OpenAI Assistant API - Core Concepts

Christian Tzolov (christzolov)
https://github.com/logaritex/assistant-api

HAS A
IS A
xxx CLASS

Allows you to describe functions to the Assistants and have it intelligently return the functions that need to be called along with their arguments. The Assistants API will pause execution during a Run when it invokes functions, and you can supply the results of the function call back to continue the Run execution.

Retrieval augments the Assistant with knowledge from outside its model, such as proprietary product information or documents provided by your users. Once a file is uploaded and passed to the Assistant, OpenAI will automatically chunk your documents, index and store the embeddings, and implement vector search to retrieve relevant content to answer user queries.

Allows the Assistants API to write and run **Python** code in a sandboxed execution environment. Can process files with diverse data and formatting, and generate files with data and images of graphs. Run code iteratively to solve **code** and **math problems**.

Model

1 — model

Assistant has **instructions** and can leverage **models**, **tools**, and knowledge to respond to user queries.

Assistant
+id
+name
+instructions
+tools
+file_ids
+metadata

function
retrieval (aka RAG)
code_interpreter

Tool — tools — 0..128

instructions — 1 — **instructions**

Instructions guide how the Assistant and model should behave or respond. **Similar to system messages** in the Chat Completions API.

1 — file_ids — 0..*

Files
Supported Files

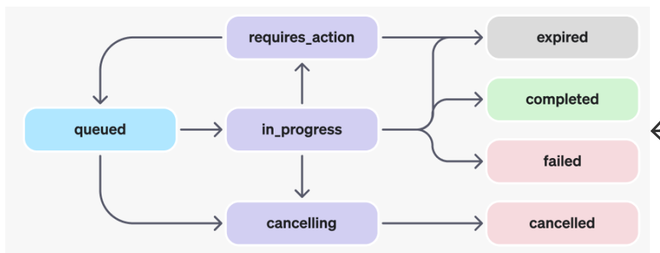
Files are uploaded using the File upload endpoint and must have the **purpose** set to **assistants** to be used with this API. Max of 20 files per Assistant, at most 512 MB each. Total size <=100GB.

0..* — file_ids — file_ids

The files set at Thread and Message level are only accessible in the specific Thread.

Message is created by an Assistant or a User and stored on the Thread. The message includes text, images, and other files.

Run represents an invocation of an **Assistant** on a **Thread**. The Assistant uses it's configuration and the Thread's Messages to perform tasks by calling **models** and **tools**. As part of a Run, the Assistant appends Messages to the Thread.



RunError

0..1

Run
+ id
+thread_id
+assistant_id
+status
+required_action
+metadata

assistant_id — 1

status

thread_id — 1

Thread
+ id
+list<messages>
+list<file_id>
+metadata

thread_id — 1 — 0..*

Message
+ id
+ thread_id
+ (optional) assistant_id
+role
+list<content>
+list<file_id>
+metadata
+metadata

1 — role — **Role**

User
Assistant

required_action — 0..1

Function
+ name
+ arguments

1 — function

TooCall
+ id
+ type
+ function

1..* — submit_tool_outputs — #tool_calls

RequiredAction
+ type
+ submit_tool_outputs

Function definition, including the name of the function and the arguments that the model provides you to pass to the function.

Details on the action required to continue the run. Will be null if no action is required.

Thread is a **conversation** session **between** an **Assistant** and a **User**. Stores Messages and **automatically handle truncation to fit content into a model's context.**

There is no limit to the number of Messages you can store in a Thread. Once the size of the Messages exceeds the context window of the model, the Thread will attempt to include as many messages as possible that fit in the context window and **drop the oldest messages**. **Note that this truncation strategy may evolve over time.**

1..* — content

Content
+ type
+ text
+ Image_file

type — 1 — **Type**

text
image_file

image_file — 0..1 — **ImageFile**
+ file_id

text — 0..1 — **Text**
+ value
+ annotations

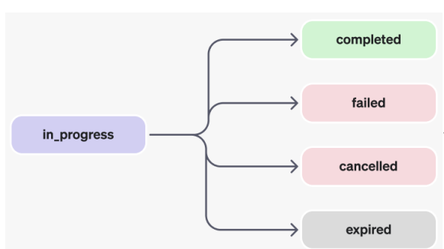
annotations — 0..* — **Annotation**

Mutually exclusive. Depends on the content Type content, only one either the Text or the ImageFile can be presented in the Content.

Annotations provide information around how you should annotate the text in the Message.When present in the Message object, you'll find related placeholders in the message text that have to be replaced with those annotations.

file_citation
Created by the retrieval tool and define references to a specific quote in a specific file that was uploaded and used by the Assistant to generate the response.

file_path
created by the code_interpreter tool and contain references to the files generated by the tool.



status

RunStep
+ assistant_id
+ thread_id
+ run_id
+ type
+ status
+ error
+ details

Details

0..1 — **Error**

RunStep provides a detailed list of steps the **Assistant** took as part of a **Run**, such as **calling tools** or **creating Messages**.