OpenAPI Callbacks

You could create an API with a *path operation* that could trigger a request to an *external API* created by someone else (probably the same developer that would be *using* your API).

The process that happens when your API app calls the *external API* is named a "callback". Because the software that the external developer wrote sends a request to your API and then your API *calls back*, sending a request to an *external API* (that was probably created by the same developer).

In this case, you could want to document how that external API *should* look like. What *path operation* it should have, what body it should expect, what response it should return, etc.

An app with callbacks

Let's see all this with an example.

Imagine you develop an app that allows creating invoices.

These invoices will have an `id`, `title` (optional), `customer`, and `total`.

The user of your API (an external developer) will create an invoice in your API with a POST request.

Then your API will (let's imagine):

- Send the invoice to some customer of the external developer.
- Collect the money.
- Send a notification back to the API user (the external developer).
    - This will be done by sending a POST request (from *your API*) to some *external API* provided by that external developer (this is the "callback").

The normal FastAPI app

Let's first see how the normal API app would look like before adding the callback.

It will have a *path operation* that will receive an `Invoice` body, and a query parameter `callback_url` that will contain the URL for the callback.

This part is pretty normal, most of the code is probably already familiar to you:

```
{!../../../docs_src/openapi_callbacks/tutorial001.py!}
```

!!! tip The `callback_url` query parameter uses a Pydantic [URL](https://pydantic-docs.helpmanual.io/usage/types/#urls) type.

The only new thing is the `callbacks=messages_callback_router.routes` as an argument to the *path operation decorator*. We'll see what that is next.

Documenting the callback

The actual callback code will depend heavily on your own API app.

And it will probably vary a lot from one app to the next.

It could be just one or two lines of code, like:

```
callback_url = "https://example.com/api/v1/invoices/events/"
requests.post(callback_url, json={"description": "Invoice paid", "paid": True})
```

But possibly the most important part of the callback is making sure that your API user (the external developer) implements the *external API* correctly, according to the data that *your API* is going to send in the request body of the callback, etc.

So, what we will do next is add the code to document how that *external API* should look like to receive the callback from *your API*.

That documentation will show up in the Swagger UI at `/docs` in your API, and it will let external developers know how to build the *external API*.

This example doesn't implement the callback itself (that could be just a line of code), only the documentation part.

!!! tip The actual callback is just an HTTP request.

```
When implementing the callback yourself, you could use something like <a
href="https://www.encode.io/httpx/" class="external-link" target="_blank">HTTPX</a> or
<a href="https://requests.readthedocs.io/" class="external-link"
target="_blank">Requests</a>.
```

Write the callback documentation code

This code won't be executed in your app, we only need it to *document* how that *external API* should look like.

But, you already know how to easily create automatic documentation for an API with **FastAPI**.

So we are going to use that same knowledge to document how the *external API* should look like... by creating the *path operation(s)* that the external API should implement (the ones your API will call).

!!! tip When writing the code to document a callback, it might be useful to imagine that you are that *external developer*. And that you are currently implementing the *external API*, not *your API*.

```
Temporarily adopting this point of view (of the *external developer*) can help you
feel like it's more obvious where to put the parameters, the Pydantic model for the
body, for the response, etc. for that *external API*.
```

Create a callback `APIRouter`

First create a new `APIRouter` that will contain one or more callbacks.

```
{!../../../docs_src/openapi_callbacks/tutorial001.py!}
```

Create the callback *path operation*

To create the callback *path operation* use the same `APIRouter` you created above.

It should look just like a normal FastAPI *path operation*:

- It should probably have a declaration of the body it should receive, e.g. `body: InvoiceEvent`.

- And it could also have a declaration of the response it should return, e.g. `response_model=InvoiceEventReceived`.

```
{!../../../docs_src/openapi_callbacks/tutorial001.py!}
```

There are 2 main differences from a normal *path operation*:

- It doesn't need to have any actual code, because your app will never call this code. It's only used to document the *external API*. So, the function could just have `pass`.
- The *path* can contain an OpenAPI 3 expression (see more below) where it can use variables with parameters and parts of the original request sent to *your API*.

The callback path expression

The callback *path* can have an OpenAPI 3 expression that can contain parts of the original request sent to *your API*.

In this case, it's the `str`:

```
"{$callback_url}/invoices/{$request.body.id}"
```

So, if your API user (the external developer) sends a request to *your API* to:

```
https://yourapi.com/invoices/?callback_url=https://www.external.org/events
```

with a JSON body of:

```
{
    "id": "2expen51ve",
    "customer": "Mr. Richie Rich",
    "total": "9999"
}
```

Then *your API* will process the invoice, and at some point later, send a callback request to the `callback_url` (the *external API*):

```
https://www.external.org/events/invoices/2expen51ve
```

with a JSON body containing something like:

```
{
    "description": "Payment celebration",
    "paid": true
}
```

and it would expect a response from that *external API* with a JSON body like:

```
{
    "ok": true
}
```

!!! tip Notice how the callback URL used contains the URL received as a query parameter in `callback_url` (`https://www.external.org/events`) and also the invoice `id` from inside of the JSON body (`2expen51ve`).

Add the callback router

At this point you have the *callback path operation(s)* needed (the one(s) that the *external developer* should implement in the *external API*) in the callback router you created above.

Now use the parameter `callbacks` in *your API's path operation decorator* to pass the attribute `.routes` (that's actually just a `list` of routes/*path operations*) from that callback router:

```
{!../../../docs_src/openapi_callbacks/tutorial001.py!}
```

!!! tip Notice that you are not passing the router itself (`invoices_callback_router`) to `callback=`, but the attribute `.routes`, as in `invoices_callback_router.routes`.

Check the docs

Now you can start your app with Uvicorn and go to http://127.0.0.1:8000/docs.

You will see your docs including a "Callback" section for your *path operation* that shows how the *external API* should look like: