


File search

Allow models to search your files for relevant information before generating a response.

File search is a tool available in the [Responses API](#). It enables models to retrieve information in a knowledge base of previously uploaded files through semantic and keyword search. By creating vector stores and uploading files to them, you can augment the models' inherent knowledge by giving them access to these knowledge bases or `vector_stores`.

 To learn more about how vector stores and semantic search work, refer to our [retrieval guide](#).

This is a hosted tool managed by OpenAI, meaning you don't have to implement code on your end to handle its execution. When the model decides to use it, it will automatically call the tool, retrieve information from your files, and return an output.

How to use

Prior to using file search with the Responses API, you need to have set up a knowledge base in a vector store and uploaded files to it.

> Create a vector store and upload a file

Once your knowledge base is set up, you can include the `file_search` tool in the list of tools available to the model, along with the list of vector stores in which to search.

File search tool

python ↕



```
1 from openai import OpenAI
2 client = OpenAI()
3
4 response = client.responses.create(
5     model="gpt-4.1",
6     input="What is deep research by OpenAI?",
7     tools=[
```

```

8         "type": "file_search",
9         "vector_store_ids": ["<vector_store_id>"]
10    }]
11 )
12 print(response)

```

When this tool is called by the model, you will receive a response with multiple outputs:

- 1 A `file_search_call` output item, which contains the id of the file search call.
- 2 A `message` output item, which contains the response from the model, along with the file citations.

File search response

json 

```

1  {
2    "output": [
3      {
4        "type": "file_search_call",
5        "id": "fs_67c09ccea8c48191ade9367e3ba71515",
6        "status": "completed",
7        "queries": ["What is deep research?"],
8        "search_results": null
9      },
10     {
11       "id": "msg_67c09cd3091c819185af2be5d13d87de",
12       "type": "message",
13       "role": "assistant",
14       "content": [
15         {
16           "type": "output_text",
17           "text": "Deep research is a sophisticated capability that allows for
18           "annotations": [
19             {
20               "type": "file_citation",
21               "index": 992,
22               "file_id": "file-2dtbBZdjtdKS8eqWxqbgDi",
23               "filename": "deep_research_blog.pdf"
24             },
25             {
26               "type": "file_citation",
27               "index": 992,
28               "file_id": "file-2dtbBZdjtdKS8eqWxqbgDi",
29               "filename": "deep_research_blog.pdf"
30             },
31             {
32               "type": "file_citation",

```

```

33         "index": 1176,
34         "file_id": "file-2dtbBZdjtDKS8eqWxqbgDi",
35         "filename": "deep_research_blog.pdf"
36     },
37     {
38         "type": "file_citation",
39         "index": 1176,
40         "file_id": "file-2dtbBZdjtDKS8eqWxqbgDi",
41         "filename": "deep_research_blog.pdf"
42     }
43 ]
44 }
45 ]
46 }
47 ]
48 }

```

Retrieval customization

Limiting the number of results

Using the file search tool with the Responses API, you can customize the number of results you want to retrieve from the vector stores. This can help reduce both token usage and latency, but may come at the cost of reduced answer quality.

Limit the number of results

python ↕ 

```

1 response = client.responses.create(
2     model="gpt-4.1",
3     input="What is deep research by OpenAI?",
4     tools=[{
5         "type": "file_search",
6         "vector_store_ids": ["<vector_store_id>"],
7         "max_num_results": 2
8     }]
9 )
10 print(response)

```

Include search results in the response

While you can see annotations (references to files) in the output text, the file search call will not return search results by default.

To include search results in the response, you can use the `include` parameter when creating the response.

Include search results

python ↕ 

```
1 response = client.responses.create(  
2     model="gpt-4.1",  
3     input="What is deep research by OpenAI?",  
4     tools=[  
5         "type": "file_search",  
6         "vector_store_ids": ["<vector_store_id>"]  
7     ]],  
8     include=["file_search_call.results"]  
9 )  
10 print(response)
```

Metadata filtering

You can filter the search results based on the metadata of the files. For more details, refer to our [retrieval guide](#), which covers:

How to [set attributes on vector store files](#)

How to [define filters](#)

Metadata filtering

python ↕ 

```
1 response = client.responses.create(  
2     model="gpt-4.1",  
3     input="What is deep research by OpenAI?",  
4     tools=[  
5         "type": "file_search",  
6         "vector_store_ids": ["<vector_store_id>"],  
7         "filters": {  
8             "type": "in",  
9             "key": "category",  
10            "value": ["blog", "announcement"]  
11        }  
12    ]]  
13  
14
```

```
)  
print(response)
```

Supported files

For `text/` MIME types, the encoding must be one of `utf-8`, `utf-16`, or `ascii`.

| FILE FORMAT | MIME TYPE |
|--------------------|--|
| <code>.c</code> | <code>text/x-c</code> |
| <code>.cpp</code> | <code>text/x-c++</code> |
| <code>.cs</code> | <code>text/x-csharp</code> |
| <code>.css</code> | <code>text/css</code> |
| <code>.doc</code> | <code>application/msword</code> |
| <code>.docx</code> | <code>application/vnd.openxmlformats-officedocument.wordprocessingml.document</code> |
| <code>.go</code> | <code>text/x-golang</code> |
| <code>.html</code> | <code>text/html</code> |
| <code>.java</code> | <code>text/x-java</code> |
| <code>.js</code> | <code>text/javascript</code> |
| <code>.json</code> | <code>application/json</code> |
| <code>.md</code> | <code>text/markdown</code> |
| <code>.pdf</code> | <code>application/pdf</code> |
| <code>.php</code> | <code>text/x-php</code> |
| <code>.pptx</code> | <code>application/vnd.openxmlformats-officedocument.presentationml.presentation</code> |
| <code>.py</code> | <code>text/x-python</code> |
| <code>.py</code> | <code>text/x-script.python</code> |
| <code>.rb</code> | <code>text/x-ruby</code> |
| <code>.sh</code> | <code>application/x-sh</code> |

| FILE FORMAT | MIME TYPE |
|-------------|------------------------|
| .tex | text/x-tex |
| .ts | application/typescript |
| .txt | text/plain |

Usage notes

| API AVAILABILITY | RATE LIMITS | NOTES |
|--------------------|---------------------|--|
| ✔ Responses | Tier 1 | Pricing |
| ⊗ Chat Completions | 100 RPM | ZDR and data residency |
| ✔ Assistants | Tier 2 and 3 | |
| | 500 RPM | |
| | Tier 4 and 5 | |
| | 1000 RPM | |