

How to Write API Documentation as a Technical Writer

BY JORDAN STANCHEV

JPDOCU SCHOOL OF TECHNICAL WRITING



Writing Documentation using Swagger UI

BY JORDAN STANCHEV

JPDOCU SCHOOL OF TECHNICAL WRITING

Agenda

- ❑ What is Swagger?
- ❑ Benefits from using Swagger for writing documentation
- ❑ 7 Steps to wringing API documentation using Swagger

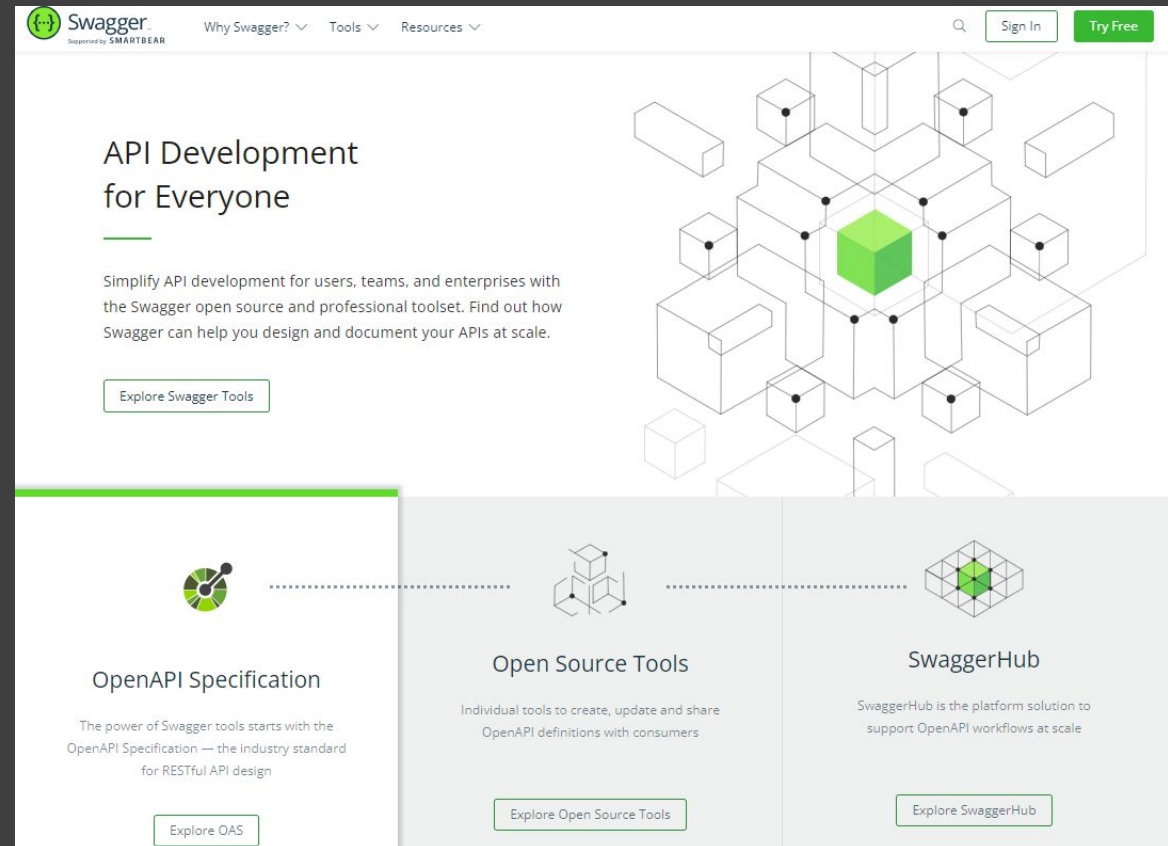
What is Swagger?

<https://swagger.io/>

An API development platform that brings together capabilities of the open source Swagger framework, as well as additional capabilities that allows you to:

- build,
- document,
- manage,
- and deploy

APIs”



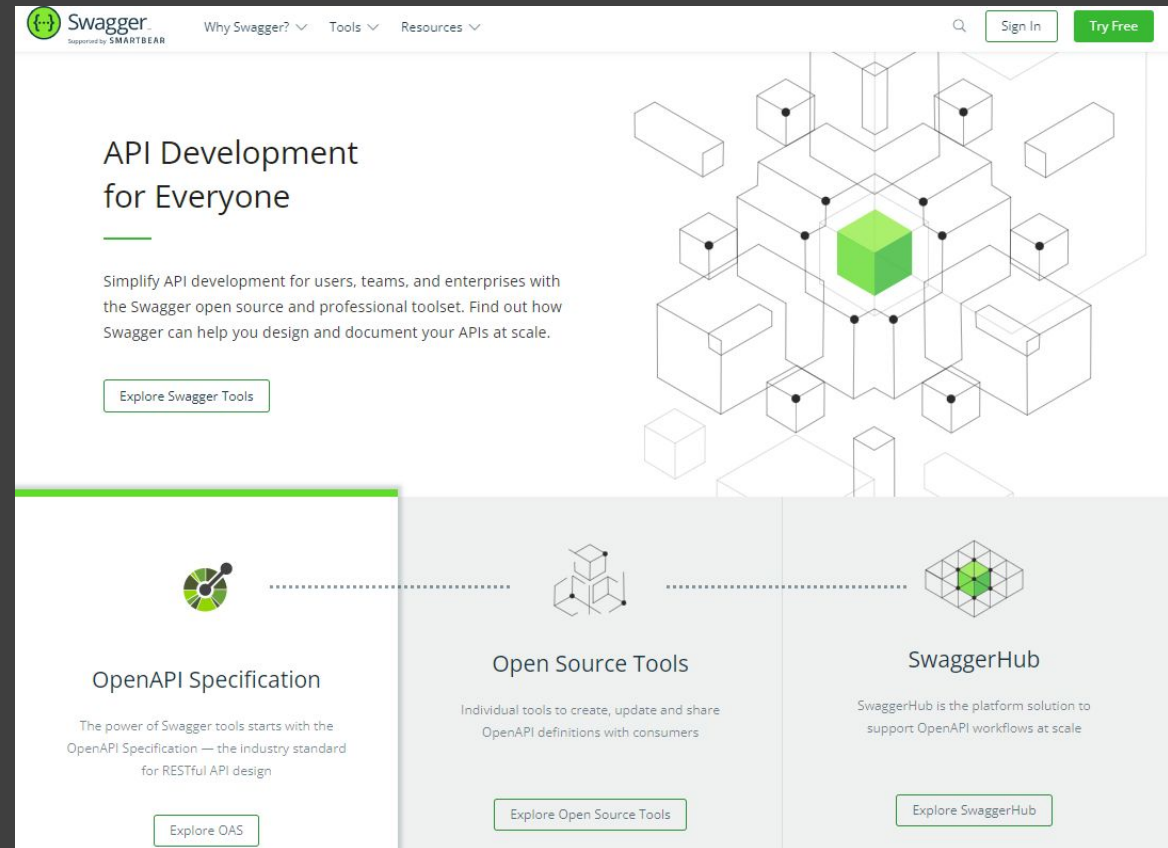
What is Swagger?

<https://swagger.io/>

An API development platform that brings together capabilities of the open source Swagger framework, as well as additional capabilities that allows you to:

- build,
- document,
- manage,
- and deploy

APIs”



Benefits from using Swagger for writing documentation

Write as you design approach:

- Create a new REST (that is, representational state transfer) API
- Document it while designing your REST API in Swagger
- Predefined tools to avoid the need to write directly in JSON or YAML

Step 1: Create New API

Prerequisites

Create your account and log on to the SwaggerHub:

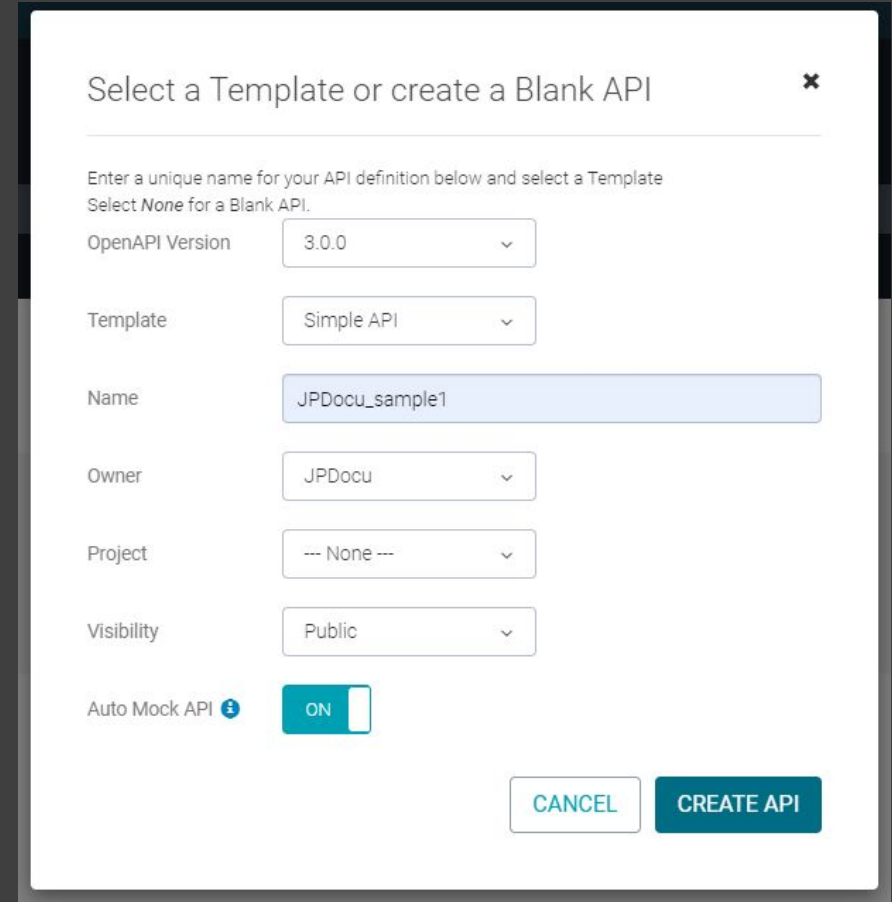
<https://app.swaggerhub.com/home>

Steps

1. Choose *Create New*.
2. Choose *Create New API*.
3. In the dialog that appears, enter your sample API details.
Note: On the free plan you can only have a Public API.
4. Choose *Create API* button.

Result

Your new API appears in the list of APIs under *My Hub*.



The screenshot shows a modal dialog titled "Select a Template or create a Blank API" with a close button (X) in the top right corner. Below the title, there is a text prompt: "Enter a unique name for your API definition below and select a Template. Select *None* for a Blank API." The form contains several fields: "OpenAPI Version" with a dropdown menu showing "3.0.0"; "Template" with a dropdown menu showing "Simple API"; "Name" with a text input field containing "JPDocu_sample1"; "Owner" with a dropdown menu showing "JPDocu"; "Project" with a dropdown menu showing "--- None ---"; "Visibility" with a dropdown menu showing "Public"; and "Auto Mock API" with a toggle switch currently set to "ON". At the bottom right of the dialog are two buttons: "CANCEL" and "CREATE API".

Demo

Step 2: Create the API Introductory Page

1. Under *My Hub*, select your newly created API.
2. Click on *Open to Edit* that appears.
Your API is loaded in the Swagger UI editor tool.

Step 2.1: Edit the API Info

1. Under *My Hub*, select your newly created API.
2. Click on *Open to Edit* that appears. Your API is loaded in the Swagger UI editor tool.
3. Enter the API summary information. Make it clear what is the intended use of the API. This is like a landing page for your API documentation for people who see it for the first time. Make it easy to follow.

Sample Introductory Info

The image shows the SwaggerHub interface for the 'Simple Inventory API'. The left sidebar contains the 'Info' tab, which is highlighted with a red box. Below the 'Info' tab, the 'General' section is expanded, showing fields for 'Title', 'Description', 'Contact Email', and 'License'. These fields are also highlighted with red boxes. Red arrows point from these fields to the corresponding fields in the main API configuration area on the right.

API Configuration Fields:

- Title:** Simple Inventory API
- Description:** This is a simple API
- Contact Email:** you@your-company.com
- License:** Apache 2.0

API Endpoints:

- admins:** Secured Admin-only calls
 - POST /inventory:** adds an inventory item
- developers:** Operations available to regular developers
 - GET /inventory:** searches inventory

Schemas:

- InventoryItem**
- Manufacturer**

Footer:

SSL certificates are validated | Do not validate
Routing requests via SwaggerHub proxy | Use browser instead

Demo

Step 2.2: Edit the API Contacts and Links to Further Documentation

1. Make it clear whom to contact in case of questions about the API.
2. Provide a link to your further documentation – getting started guide, tutorials, end-users guides, etc., that are NOT a part of the API documentation itself.

Remember the documentation framework steps – supply information about the connection and authentication mechanism to connect to the API.

Sample Contacts and Documentation Links

The screenshot displays the SwaggerHub interface for a project named 'JPDocu_sample1' at version '1.0.0'. The left sidebar contains navigation options: 'Info' (selected), 'Tags', 'Servers', 'Search', 'admins', 'developers', and 'Schemas'. The 'Info' tab is active, showing a form for 'Contact' information. The form fields are: 'Contact Name' (Jordan Stanchev), 'Contact Email' (jordan.stanchev@jpdocu.com), and 'Contact URL' (empty). Below the form, there are sections for 'License', 'Documentation', and 'External Docs'. The 'External Docs' section contains a text input with 'Sample Documentation about using the API' and a 'Documentation URL' input with 'https://github.com/JordanStanchev/Getting-Started-as-User-Assista'. Two red arrows point from the 'Contact Name' and 'Contact Email' fields to the 'Contact' section of the 'Simple Inventory API' details on the right. The 'Simple Inventory API' details include a description: 'This is a simple API. And this is the description that will tell you what you can use this API for.' Below the description are links for 'Contact Jordan Stanchev', 'Apache 2.0', and 'Sample Documentation about using the API'. The 'Servers' section shows a dropdown menu with the URL 'https://virtserver.swaggerhub.com/JPDocu/JPDocu_sam...'. The 'SwaggerHub API Auto Mocking' section is also visible. The 'admins' section is titled 'Secured Admin-only calls' and contains a 'POST /inventory' endpoint that 'adds an inventory item'. The 'developers' section is titled 'Operations available to regular developers' and contains a 'GET /inventory' endpoint that 'searches inventory'.

SMARTBEAR
SwaggerHub

JPDocu_sample1 1.0.0

Info

Tags

Servers

Search

admins

developers

Schemas

SCHEMA InventoryItem

SCHEMA Manufacturer

Info

SAVE

SYNC

General

Contact

Contact Name

Jordan Stanchev

Contact Email

jordan.stanchev@jpdocu.com

Contact URL

License

Documentation

External Docs

Sample Documentation about using the API

Documentation URL *

https://github.com/JordanStanchev/Getting-Started-as-User-Assista

Simple Inventory API

1.0.0 OAS3

This is a simple API. And this is the description that will tell you what you can use this API for.

Contact Jordan Stanchev

Apache 2.0

Sample Documentation about using the API

Servers

https://virtserver.swaggerhub.com/JPDocu/JPDocu_sam...

SwaggerHub API Auto Mocking

admins Secured Admin-only calls

POST /inventory adds an inventory item

developers Operations available to regular developers

GET /inventory searches inventory

Demo

Step 3: Define the Tags of your API

To make your content well-structured and organized, you can tag content with metadata. For example, if your content is relevant for admins, you can create a tag to cluster information about these roles:

The screenshot shows the SwaggerHub interface for a project named 'JPDocu_sample1' with version '1.0.0'. The left sidebar contains navigation links for 'Info', 'Tags', 'Servers', and 'Search'. The 'Tags' link is highlighted with a red box. The main editor area displays the OpenAPI specification for the 'Simple Inventory API'. The 'tags' section of the specification is highlighted with a red box and contains two tags:

```
tags:
  - name: admins
    description: Secured Admin-only calls
  - name: developers
    description: Operations available to regular developers
```

Red arrows point from these tags in the code to the corresponding tags in the UI on the right. The UI shows the 'Simple Inventory API' with version '1.0.0' and 'OAS3' format. It includes a description, contact information, and a list of servers. The 'admins' and 'developers' tags are listed with their descriptions:

- admins** Secured Admin-only calls
- developers** Operations available to regular developers

Step 4: Define the API Operations and Paths

1. Define which are the *operations* that your API supported (GET, SET, POST, DELETE, etc.)
2. Define the path (that is, *endpoints*) in the URL that was defined to access this part of the API. (/path)
3. Define the tags (if applicable) to use for this path.
4. Write a short summary and a more detailed description about this part of your API.
5. Describe the API request parameters.
6. Describe the API responses.

The screenshot displays the Swagger UI editor interface. On the left, the 'Info' sidebar shows the 'developers' tag selected. The main editor area shows the JSON definition for the `getInventory` operation. Red boxes and arrows highlight the following elements:

- Path:** `/inventory` is highlighted in the JSON and mapped to the `/inventory` path in the UI.
- Method:** `get` is highlighted in the JSON and mapped to the `GET` method in the UI.
- Tags:** `developers` is highlighted in the JSON and mapped to the `developers` tag in the UI.
- Summary:** `summary: searches inventory` is highlighted in the JSON and mapped to the `searches inventory` summary in the UI.
- Description:** `By passing in the appropriate options, you can search for available inventory in the system` is highlighted in the JSON and mapped to the description in the UI.
- Parameters:** The JSON defines three query parameters: `searchString`, `skip`, and `limit`. These are mapped to the 'Parameters' section in the UI, which includes input fields for each parameter.
- Responses:** The JSON defines a `200` response with a description of 'search results matching criteria'. This is mapped to the 'Responses' section in the UI, which shows the `200` status code and its description.

The bottom of the editor shows the status 'VALID' and the last saved time '9:32:32 am - Nov 22, 2021'.

Sample API Operation and Path

The screenshot displays an API documentation interface with a sidebar on the left and a main content area on the right. The sidebar includes sections for 'Info', 'Tags', 'Servers', 'Search', and 'Schemas'. The main content area shows the details of an API operation for the 'developers' group.

API Operation Details (Left Panel):

```
paths:
  /inventory:
    get:
      tags:
        - developers
      summary: searches inventory
      operationId: searchInventory
      description: |
        By passing in the appropriate options, you can search for
        available inventory in the system
      parameters:
        - in: query
          name: searchString
          description: pass an optional search string for looking up inventory
          required: false
          schema:
            type: string
        - in: query
          name: skip
          description: number of records to skip for pagination
          schema:
            type: integer
            format: int32
            minimum: 0
        - in: query
          name: limit
          description: maximum number of records to return
          schema:
            type: integer
            format: int32
            minimum: 0
            maximum: 50
      responses:
        '200':
          description: search results matching criteria
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/InventoryItem'
```

API Operation Details (Right Panel):

GET /inventory searches inventory

By passing in the appropriate options, you can search for available inventory in the system

Parameters

Name	Description
searchString	pass an optional search string for looking up inventory
skip	number of records to skip for pagination
limit	maximum number of records to return

Responses

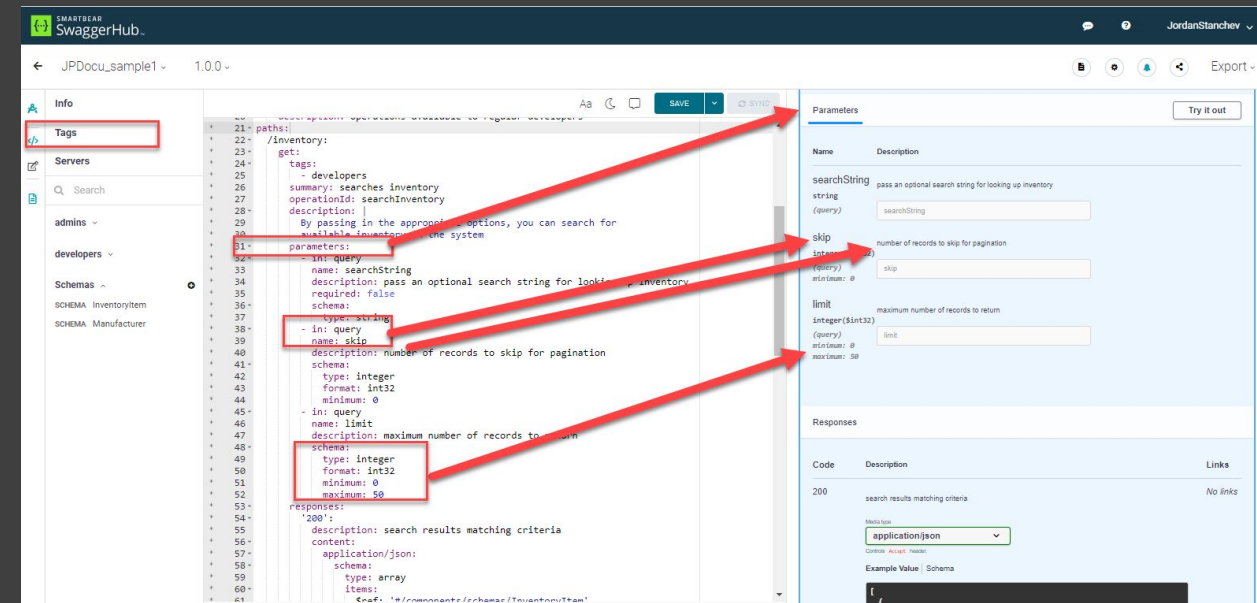
Code	Description	Links
200	search results matching criteria	No links

Red arrows indicate the mapping from the OpenAPI spec to the UI: from the `paths` section to the operation path, from the `get` method to the HTTP method, from the `summary` to the operation summary, and from the `description` to the operation description.

Demo

Step 5: Document the API Parameters

1. Define which are the *parameters* of the API requests.
2. Define how the parameter is received.
3. Define the parameter name and its description.
4. Define if the parameter is mandatory (required: true) or optional (required: false).
5. Define additional parameter properties, as required.



Sample Parameters

The screenshot displays the SwaggerHub interface for a sample API. The left sidebar shows the 'Tags' tab selected. The main editor shows the OpenAPI specification for the `/inventory` endpoint. The `parameters` section is highlighted with a red box. Three red arrows point from specific parameter definitions in the spec to their corresponding interactive UI elements on the right:

- From the `searchString` parameter definition to the `searchString` input field.
- From the `skip` parameter definition to the `skip` input field.
- From the `limit` parameter definition to the `limit` input field.

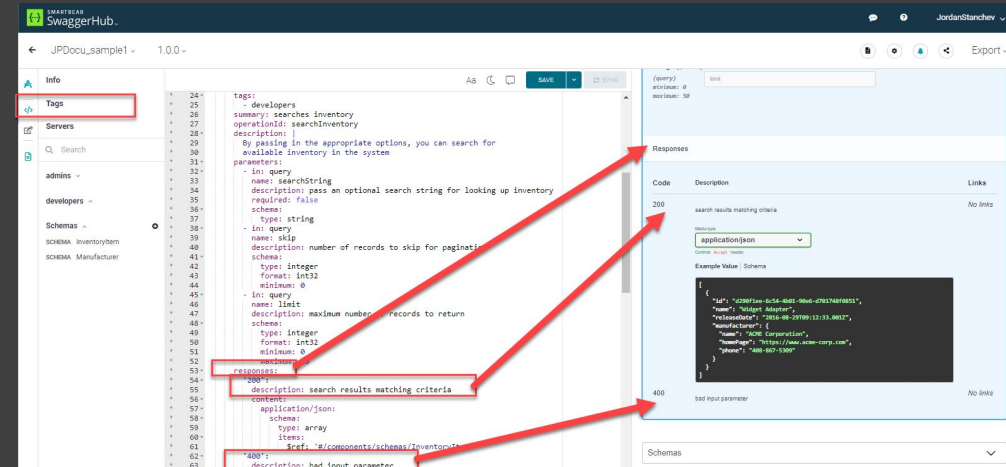
The right sidebar shows the 'Parameters' section with the following details:

Name	Description
searchString	pass an optional search string for looking up inventory
skip	number of records to skip for pagination
limit	maximum number of records to return

Below the parameters, the 'Responses' section shows a 200 response with the description 'search results matching criteria'. The 'Code' column shows '200' and the 'Description' column shows 'search results matching criteria'. The 'Links' column shows 'No links'. The 'Media type' dropdown is set to 'application/json'. The 'Example Value' field shows a JSON array structure.

Step 6: Document the API Responses

1. Define which are the responses that your API returns.
2. Define responses codes and error response codes that the API returns.



Sample Response

The screenshot displays the SwaggerHub interface for a project named "JPDocu_sample1" version "1.0.0". The left sidebar shows navigation options: Info, Tags (highlighted with a red box), Servers, Search, admins, developers, and Schemas. The central area shows the OpenAPI specification in YAML format. The right panel displays the details for a specific endpoint, including a table of responses.

OpenAPI Specification (YAML):

```
tags:
  - developers
summary: searches inventory
operationId: searchInventory
description: |
  By passing in the appropriate options, you can search for
  available inventory in the system
parameters:
  - in: query
    name: searchString
    description: pass an optional search string for looking up inventory
    required: false
    schema:
      type: string
  - in: query
    name: skip
    description: number of records to skip for pagination
    schema:
      type: integer
      format: int32
      minimum: 0
  - in: query
    name: limit
    description: maximum number of records to return
    schema:
      type: integer
      format: int32
      minimum: 0
      maximum: 50
responses:
  '200':
    description: search results matching criteria
    content:
      application/json:
        schema:
          type: array
          items:
            $ref: '#/components/schemas/InventoryItem'
  '400':
    description: bad input parameter
```

Endpoint Details:

The endpoint is a GET request with a query parameter "limit" (type: integer, minimum: 0, maximum: 50). The media type is set to "application/json".

Responses Table:

Code	Description	Links
200	search results matching criteria	No links
400	bad input parameter	No links

Sample Response (JSON):

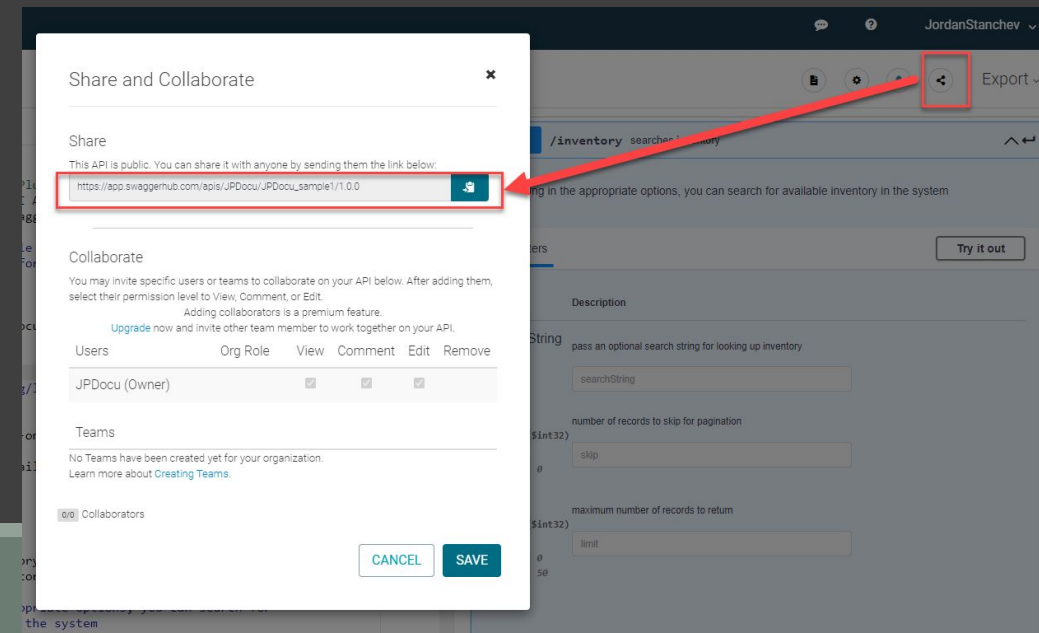
```
{
  "id": "d290f1ee-6c54-4b01-90e6-d701748f0851",
  "name": "Widget Adapter",
  "releaseDate": "2016-08-29T09:12:33.001Z",
  "manufacturer": {
    "name": "ACME Corporation",
    "homePage": "https://www.acme-corp.com",
    "phone": "408-867-5309"
  }
}
```

Step 7: Export the API documentation

Congratulations! You have just finished writing your first API documentation! Let's deliver it to the rest of the world!

Choose *Share and Collaborate* button.

This gives you the URL that points to your documentation!



Sample

The image shows a 'Share and Collaborate' modal window overlaid on a SwaggerHub API interface. The modal has a title bar with a close button. It is divided into two main sections: 'Share' and 'Collaborate'.

Share Section:

- Text: "This API is public. You can share it with anyone by sending them the link below:"
- A text input field containing the URL: `https://app.swaggerhub.com/apis/JPDocu/JPDocu_sample1/1.0.0`. This field and its copy icon are highlighted with a red box.
- A red arrow points from the copy icon in the modal to the share icon in the background interface.

Collaborate Section:

- Text: "You may invite specific users or teams to collaborate on your API below. After adding them, select their permission level to View, Comment, or Edit. Adding collaborators is a premium feature. Upgrade now and invite other team member to work together on your API."
- A table with columns: Users, Org Role, View, Comment, Edit, Remove.
- Table content:

Users	Org Role	View	Comment	Edit	Remove
JPDocu (Owner)		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
- Text: "No Teams have been created yet for your organization. Learn more about [Creating Teams](#)."
- Text: "0/0 Collaborators"
- Buttons: "CANCEL" and "SAVE"

Background Interface:

- Top right: User name "JordanStanchev" and a dropdown arrow.
- Below user name: A row of icons including a share icon, which is highlighted with a red box.
- Below share icon: An "Export" dropdown menu.
- Main content area: A search endpoint `/inventory` with a description: "pass an optional search string for looking up inventory". It includes input fields for `searchString`, `skip` (number of records to skip for pagination), and `limit` (maximum number of records to return).
- A "Try it out" button is visible on the right side of the main content area.

Thank you!

More courses from JPDocu School of Technical Writing:

UX Writing for Technical Writers:

- ❖ [A Quick Start to Software Documentation](#)
- ❖ [How to Write Software Documentation](#)
- ❖ [UX and Information Architecture Basics for the Technical Writer](#)
- ❖ [Project Management for Technical Writers](#)
- ❖ [Graphics For Software Documentation](#)
- ❖ [How to Create Instructional Videos](#)

DITA XML Writing for Technical Writers:

- ❖ [A Quick Start to Technical Writing with DITA](#)
- ❖ [Common DITA XML Map and Topic Elements](#)
- ❖ [How to Write Using DITA XML](#)
- ❖ [How to Reuse Content in DITA XML](#)
- ❖ [Linking in DITA XML](#)
- ❖ [How to Build a DITA XML Technical Writing Portfolio](#)