.flowkit.config`, 8  
`src.aali.flowkit.endpoints`, 8  
`src.aali.flowkit.endpoints.mechscriptbot`, 8  
`src.aali.flowkit.endpoints.splitter`, 9  
`src.aali.flowkit.fastapi_utils`, 20  
`src.aali.flowkit.flowkit_service`, 23  
`src.aali.flowkit.models`, 12  
`src.aali.flowkit.models.functions`, 12  
`src.aali.flowkit.models.mechscriptbot`, 15  
`src.aali.flowkit.models.splitter`, 17  
`src.aali.flowkit.utils`, 19  
`src.aali.flowkit.utils.decorators`, 19

## N

name (in module *EndpointInfo*), 14

name (in module *ParameterInfo*), 13

## O

output (in module *MechScriptBotResponse*), 17

outputs (in module *EndpointInfo*), 14

## P

parse\_cli\_args() (in module *\_\_main\_\_*), 20

path (in module *EndpointInfo*), 14

process\_pdf() (in module *splitter*), 11

process\_ppt() (in module *splitter*), 10

process\_python\_code() (in module *splitter*), 11

## Q

question (in module *MechScriptBotRequest*), 16

## R

router (in module *mechscriptbot*), 9

router (in module *splitter*), 11

## S

split\_pdf() (in module *splitter*), 10

split\_ppt() (in module *splitter*), 10

split\_py() (in module *splitter*), 10

src.aali.flowkit  
module, 7

src.aali.flowkit.\_\_main\_\_  
module, 20

src.aali.flowkit.config  
module, 8

src.aali.flowkit.endpoints  
module, 8

src.aali.flowkit.endpoints.mechscriptbot  
module, 8

src.aali.flowkit.endpoints.splitter  
module, 9

src.aali.flowkit.fastapi\_utils  
module, 20

src.aali.flowkit.flowkit\_service  
module, 23

src.aali.flowkit.models  
module, 12

src.aali.flowkit.models.functions  
module, 12

src.aali.flowkit.models.functions.EndpointInfo  
(built-in class), 13

src.aali.flowkit.models.functions.FunctionCategory  
(built-in class), 14

src.aali.flowkit.models.functions.ParameterInfo  
(built-in class), 12

src.aali.flowkit.models.mechscriptbot

module, 15

src.aali.flowkit.models.mechscriptbot.MechScriptBotRequest  
(built-in class), 15

src.aali.flowkit.models.mechscriptbot.MechScriptBotResponse  
(built-in class), 16

src.aali.flowkit.models.splitter  
module, 17

src.aali.flowkit.models.splitter.SplitterRequest  
(built-in class), 17

src.aali.flowkit.models.splitter.SplitterResponse  
(built-in class), 18

src.aali.flowkit.utils  
module, 19

src.aali.flowkit.utils.decorators  
module, 19

substitute\_empty\_values() (in module *\_\_main\_\_*),  
20

## T

TOKEN\_TO\_CHARACTER\_MULTIPLIER (in module *splitter*), 11

triggermechscriptbot() (in module *mechscriptbot*),  
8

type (in module *ParameterInfo*), 13

## U

updated\_ai\_memory (in module *MechScriptBotResponse*), 17

updated\_human\_memory (in module *MechScriptBotResponse*), 17

updated\_mechanical\_objects (in module *MechScriptBotResponse*), 17

updated\_variables (in module *MechScriptBotResponse*), 17

## V

validate\_request() (in module *splitter*), 11