

Backend APIs

Ingestion API

Ingest Arbitrary Documents

Typical uses for the Ingestion API

A set of backend APIs are provided to take in and process arbitrary document data sent directly to the backend API server. This is generally used for:

Creating arbitrary documents that may not exist in any actual sources but contain useful information.

Programmatically passing in documents to Onyx. This is sometimes simpler than creating a connector.

Editing specific documents in Onyx when the Onyx admin either does not want to or does not have permission to update the original document in the source.

Supplementing existing connector functionalities. For example, passing in README file contents and attributing it to the GitHub or GitLab source type.

Example Document Ingestion

This example creates a new Document in Onyx of the "Web" type. This document will now show up in Onyx's search flows like any other webpage pulled in by a Web connector.



>

```
1 {
2   "document": {
3     "id": "ingestion_document_1",
4     "sections": [
5       {
6         "text": "This is the contents of the document that will be processed and saved into the vector+keyword document index.",
7         "link": "https://docs.danswer.dev/introduction#what-is-danswer"
8       },
9       {
10        "text": "You can include multiple content sections each with their own link or combine them.",
11        "link": "https://docs.danswer.dev/introduction#main-features"
12      }
13    ],
14    "source": "web",
15    "semantic_identifier": "Danswer Ingestion Example",
16    "metadata": {
17      "tag": "informational",
18      "topics": ["danswer", "api"]
19    },
20    "doc_updated_at": "2024-04-25T08:20:00Z"
21  },
22  "cc_pair_id": 1
23 }
```

Body Cookies Headers (4) Test Results Status: 200 OK Time: 1089 ms Size: 186 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "document_id": "ingestion_document_1",
3   "already_existed": true
4 }
```



>

```

"document": {
  "id": "ingestion_document_1",
  "sections": [
    {
      "text": "This is the contents of the document that will be processed and saved into",
      "link": "https://docs.onyx.app/introduction#what-is-onyx"
    },
    {
      "text": "You can include multiple content sections each with their own link or content",
      "link": "https://docs.onyx.app/introduction#main-features"
    }
  ],
  "source": "web",
  "semantic_identifier": "Onyx Ingestion Example",
  "metadata": {
    "tag": "informational",
    "topics": ["onyx", "api"]
  },
  "doc_updated_at": "2024-04-25T08:20:00Z"
},
"cc_pair_id": 1
}'

```

Note: The Bearer auth token is generated on server startup in Onyx MIT. There is better API Key support as part of Onyx EE.

See below for a breakdown of the different fields provided:

id : this is the unique ID of the document, if a document of this ID exists it will be updated/replaced. If not provided, a document ID is generated from the `semantic_identifier` field instead and returned in the response.

sections : list of sections each containing textual content and an optional link. The document chunking tries to avoid splitting sections internally and favors splitting at section borders. Also the link of the document at query time is the link of the best matched section.

source : Source type, full list can be checked by searching for DocumentSource [here](#)

semantic_identifier : This is the "Title" of the document as shown in the UI (see image below)



>

cc_pair_id : This is the “Connector” ID seen on the Connector Status pages. For example, if running locally, it might be `http://localhost:3000/admin/connector/2` . This allows attaching the ingestion doc to existing connectors so they can be assigned to groups or deleted together with the connector. If not provided or set to `1` explicitly, it is considered part of the default catch-all connector.

For even more details, the code for the relevant object is found [here](#), called “DocumentBase”

The screenshot displays the Onyx ingestion API interface. On the left, there are filter sections: 'Filters' with a dropdown arrow, 'Time Range' with a calendar icon and 'Any time...' dropdown, 'Sources' with a globe icon and 'Web' selected, and 'Tags' with a text input 'Find a tag'. Below these is an 'AI Assistant' box with a message: 'This doesn't seem like a question for a Generative AI. Do you still want to have GPT give a response?'. On the right, the 'Danswer' dropdown is set to 'Danswer'. A search bar contains 'Ingestion Document'. Below the search bar, the 'Results' section shows a single result: '0.50 Danswer Ingestion Example'. This result includes metadata: 'Updated 11 days ago', 'tag=informational', and 'topics=danswer,api'. The main content of the result is: 'Danswer Ingestion Example This is the contents of the document that will be processed and saved into the vector+keyword document index. You can include multiple content sections each with their own link or combine them.'

Checking Ingestion Documents



>

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Cookies

Type

Bearer Token

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#)

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization

Token

dn_qODmg9r8NI9PR4R9GF_z1UA0smcwVlc...

Body

Cookies

Headers (4)

Test Results

Status: 200 OK

Time: 25 ms

Size: 271 B

Save as example

Pretty

Raw

Preview

Visualize

JSON

1

[

2

{

3

"document_id": "ingestion_document_1",

4

"semantic_id": "Danswer Ingestion Example",

5

"link": "[https://docs.danswer.dev/introduction/what-is-danswer](\"https://docs.danswer.dev/introduction/what-is-danswer\")"

6

}

7

]

< Custom Tools

Answer with Quote >

Powered by Mintlify