OpenAI Platform

Streaming API responses

Copy page

Learn how to stream model responses from the OpenAI API using server-sent events.

By default, when you make a request to the OpenAI API, we generate the model's entire output before sending it back in a single HTTP response. When generating long outputs, waiting for a response can take time. Streaming responses lets you start printing or processing the beginning of the model's output while it continues generating the full response.

Enable streaming

To start streaming responses, set `stream=True` in your request to the Responses endpoint:

python

```python
from openai import OpenAI
client = OpenAI()

stream = client.responses.create(
    model="gpt-4.1",
    input=[
        {
            "role": "user",
            "content": "Say 'double bubble bath' ten times fast.",
        },
    ],
    stream=True,
)

for event in stream:
    print(event)
```

The Responses API uses semantic events for streaming. Each event is typed with a predefined schema, so you can listen for events you care about.

For a full list of event types, see the API reference for streaming. Here are a few examples:

```
type StreamingEvent =
    | ResponseCreatedEvent
    | ResponseInProgressEvent
    | ResponseFailedEvent
    | ResponseCompletedEvent
    | ResponseOutputItemAdded
    | ResponseOutputItemDone
    | ResponseContentPartAdded
    | ResponseContentPartDone
    | ResponseOutputTextDelta
    | ResponseOutputTextAnnotationAdded
    | ResponseTextDone
    | ResponseRefusalDelta
    | ResponseRefusalDone
    | ResponseFunctionCallArgumentsDelta
    | ResponseFunctionCallArgumentsDone
    | ResponseFileSearchCallInProgress
    | ResponseFileSearchCallSearching
    | ResponseFileSearchCallCompleted
    | ResponseCodeInterpreterInProgress
    | ResponseCodeInterpreterCallCodeDelta
    | ResponseCodeInterpreterCallCodeDone
    | ResponseCodeInterpreterCallIntepreting

 | ResponseCodeInterpreterCallCompleted
 | Error
```

Read the responses

If you're using our SDK, every event is a typed instance. You can also identity individual events using the `type` property of the event.

Some key lifecycle events are emitted only once, while others are emitted multiple times as the response is generated. Common events to listen for when streaming text are:

```
- `response.created`
- `response.output_text.delta`
- `response.completed`
- `error`
```

For a full list of events you can listen for, see the API reference for streaming.

Advanced use cases

For more advanced use cases, like streaming tool calls, check out the following dedicated guides:

- Streaming function calls
- Streaming structured output

Moderation risk

Note that streaming the model's output in a production application makes it more difficult to moderate the content of the completions, as partial completions may be more difficult to evaluate. This may have implications for approved usage.