

## Header Parameter Models

If you have a group of related header parameters, you can create a Pydantic model to declare them.

This would allow you to re-use the model in multiple places and also to declare validations and metadata for all the parameters at once. 😎

### Note

This is supported since FastAPI version 0.115.0. 😎

### Header Parameters with a Pydantic Model

Declare the header parameters that you need in a Pydantic model, and then declare the parameter as `Header`:

#### Python 3.10+

```
from typing import Annotated

from fastapi import FastAPI, Header
from pydantic import BaseModel

app = FastAPI()

class CommonHeaders(BaseModel):
    host: str
    save_data: bool
    if_modified_since: str | None = None
    traceparent: str | None = None
    x_tag: list[str] = []

@app.get("/items/")
async def read_items(headers: Annotated[CommonHeaders, Header()]):
    return headers
```

#### ► 😎 Other versions and variants

#### Python 3.9+

```
from typing import Annotated, Union

from fastapi import FastAPI, Header
from pydantic import BaseModel

app = FastAPI()

class CommonHeaders(BaseModel):
    host: str
    save_data: bool
    if_modified_since: Union[str, None] = None
    traceparent: Union[str, None] = None
    x_tag: list[str] = []

@app.get("/items/")
async def read_items(headers: Annotated[CommonHeaders, Header()]):
    return headers
```

#### Python 3.8+

```
from typing import List, Union

from fastapi import FastAPI, Header
from pydantic import BaseModel
from typing_extensions import Annotated

app = FastAPI()

class CommonHeaders(BaseModel):
    host: str
    save_data: bool
    if_modified_since: Union[str, None] = None
    traceparent: Union[str, None] = None
    x_tag: List[str] = []

@app.get("/items/")
async def read_items(headers: Annotated[CommonHeaders, Header()]):
    return headers
```

#### Python 3.10+ - non-Annotated

**Tip**

Prefer to use the `Annotated` version if possible.

```
from fastapi import FastAPI, Header
from pydantic import BaseModel

app = FastAPI()

class CommonHeaders(BaseModel):
    host: str
    save_data: bool
    if_modified_since: str | None = None
    traceparent: str | None = None
    x_tag: list[str] = []

@app.get("/items/")
async def read_items(headers: CommonHeaders = Header()):
    return headers
```

**Python 3.9+ - non-Annotated****Tip**

Prefer to use the `Annotated` version if possible.

```
from typing import Union

from fastapi import FastAPI, Header
from pydantic import BaseModel

app = FastAPI()

class CommonHeaders(BaseModel):
    host: str
    save_data: bool
    if_modified_since: Union[str, None] = None
    traceparent: Union[str, None] = None
    x_tag: list[str] = []

@app.get("/items/")
async def read_items(headers: CommonHeaders = Header()):
    return headers
```

**Python 3.8+ - non-Annotated****Tip**

Prefer to use the `Annotated` version if possible.

```
from typing import List, Union

from fastapi import FastAPI, Header
from pydantic import BaseModel

app = FastAPI()

class CommonHeaders(BaseModel):
    host: str
    save_data: bool
    if_modified_since: Union[str, None] = None
    traceparent: Union[str, None] = None
    x_tag: List[str] = []

@app.get("/items/")
async def read_items(headers: CommonHeaders = Header()):
    return headers
```

FastAPI will extract the data for each field from the headers in the request and give you the Pydantic model you defined.

**Check the Docs**

You can see the required headers in the docs UI at `/docs`:

The screenshot shows the FastAPI documentation interface for the `/items/` endpoint. The top navigation bar indicates the version is 0.1.0 and the OAS version is 3.1. Below the header, there's a link to `/openapi.json`. The main content area is titled "default".

**GET /items/ Read Items**

**Parameters**

| Name                                        | Description       |
|---------------------------------------------|-------------------|
| host * required<br>string<br>(header)       | host              |
| save_data * required<br>boolean<br>(header) | --                |
| if_modified_since<br>string<br>(header)     | if_modified_since |
| traceparent<br>string<br>(header)           | traceparent       |
| x_tag<br>array[string]<br>(header)          | Add string item   |

**Servers**

These operation-level options override the global server options.

/

**Execute**

**Responses**

### Forbid Extra Headers

In some special use cases (probably not very common), you might want to restrict the headers that you want to receive.

You can use Pydantic's model configuration to `forbid` any extra fields:

#### Python 3.10+

```
from typing import Annotated
from fastapi import FastAPI, Header
from pydantic import BaseModel

app = FastAPI()

class CommonHeaders(BaseModel):
    model_config = {"extra": "forbid"}

    host: str
    save_data: bool
    if_modified_since: str | None = None
    traceparent: str | None = None
    x_tag: list[str] = []

@app.get("/items/")
async def read_items(headers: Annotated[CommonHeaders, Header()]):
    return headers
```

▶ Other versions and variants

### Python 3.9+

```
from typing import Annotated, Union

from fastapi import FastAPI, Header
from pydantic import BaseModel

app = FastAPI()

class CommonHeaders(BaseModel):
    model_config = {"extra": "forbid"}

    host: str
    save_data: bool
    if_modified_since: Union[str, None] = None
    traceparent: Union[str, None] = None
    x_tag: list[str] = []

@app.get("/items/")
async def read_items(headers: Annotated[CommonHeaders, Header()]):
    return headers
```

### Python 3.8+

```
from typing import List, Union

from fastapi import FastAPI, Header
from pydantic import BaseModel
from typing_extensions import Annotated

app = FastAPI()

class CommonHeaders(BaseModel):
    model_config = {"extra": "forbid"}

    host: str
    save_data: bool
    if_modified_since: Union[str, None] = None
    traceparent: Union[str, None] = None
    x_tag: List[str] = []

@app.get("/items/")
async def read_items(headers: Annotated[CommonHeaders, Header()]):
    return headers
```

### Python 3.10+ - non-Annotated

#### Tip

Prefer to use the `Annotated` version if possible.

```
from fastapi import FastAPI, Header
from pydantic import BaseModel

app = FastAPI()

class CommonHeaders(BaseModel):
    model_config = {"extra": "forbid"}

    host: str
    save_data: bool
    if_modified_since: str | None = None
    traceparent: str | None = None
    x_tag: list[str] = []

@app.get("/items/")
async def read_items(headers: CommonHeaders = Header()):
    return headers
```

### Python 3.9+ - non-Annotated

#### Tip

Prefer to use the `Annotated` version if possible.

```
from typing import Union
```

```
from fastapi import FastAPI, Header
from pydantic import BaseModel

app = FastAPI()

class CommonHeaders(BaseModel):
    model_config = {"extra": "forbid"}

    host: str
    save_data: bool
    if_modified_since: Union[str, None] = None
    traceparent: Union[str, None] = None
    x_tag: list[str] = []

@app.get("/items/")
async def read_items(headers: CommonHeaders = Header()):
    return headers
```

#### Python 3.8+ - non-Annotated

##### Tip

Prefer to use the `Annotated` version if possible.

```
from typing import List, Union

from fastapi import FastAPI, Header
from pydantic import BaseModel

app = FastAPI()

class CommonHeaders(BaseModel):
    model_config = {"extra": "forbid"}

    host: str
    save_data: bool
    if_modified_since: Union[str, None] = None
    traceparent: Union[str, None] = None
    x_tag: List[str] = []

@app.get("/items/")
async def read_items(headers: CommonHeaders = Header()):
    return headers
```

If a client tries to send some `extra` headers, they will receive an error response.

For example, if the client tries to send a `tool` header with a value of `plumbus`, they will receive an error response telling them that the header parameter `tool` is not allowed:

```
{
  "detail": [
    {
      "type": "extra_forbidden",
      "loc": ["header", "tool"],
      "msg": "Extra inputs are not permitted",
      "input": "plumbus",
    }
  ]
}
```

#### Summary

You can use Pydantic models to declare headers in FastAPI. 😎

© 2018 Sebastián Ramírez  
Licensed under the MIT License.  
<https://fastapi.tiangolo.com/tutorial/header-param-models/>

Exported from DevDocs — <https://devdocs.io>