What are Actions?

In addition to using our built-in capabilities, you can also define custom actions by making one or more APIs available to the GPT

What are Actions?

An action in the context of GPT is a custom API that allows you to extend the capabilities of your GPT model by integrating external data or interacting with the real world.

Actions are similar to plugins, but they offer greater control over the model and how your APIs are called.

Here are some examples of how actions can be used:

- Connecting a GPT to a database to retrieve information
- Plugging a GPT into an email inbox to process and respond to messages
- Facilitating e-commerce orders by connecting a GPT to a payment gateway

Actions are designed to be easy to use and integrate with your existing GPT models. You can use your existing plugin manifest to define actions for your GPT, or you can create new actions from scratch.

What are Actions?

Here are some of the benefits of using actions:

- Extend the capabilities of your GPT models: Actions allow you to add new functionality to your GPT models that would not be possible with the built-in capabilities.
- Integrate external data: Actions allow you to connect your GPT models to external data sources, such as databases or APIs. This can make your models more informative and powerful.
- Interact with the real world: Actions allow your GPT models to interact with the real world, such as sending emails or controlling devices. This can make your models more useful and practical.

If you are looking to extend the capabilities of your GPT models, then actions are a great way to do it. Actions are easy to use, powerful, and can be used to create a wide variety of applications.

What are API’s

An API, or Application Programming Interface, is a set of rules and protocols for building and interacting with software applications. APIs define the methods and data formats that developers can use to interact with the software component, be it operating systems, libraries, or different services. They are crucial in modern software development as they enable different software systems to communicate and share data.

Web API’s

Web APIs are interfaces that allow applications to interact with each other over the internet. They enable the integration of different software systems, often using HTTP/HTTPS protocols. Web APIs are crucial in modern web development, allowing for the creation of rich, feature-filled web applications by leveraging services provided by external servers.

REST APIs

REST, or Representational State Transfer, is an architectural style for designing networked applications. RESTful APIs, or REST APIs, adhere to this style and use HTTP requests to access and use data. They are stateless, meaning that each request from a client to server must contain all the information needed to understand and complete the request. REST APIs are known for their simplicity and scalability.

RESTful API Methods: GET

- GET: Retrieve data from the server. It should have no side effects and be idempotent (the same request can be made repeatedly with the same effect).

  Example of a GET Request : A GET request is used to retrieve data from the server. In the case of the Quotable API, you can use a GET request to fetch a random quote or quotes by a specific author, tag, etc.

  - `GET https://api.quotable.io/random`
  - Get one or more random quotes from the database. This method supports several filters that can be used to get random quotes with specific properties (ie tags, quote length, etc.)
  - By default, this methods returns a single random quote. You can specify the number of random quotes to return via the `limit` parameter.
  - Command line: `curl https://api.quotable.io/random`
  - Expected Response:

```
{

"_id": "RandomQuoteID",

"content": "Random quote text.",

"author": "Author Name",

// ... other fields }
```

Let’s Build our first action for Authentication Type None

Generate a Random Quote https://api.quotable.io/random

Open Your Browser: https://api.quotable.io/random

{"_id":"_hJS3LX4Qz","content":"Technology has to be invented or adopted.","author":"Jared Diamond","tags":["Technology"],"authorSlug":"jared-diamond","length":41,"dateAdded":"2021-03-08","dateModified":"2023-04-14"}

Default Schema

```
{
  "openapi": "3.1.0",
  "info": {
    "title": "Untitled",
    "description": "Your OpenAPI specification",
    "version": "v1.0.0"
  },
"servers": [
    {
      "url": ""
    }
  ],
  "paths": {},
  "components": {
    "schemas": {}
  }
}
```

What does that schema mean?

The JSON structure is a template for an OpenAPI Specification (OAS) document.

The OpenAPI Specification, formerly known as the Swagger Specification, is a standard, language-agnostic interface to RESTful APIs which allows both humans and computers to discover and understand the capabilities of a service without access to source code, documentation, or through network traffic inspection.

Here's a breakdown of the components in the template:

- `openapi`: Specifies the version of the OpenAPI Specification that this document adheres to. In your case, it's `3.1.0`, which is a recent version of the specification.
- `info`: Provides metadata about the API. This includes:
  - `title`: The title of the API.
  - `description`: A short description of the API.
  - `version`: The version of the API.

Here's a breakdown of the components in the template:

- `servers`: An array of Server Objects, which provide connectivity information to a target server. In your template, the `url` field is empty, which should normally contain the URL of the API server.
- `paths`: Holds the relative paths to the individual endpoints. The path is appended to the URL from the Server Object in order to construct the full URL. Each path then contains an HTTP operation and its details. In your template, this is an empty object, which means no endpoints are defined.
- `components`: Holds a set of reusable components for the API specification, like schemas (which represent data models), responses, parameters, examples, request bodies, headers, security schemes, links, and callbacks. In your template, `schemas` is empty, indicating no data models are defined.

Updated Schema for our example

```
{
  "openapi": "3.1.0",
  "info": {
    "title": "Random Quote API",
    "description": "An API for fetching a random quote.",
    "version": "1.0.0"
  },
  "servers": [
    {
      "url": "https://api.quotable.io"
    }
  ],
  "paths": {
    "/random": {
      "get": {
        "summary": "Get a Random Quote",
        "operationId": "getRandomQuote",
        "responses": {
          "200": {
            "description": "Successful response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Quote"
                  }
                }
              }
            }
          }
        }
      }
    }
  },
```

Updated Schema for our example

```
"components": {
    "schemas": {
      "Quote": {
        "type": "object",
        "properties": {
          "_id": {
            "type": "string"
          },
          "content": {
            "type": "string"
          },
          "author": {
            "type": "string"
          },
          "length": {
            "type": "integer"
          }
        },
        "required": ["_id", "content", "author",
"length"]
      }
    }
  }
}
```

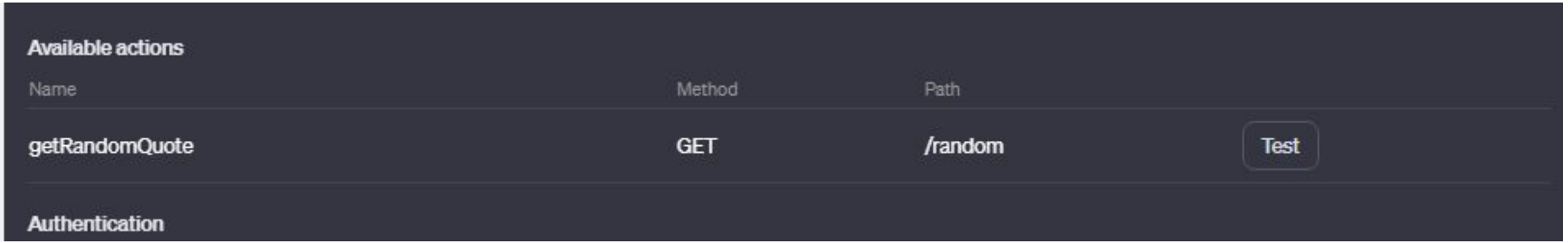
Wonder It’s too complex?

Use the prompts to generate schema

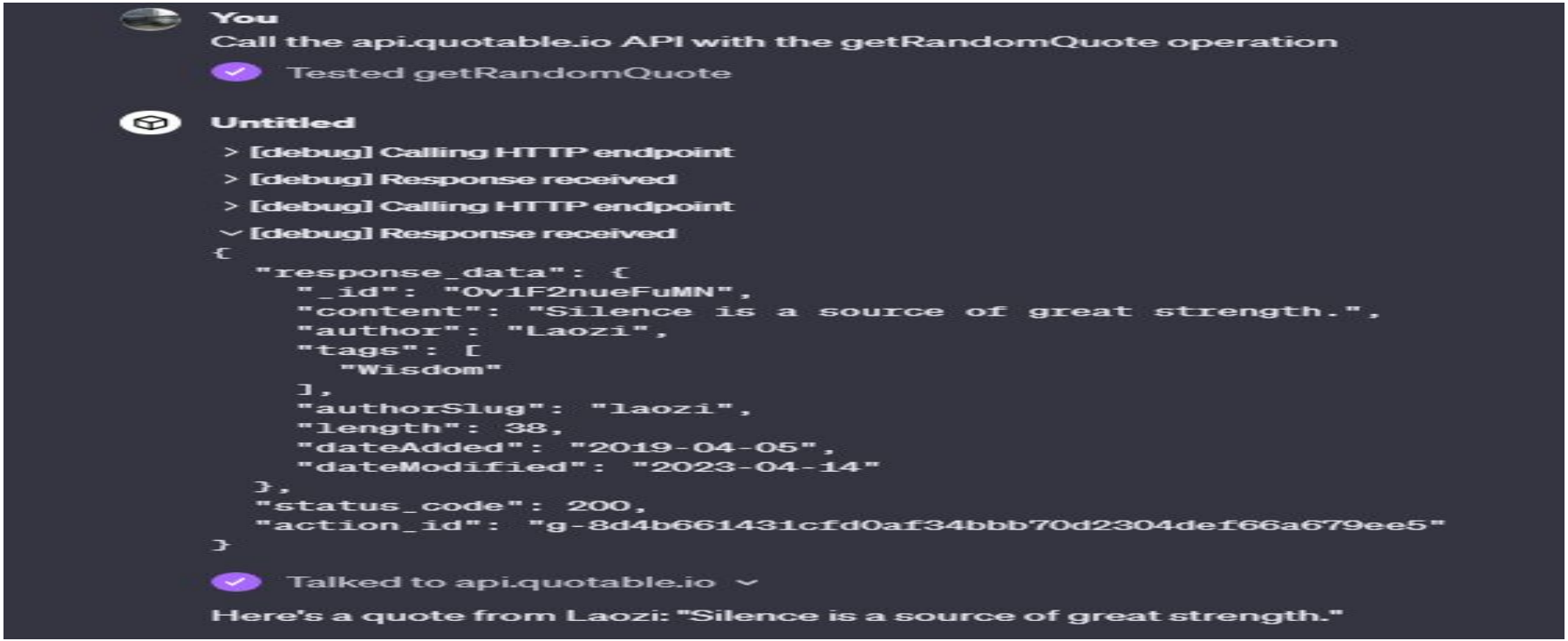
i am trying to build a GPT which allows passing data with webhooks, Below is the example structure we need to abide and server we need is https:api.quotable.io , the action i need only is when we call upon this schema we receive a random quote

```
{
  "openapi": "3.1.0",
  "info": {
    "title": "Untitled",
    "description": "Your OpenAPI specification",
    "version": "v1.0.0"
  },
  "servers": [
    {
      "url": ""
    }
  ],
  "paths": {},
  "components": {
    "schemas": {}
  }
}
```

Once we have populated our Schema We will get this



When we Click on Test



Still if you have issues?

Prompt to debug: show json payload for this action

Other Potential Issues

1. Method and Path Not Specified: The debug log shows `"method": null` and `"path": null`. This suggests that the actual HTTP method (GET) and the path (`/random`) are not being correctly specified in the API call. The API call should be a GET request to `https://api.quotable.io/random`.
2. Incorrect Operation Name: The operation name `get__random` does not directly correspond to a standard REST API call. In RESTful services, you typically make a GET request directly to a URL endpoint (like `/random`) rather than calling an operation by name. The operation name `get__random` might not be recognized by the system you are using to make the API call.