Realtime Beta

Communicate with a GPT-4o class model in real time using WebRTC or WebSockets. Supports text and audio inputs and ouputs, along with audio transcriptions. Learn more about the Realtime API.

Session tokens

REST API endpoint to generate ephemeral session tokens for use in client-side applications.

Create session

`POST https://api.openai.com/v1/realtime/sessions`

Create an ephemeral API token for use in client-side applications with the Realtime API. Can be configured with the same session parameters as the `session.update` client event.

It responds with a session object, plus a `client_secret` key which contains a usable ephemeral API token that can be used to authenticate browser clients for the Realtime API.

Example request curl

```
curl -X POST https://api.openai.com/v1/real
  -H "Authorization: Bearer $OPENAI_API_KEY
  -H "Content-Type: application/json" \
  -d '{
    "model": "gpt-4o-realtime-preview",
    "modalities": ["audio", "text"],
    "instructions": "You are a friendly ass
  }'
```

Request body

`client_secret` object Optional

Configuration options for the generated client secret.

Show properties

`input_audio_format` string Optional Defaults to pcm16

The format of input audio. Options are `pcm16`, `g711_ulaw`, or `g711_alaw`. For `pcm16`, input audio must be 16-bit PCM at a 24kHz sample rate, single channel (mono), and little-endian byte order.

`input_audio_noise_reduction` object Optional Defaults to null

Configuration for input audio noise reduction. This can be set to `null` to turn off. Noise reduction filters audio added to the input audio buffer before it is sent to VAD and the model. Filtering the audio can improve VAD and turn detection accuracy (reducing false positives) and model performance by improving perception of the input audio.

Show properties

`input_audio_transcription` object Optional

Configuration for input audio transcription, defaults to off and can be set to `null` to turn off once on. Input audio transcription is not native to the model, since the model consumes audio directly. Transcription runs asynchronously through the /audio/transcriptions endpoint and should be treated as guidance

Response

```json
{
  "id": "sess_001",
  "object": "realtime.session",
  "model": "gpt-4o-realtime-preview",
  "modalities": ["audio", "text"],
  "instructions": "You are a friendly assi
  "voice": "alloy",
  "input_audio_format": "pcm16",
  "output_audio_format": "pcm16",
  "input_audio_transcription": {
      "model": "whisper-1"
  },
  "turn_detection": null,
  "tools": [],
  "tool_choice": "none",
  "temperature": 0.7,
  "max_response_output_tokens": 200,
  "speed": 1.1,
  "tracing": "auto",
  "client_secret": {
    "value": "ek_abc123",
    "expires_at": 1234567890
```

of input audio content rather than precisely what the model heard. The client can optionally set the language and prompt for transcription, these offer additional guidance to the transcription service.

Show properties

`instructions` string Optional

The default system instructions (i.e. system message) prepended to model calls. This field allows the client to guide the model on desired responses. The model can be instructed on response content and format, (e.g. "be extremely succinct", "act friendly", "here are examples of good responses") and on audio behavior (e.g. "talk quickly", "inject emotion into your voice", "laugh frequently"). The instructions are not guaranteed to be followed by the model, but they provide guidance to the model on the desired behavior.

Note that the server sets default instructions which will be used if this field is not set and are visible in the `session.created` event at the start of the session.

`max_response_output_tokens` integer or "inf" Optional

Maximum number of output tokens for a single assistant response, inclusive of tool calls. Provide an integer between 1 and 4096 to limit output tokens, or `inf` for the maximum available tokens for a given model. Defaults to `inf`.

`modalities` Optional

The set of modalities the model can respond with. To disable audio, set this to ["text"].

`model` string Optional

The Realtime model used for this session.

`output_audio_format` string Optional Defaults to pcm16

The format of output audio. Options are `pcm16`, `g711_ulaw`, or `g711_alaw`. For `pcm16`, output audio is sampled at a rate of 24kHz.

`speed` number Optional Defaults to 1

The speed of the model's spoken response. 1.0 is the default speed. 0.25 is the minimum speed. 1.5 is the maximum speed. This value can only be changed in between model turns, not while a response is in progress.

`temperature` number Optional Defaults to 0.8

Sampling temperature for the model, limited to [0.6, 1.2]. For audio models a temperature of 0.8 is highly recommended for best performance.

`tool_choice` string Optional Defaults to auto

How the model chooses tools. Options are `auto`, `none`, `required`, or specify a function.

`tools` array Optional

Tools (functions) available to the model.

Show properties

`tracing` "auto" or object Optional

Configuration options for tracing. Set to null to disable tracing. Once tracing is enabled for a session, the configuration cannot be modified.

`auto` will create a trace for the session with default values for the workflow name, group id, and metadata.

Show possible types

`turn_detection` object Optional

Configuration for turn detection, ether Server VAD or Semantic VAD. This can be set to `null` to turn off, in which case the client must manually trigger model response. Server VAD means that the model will detect the start and end of speech based on audio volume and respond at the end of user speech. Semantic VAD is more advanced and uses a turn detection model (in conjuction with VAD) to semantically estimate whether the user has finished speaking, then dynamically sets a timeout based on this probability. For example, if user audio trails off with "uhhm", the model will score a low probability of turn end and wait longer for the user to continue speaking. This can be useful for more natural conversations, but may have a higher latency.

Show properties

`voice` string Optional

The voice the model uses to respond. Voice cannot be changed during the session once the model has responded with audio at least once. Current voice options are `alloy`, `ash`, `ballad`, `coral`, `echo`, `sage`, `shimmer`, and `verse`.

Returns

The created Realtime session object, plus an ephemeral key

Create transcription session

```
POST https://api.openai.com/v1/realtime/transcription_sessions
```

Create an ephemeral API token for use in client-side applications with the Realtime API specifically for realtime transcriptions. Can be configured with the same session parameters as the `transcription_session.update` client event.
It responds with a session object, plus a `client_secret` key which contains a usable ephemeral API token that can be used to authenticate browser clients for the Realtime API.

Example request curl

```
curl -X POST https://api.openai.com/v1/real
  -H "Authorization: Bearer $OPENAI_API_KEY
  -H "Content-Type: application/json" \
  -d '{}'
```

Response

```
{
  "id": "sess_BBwZc7cFV3XizEyKGDCGL",
  "object": "realtime.transcription_sessio
```

Request body

`client_secret` object Optional

Configuration options for the generated client secret.

Show properties

`include` array Optional

The set of items to include in the transcription. Current available items are:

null.

`input_audio_format` string Optional Defaults to pcm16

The format of input audio. Options are `pcm16`, `g711_ulaw`, or `g711_alaw`. For `pcm16`, input audio must be 16-bit PCM at a 24kHz sample rate, single channel (mono), and little-endian byte order.

`input_audio_noise_reduction` object Optional Defaults to null

Configuration for input audio noise reduction. This can be set to `null` to turn off. Noise reduction filters audio added to the input audio buffer before it is sent to VAD and the model. Filtering the audio can improve VAD and turn detection accuracy (reducing false positives) and model performance by improving perception of the input audio.

Show properties

```
  "modalities": ["audio", "text"],
  "turn_detection": {
    "type": "server_vad",
    "threshold": 0.5,
    "prefix_padding_ms": 300,
    "silence_duration_ms": 200
  },
  "input_audio_format": "pcm16",
  "input_audio_transcription": {
    "model": "gpt-4o-transcribe",
    "language": null,
    "prompt": ""
  },
  "client_secret": null
}
```

`input_audio_transcription` object Optional

Configuration for input audio transcription. The client can optionally set the language and prompt for transcription, these offer additional guidance to the transcription service.

Show properties

`modalities` Optional

The set of modalities the model can respond with. To disable audio, set this to ["text"].

`turn_detection` object Optional

Configuration for turn detection, ether Server VAD or Semantic VAD. This can be set to `null` to turn off, in which case the client must manually trigger model response. Server VAD means that the model will detect the start and end of speech based on audio volume and respond at the end of user speech. Semantic VAD is more advanced and uses a turn detection model (in conjuction with VAD) to semantically estimate whether the user has finished speaking, then dynamically sets a timeout based on this probability. For example, if user audio trails off with "uhhm", the model will score a low probability of turn end and wait longer for the user to continue speaking. This can be useful for more natural conversations, but may have a higher latency.

Show properties

Returns

The created Realtime transcription session object, plus an ephemeral key

The session object

A new Realtime session configuration, with an ephermeral key. Default TTL for keys is one minute.

`client_secret` object

Ephemeral key returned by the API.

Show properties

`input_audio_format` string

The format of input audio. Options are `pcm16`, `g711_ulaw`, or `g711_alaw`.

`input_audio_transcription` object

Configuration for input audio transcription, defaults to off and can be set to `null` to turn off once on. Input audio transcription is not native to the model, since the model consumes audio directly.

OBJECT The session object

```
{
  "id": "sess_001",
  "object": "realtime.session",
  "model": "gpt-4o-realtime-preview",
  "modalities": ["audio", "text"],
  "instructions": "You are a friendly assi
  "voice": "alloy",
  "input_audio_format": "pcm16",
  "output_audio_format": "pcm16",
  "input_audio_transcription": {
      "model": "whisper-1"
  },
  "turn_detection": null,
  "tools": [],
  "tool_choice": "none",
  "temperature": 0.7,
```

Transcription runs asynchronously and should be treated as rough guidance rather than the representation understood by the model.

Show properties

`instructions` string

The default system instructions (i.e. system message) prepended to model calls. This field allows the client to guide the model on desired responses. The model can be instructed on response content and format, (e.g. "be extremely succinct", "act friendly", "here are examples of good responses") and on audio behavior (e.g. "talk quickly", "inject emotion into your voice", "laugh frequently"). The instructions are not guaranteed to be followed by the model, but they provide guidance to the model on the desired behavior.

Note that the server sets default instructions which will be used if this field is not set and are visible in the `session.created` event at the start of the session.

`max_response_output_tokens` integer or "inf"

Maximum number of output tokens for a single assistant response, inclusive of tool calls. Provide an integer between 1 and 4096 to limit output tokens, or `inf` for the maximum available tokens for a given model. Defaults to `inf`.

`modalities`

The set of modalities the model can respond with. To disable audio, set this to ["text"].

```
  "speed": 1.1,
  "tracing": "auto",
  "max_response_output_tokens": 200,
  "client_secret": {
    "value": "ek_abc123",
    "expires_at": 1234567890
  }
}
```

`output_audio_format` string

The format of output audio. Options are `pcm16`, `g711_ulaw`, or `g711_alaw`.

`speed` number

The speed of the model's spoken response. 1.0 is the default speed. 0.25 is the minimum speed. 1.5 is the maximum speed. This value can only be changed in between model turns, not while a response is in progress.

`temperature` number

Sampling temperature for the model, limited to [0.6, 1.2]. Defaults to 0.8.

`tool_choice` string

How the model chooses tools. Options are `auto`, `none`, `required`, or specify a function.

`tools` array

Tools (functions) available to the model.

Show properties

`tracing` "auto" or object

Configuration options for tracing. Set to null to disable tracing. Once tracing is enabled for a session, the configuration cannot be modified.

`auto` will create a trace for the session with default values for the workflow name, group id, and metadata.

Show possible types

`turn_detection` object

Configuration for turn detection. Can be set to `null` to turn off. Server VAD means that the model will detect the start and end of speech based on audio volume and respond at the end of user speech.

Show properties

`voice` string

The voice the model uses to respond. Voice cannot be changed during the session once the model has responded with audio at least once. Current voice options are `alloy`, `ash`, `ballad`, `coral`, `echo`, `sage`, `shimmer`, and `verse`.

The transcription session object

A new Realtime transcription session configuration.

When a session is created on the server via REST API, the session object also contains an ephemeral key. Default TTL for keys is 10 minutes. This property is not present when a

OBJECT The transcription session object

```
{
  "id": "sess_BBwZc7cFV3XizEyKGDCGL",
  "object": "realtime.transcription_sessio
```

session is updated via the WebSocket API.

`client_secret` object

Ephemeral key returned by the API. Only present when the session is created on the server via REST API.

Show properties

`input_audio_format` string

The format of input audio. Options are `pcm16`, `g711_ulaw`, or `g711_alaw`.

`input_audio_transcription` object

Configuration of the transcription model.

Show properties

`modalities`

The set of modalities the model can respond with. To disable audio, set this to ["text"].

`turn_detection` object

Configuration for turn detection. Can be set to `null` to turn off. Server VAD means that the model will detect the start and end of speech based on audio volume and respond at the end of user speech.

Show properties

```json
    "expires_at": 1742188264,
    "modalities": ["audio", "text"],
    "turn_detection": {
      "type": "server_vad",
      "threshold": 0.5,
      "prefix_padding_ms": 300,
      "silence_duration_ms": 200
    },
    "input_audio_format": "pcm16",
    "input_audio_transcription": {
      "model": "gpt-4o-transcribe",
      "language": null,
      "prompt": ""
    },
    "client_secret": null
  }
```

Client events

These are events that the OpenAI Realtime WebSocket server will accept from the client.

session.update

Send this event to update the session's default configuration. The client may send this event at any time to update any field, except for `voice`. However, note that once a session has been initialized with a particular `model`, it can't be changed to another model using `session.update`.

When the server receives a `session.update`, it will respond with a `session.updated` event showing the full, effective configuration. Only the fields that are present are updated. To clear a field like `instructions`, pass an empty string.

OBJECT session.update

```
{
    "event_id": "event_123",
    "type": "session.update",
    "session": {
        "modalities": ["text", "audio"],
        "instructions": "You are a helpful
        "voice": "sage",
        "input_audio_format": "pcm16",
        "output_audio_format": "pcm16",
        "input_audio_transcription": {
            "model": "whisper-1"
```

event_id string

Optional client-generated ID used to identify this event.

session object

Realtime session object configuration.

Show properties

type string

The event type, must be `session.update` .

```
        },
        "turn_detection": {
            "type": "server_vad",
            "threshold": 0.5,
            "prefix_padding_ms": 300,
            "silence_duration_ms": 500,
            "create_response": true
        },
        "tools": [
            {
                "type": "function",
                "name": "get_weather",
                "description": "Get the cu
                "parameters": {
                    "type": "object",
                    "properties": {
                        "location": { "typ
                    },
                    "required": ["location
                }
            }
        ],
        "tool_choice": "auto",
```

input_audio_buffer.append

Send this event to append audio bytes to the input audio buffer. The audio buffer is temporary storage you can write to and later commit. In Server VAD mode, the audio buffer is used to detect speech and the server will decide when to commit. When Server VAD is disabled, you must commit the audio buffer manually.

The client may choose how much audio to place in each event up to a maximum of 15 MiB, for example streaming smaller chunks from the client may allow the VAD to be more responsive. Unlike made other client events, the server will not send a confirmation response to this event.

`audio` string
Base64-encoded audio bytes. This must be in the format specified by the `input_audio_format` field in the session configuration.

`event_id` string
Optional client-generated ID used to identify this event.

`type` string
The event type, must be `input_audio_buffer.append`.

OBJECT input_audio_buffer.append

```json
{
    "event_id": "event_456",
    "type": "input_audio_buffer.append",
    "audio": "Base64EncodedAudioData"
}
```

input_audio_buffer.commit

Send this event to commit the user input audio buffer, which will create a new user message item in the conversation. This event will produce an error if the input audio buffer is empty. When in Server VAD mode, the client does not need to send this event, the server will commit the audio buffer automatically.

Committing the input audio buffer will trigger input audio transcription (if enabled in session configuration), but it will not create a response from the model. The server will respond with an `input_audio_buffer.committed` event.

`event_id` string

Optional client-generated ID used to identify this event.

`type` string

The event type, must be `input_audio_buffer.commit`.

OBJECT input_audio_buffer.commit

```json
{
    "event_id": "event_789",
    "type": "input_audio_buffer.commit"
}
```

input_audio_buffer.clear

Send this event to clear the audio bytes in the buffer. The server will respond with an `input_audio_buffer.cleared` event.

`event_id` string
Optional client-generated ID used to identify this event.

`type` string
The event type, must be `input_audio_buffer.clear`.

OBJECT input_audio_buffer.clear

```
{
    "event_id": "event_012",
    "type": "input_audio_buffer.clear"
}
```

conversation.item.create

Add a new Item to the Conversation's context, including messages, function calls, and function call responses. This event can be used both to populate a "history" of the

OBJECT conversation.item.create

```
{
    "event_id": "event_345",
```

conversation and to add new items mid-stream, but has the current limitation that it cannot populate assistant audio messages.

If successful, the server will respond with a `conversation.item.created` event, otherwise an `error` event will be sent.

`event_id` string
Optional client-generated ID used to identify this event.

`item` object
The item to add to the conversation.

Show properties

`previous_item_id` string
The ID of the preceding item after which the new item will be inserted. If not set, the new item will be appended to the end of the conversation. If set to `root`, the new item will be added to the beginning of the conversation. If set to an existing ID, it allows an item to be inserted mid-conversation. If the ID cannot be found, an error will be returned and the item will not be added.

`type` string
The event type, must be `conversation.item.create`.

```
    "type": "conversation.item.create",
    "previous_item_id": null,
    "item": {
        "id": "msg_001",
        "type": "message",
        "role": "user",
        "content": [
            {
                "type": "input_text",
                "text": "Hello, how are yo
            }
        ]
    }
}
```

conversation.item.retrieve

Send this event when you want to retrieve the server's representation of a specific item in the conversation history. This is useful, for example, to inspect user audio after noise cancellation and VAD. The server will respond with a `conversation.item.retrieved` event, unless the item does not exist in the conversation history, in which case the server will respond with an error.

OBJECT conversation.item.retrieve

```
{
    "event_id": "event_901",
    "type": "conversation.item.retrieve",
    "item_id": "msg_003"
}
```

`event_id` string

Optional client-generated ID used to identify this event.

`item_id` string

The ID of the item to retrieve.

`type` string

The event type, must be `conversation.item.retrieve`.

conversation.item.truncate

Send this event to truncate a previous assistant message's audio. The server will produce audio faster than realtime, so this event is useful when the user interrupts to truncate audio that has already been sent to the client but not yet played. This will synchronize the server's understanding of the audio with the client's playback.

Truncating audio will delete the server-side text transcript to ensure there is not text in the context that hasn't been heard by the user.

If successful, the server will respond with a `conversation.item.truncated` event.

`audio_end_ms` integer

Inclusive duration up to which audio is truncated, in milliseconds. If the audio_end_ms is greater than the actual audio duration, the server will respond with an error.

`content_index` integer

The index of the content part to truncate. Set this to 0.

`event_id` string

Optional client-generated ID used to identify this event.

OBJECT conversation.item.truncate

```json
{
    "event_id": "event_678",
    "type": "conversation.item.truncate",
    "item_id": "msg_002",
    "content_index": 0,
    "audio_end_ms": 1500
}
```

`item_id` string
The ID of the assistant message item to truncate. Only assistant message items can be truncated.

`type` string
The event type, must be `conversation.item.truncate`.

conversation.item.delete

Send this event when you want to remove any item from the conversation history. The server will respond with a `conversation.item.deleted` event, unless the item does not exist in the conversation history, in which case the server will respond with an error.

OBJECT conversation.item.delete

```
{
    "event_id": "event_901",
    "type": "conversation.item.delete",
    "item_id": "msg_003"
}
```

`event_id` string

Optional client-generated ID used to identify this event.

`item_id` string

The ID of the item to delete.

`type` string

The event type, must be `conversation.item.delete`.

response.create

This event instructs the server to create a Response, which means triggering model inference. When in Server VAD

OBJECT response.create

mode, the server will create Responses automatically.

A Response will include at least one Item, and may have two, in which case the second will be a function call. These Items will be appended to the conversation history.

The server will respond with a `response.created` event, events for Items and content created, and finally a `response.done` event to indicate the Response is complete.

The `response.create` event includes inference configuration like `instructions`, and `temperature`. These fields will override the Session's configuration for this Response only.

`event_id` string

Optional client-generated ID used to identify this event.

`response` object

Create a new Realtime response with these parameters

Show properties

`type` string

The event type, must be `response.create`.

```json
{
    "event_id": "event_234",
    "type": "response.create",
    "response": {
        "modalities": ["text", "audio"],
        "instructions": "Please assist the
        "voice": "sage",
        "output_audio_format": "pcm16",
        "tools": [
            {
                "type": "function",
                "name": "calculate_sum",
                "description": "Calculates
                "parameters": {
                    "type": "object",
                    "properties": {
                        "a": { "type": "nu
                        "b": { "type": "nu
                    },
                    "required": ["a", "b"]
                }
            }
        ],
        "tool_choice": "auto",
        "temperature": 0.8,
        "max_output_tokens": 1024
    }
}
```

response.cancel

Send this event to cancel an in-progress response. The server will respond with a `response.cancelled` event or an error if there is no response to cancel.

`event_id` string
Optional client-generated ID used to identify this event.

`response_id` string
A specific response ID to cancel - if not provided, will cancel an in-progress response in the default conversation.

`type` string
The event type, must be `response.cancel`.

OBJECT response.cancel

```
{
    "event_id": "event_567",
    "type": "response.cancel"
}
```

transcription_session.update

Send this event to update a transcription session.

`event_id` string

Optional client-generated ID used to identify this event.

`session` object

Realtime transcription session object configuration.

Show properties

`type` string

The event type, must be `transcription_session.update`.

OBJECT transcription_session.update

```
{
  "type": "transcription_session.update",
  "session": {
    "input_audio_format": "pcm16",
    "input_audio_transcription": {
      "model": "gpt-4o-transcribe",
      "prompt": "",
      "language": ""
    },
    "turn_detection": {
      "type": "server_vad",
      "threshold": 0.5,
      "prefix_padding_ms": 300,
      "silence_duration_ms": 500,
      "create_response": true,
    },
    "input_audio_noise_reduction": {
      "type": "near_field"
    },
    "include": [
      "item.input_audio_transcription.logp
    ]
  }
}
```

output_audio_buffer.clear

WebRTC Only: Emit to cut off the current audio response. This will trigger the server to stop generating audio and emit a `output_audio_buffer.cleared` event. This event should be preceded by a `response.cancel` client event to stop the generation of the current response. Learn more.

`event_id` string

The unique ID of the client event used for error handling.

`type` string

The event type, must be `output_audio_buffer.clear`.

OBJECT output_audio_buffer.clear

```json
{
    "event_id": "optional_client_event_id",
    "type": "output_audio_buffer.clear"
}
```

Server events

These are events emitted from the OpenAI Realtime WebSocket server to the client.

error

Returned when an error occurs, which could be a client problem or a server problem. Most errors are recoverable and the session will stay open, we recommend to implementors to monitor and log error messages by default.

`error` object

Details of the error.

Show properties

`event_id` string

The unique ID of the server event.

`type` string

The event type, must be `error`.

OBJECT error

```
{
    "event_id": "event_890",
    "type": "error",
    "error": {
        "type": "invalid_request_error",
        "code": "invalid_event",
        "message": "The 'type' field is mi
        "param": null,
        "event_id": "event_567"
    }
}
```

session.created

Returned when a Session is created. Emitted automatically when a new connection is established as the first server event. This event will contain the default Session configuration.

`event_id` string

The unique ID of the server event.

`session` object

Realtime session object configuration.

Show properties

`type` string

The event type, must be `session.created`.

OBJECT session.created

```json
{
    "event_id": "event_1234",
    "type": "session.created",
    "session": {
        "id": "sess_001",
        "object": "realtime.session",
        "model": "gpt-4o-realtime-preview"
        "modalities": ["text", "audio"],
        "instructions": "...model instruct
        "voice": "sage",
        "input_audio_format": "pcm16",
        "output_audio_format": "pcm16",
        "input_audio_transcription": null,
        "turn_detection": {
            "type": "server_vad",
            "threshold": 0.5,
            "prefix_padding_ms": 300,
            "silence_duration_ms": 200
        },
        "tools": [],
        "tool_choice": "auto",
        "temperature": 0.8,
        "max_response_output_tokens": "inf
        "speed": 1.1,
        "tracing": "auto"
    }
}
```

session.updated

Returned when a session is updated with a `session.update` event, unless there is an error.

`event_id` string

The unique ID of the server event.

`session` object

Realtime session object configuration.

Show properties

`type` string

The event type, must be `session.updated`.

OBJECT session.updated

```
{
    "event_id": "event_5678",
    "type": "session.updated",
    "session": {
        "id": "sess_001",
        "object": "realtime.session",
        "model": "gpt-4o-realtime-preview"
        "modalities": ["text"],
        "instructions": "New instructions"
        "voice": "sage",
        "input_audio_format": "pcm16",
        "output_audio_format": "pcm16",
        "input_audio_transcription": {
            "model": "whisper-1"
        },
        "turn_detection": null,
        "tools": [],
        "tool_choice": "none",
        "temperature": 0.7,
        "max_response_output_tokens": 200,
        "speed": 1.1,
        "tracing": "auto"
    }
}
```

conversation.created

Returned when a conversation is created. Emitted right after session creation.

`conversation` object

The conversation resource.

Show properties

`event_id` string

The unique ID of the server event.

`type` string

The event type, must be `conversation.created`.

OBJECT conversation.created

```json
{
    "event_id": "event_9101",
    "type": "conversation.created",
    "conversation": {
        "id": "conv_001",
        "object": "realtime.conversation"
    }
}
```

conversation.item.created

Returned when a conversation item is created. There are several scenarios that produce this event:

- The server is generating a Response, which if successful will produce either one or two Items, which will be of type `message` (role `assistant`) or type `function_call`.
- The input audio buffer has been committed, either by the client or the server (in `server_vad` mode). The server will take the content of the input audio buffer and add it to a new user message Item.
- The client has sent a `conversation.item.create` event to add a new Item to the Conversation.

OBJECT conversation.item.created

```json
{
    "event_id": "event_1920",
    "type": "conversation.item.created",
    "previous_item_id": "msg_002",
    "item": {
        "id": "msg_003",
        "object": "realtime.item",
        "type": "message",
        "status": "completed",
        "role": "user",
        "content": []
    }
}
```

event_id string

The unique ID of the server event.

item object

The item to add to the conversation.

Show properties

previous_item_id string or null

The ID of the preceding item in the Conversation context, allows the client to understand the order of the conversation. Can be `null` if the item has no predecessor.

`type` string

The event type, must be `conversation.item.created`.

---

conversation.item.retrieved

Returned when a conversation item is retrieved with `conversation.item.retrieve`.

`event_id` string

The unique ID of the server event.

`item` object

The item to add to the conversation.

Show properties

`type` string

The event type, must be `conversation.item.retrieved`.

OBJECT conversation.item.retrieved

```
{
    "event_id": "event_1920",
    "type": "conversation.item.created",
    "previous_item_id": "msg_002",
    "item": {
        "id": "msg_003",
        "object": "realtime.item",
        "type": "message",
        "status": "completed",
        "role": "user",
        "content": [
            {
                "type": "input_audio",
                "transcript": "hello how a
                "audio": "base64encodedaud
            }
        ]
    }
}
```

conversation.item.input_audio_transcription.completed

This event is the output of audio transcription for user audio written to the user audio buffer. Transcription begins when the input audio buffer is committed by the client or server (in `server_vad` mode). Transcription runs asynchronously with Response creation, so this event may come before or after the Response events.

Realtime API models accept audio natively, and thus input transcription is a separate process run on a separate ASR (Automatic Speech Recognition) model. The transcript may diverge somewhat from the model's interpretation, and should be treated as a rough guide.

`content_index` integer

The index of the content part containing the audio.

`event_id` string

The unique ID of the server event.

`item_id` string

The ID of the user message item containing the audio.

`logprobs` array or null

The log probabilities of the transcription.

Show properties

OBJECT conversation.item.input_audio_transc…

```json
{
    "event_id": "event_2122",
    "type": "conversation.item.input_audio
    "item_id": "msg_003",
    "content_index": 0,
    "transcript": "Hello, how are you?",
    "usage": {
      "type": "tokens",
      "total_tokens": 48,
      "input_tokens": 38,
      "input_token_details": {
        "text_tokens": 10,
        "audio_tokens": 28,
      },
      "output_tokens": 10,
    }
}
```

`transcript` string

The transcribed text.

`type` string

The event type, must be `conversation.item.input_audio_transcription.completed`.

`usage` object

Usage statistics for the transcription.

Show possible types

conversation.item.input_audio_transcription.delta

Returned when the text value of an input audio transcription content part is updated.

`content_index` integer

The index of the content part in the item's content array.

`delta` string

The text delta.

`event_id` string

The unique ID of the server event.

`item_id` string

The ID of the item.

`logprobs` array or null

The log probabilities of the transcription.

Show properties

`type` string

The event type, must be `conversation.item.input_audio_transcription.delta`.

OBJECT conversation.item.input_audio_transc…

```
{
  "type": "conversation.item.input_audio_tr
  "event_id": "event_001",
  "item_id": "item_001",
  "content_index": 0,
  "delta": "Hello"
}
```

conversation.item.input_audio_transcription.failed

Returned when input audio transcription is configured, and a transcription request for a user message failed. These events are separate from other `error` events so that the client can identify the related Item.

`content_index` integer

The index of the content part containing the audio.

`error` object

Details of the transcription error.

Show properties

`event_id` string

The unique ID of the server event.

`item_id` string

The ID of the user message item.

`type` string

The event type, must be `conversation.item.input_audio_transcription.failed`.

OBJECT conversation.item.input_audio_transc…

```
{
    "event_id": "event_2324",
    "type": "conversation.item.input_audio
    "item_id": "msg_003",
    "content_index": 0,
    "error": {
        "type": "transcription_error",
        "code": "audio_unintelligible",
        "message": "The audio could not be
        "param": null
    }
}
```

conversation.item.truncated

Returned when an earlier assistant audio message item is truncated by the client with a `conversation.item.truncate` event. This event is used to synchronize the server's understanding of the audio with the client's playback.

This action will truncate the audio and remove the server-side text transcript to ensure there is no text in the context that hasn't been heard by the user.

`audio_end_ms` integer
The duration up to which the audio was truncated, in milliseconds.

`content_index` integer
The index of the content part that was truncated.

`event_id` string
The unique ID of the server event.

`item_id` string
The ID of the assistant message item that was truncated.

`type` string
The event type, must be `conversation.item.truncated`.

OBJECT conversation.item.truncated

```
{
    "event_id": "event_2526",
    "type": "conversation.item.truncated",
    "item_id": "msg_004",
    "content_index": 0,
    "audio_end_ms": 1500
}
```

conversation.item.deleted

Returned when an item in the conversation is deleted by the client with a `conversation.item.delete` event. This event is used to synchronize the server's understanding of the conversation history with the client's view.

`event_id` string
The unique ID of the server event.

`item_id` string
The ID of the item that was deleted.

`type` string
The event type, must be `conversation.item.deleted`.

OBJECT conversation.item.deleted

```
{
    "event_id": "event_2728",
    "type": "conversation.item.deleted",
    "item_id": "msg_005"
}
```

input_audio_buffer.committed

Returned when an input audio buffer is committed, either by the client or automatically in server VAD mode. The `item_id` property is the ID of the user message item that will be created, thus a `conversation.item.created` event will also be sent to the client.

`event_id` string

The unique ID of the server event.

`item_id` string

The ID of the user message item that will be created.

`previous_item_id` string or null

The ID of the preceding item after which the new item will be inserted. Can be `null` if the item has no predecessor.

`type` string

The event type, must be `input_audio_buffer.committed`.

OBJECT input_audio_buffer.committed

```json
{
    "event_id": "event_1121",
    "type": "input_audio_buffer.committed",
    "previous_item_id": "msg_001",
    "item_id": "msg_002"
}
```

input_audio_buffer.cleared

Returned when the input audio buffer is cleared by the client with a `input_audio_buffer.clear` event.

`event_id` string
The unique ID of the server event.

`type` string
The event type, must be `input_audio_buffer.cleared`.

OBJECT input_audio_buffer.cleared

```
{
    "event_id": "event_1314",
    "type": "input_audio_buffer.cleared"
}
```

input_audio_buffer.speech_started

Sent by the server when in `server_vad` mode to indicate that speech has been detected in the audio buffer. This can happen any time audio is added to the buffer (unless speech is already detected). The client may want to use this event to interrupt audio playback or provide visual feedback to the user.

The client should expect to receive a `input_audio_buffer.speech_stopped` event when speech

OBJECT input_audio_buffer.speech_started

```
{
    "event_id": "event_1516",
    "type": "input_audio_buffer.speech_star
    "audio_start_ms": 1000,
    "item_id": "msg_003"
}
```

stops. The `item_id` property is the ID of the user message item that will be created when speech stops and will also be included in the `input_audio_buffer.speech_stopped` event (unless the client manually commits the audio buffer during VAD activation).

`audio_start_ms` integer
Milliseconds from the start of all audio written to the buffer during the session when speech was first detected. This will correspond to the beginning of audio sent to the model, and thus includes the `prefix_padding_ms` configured in the Session.

`event_id` string
The unique ID of the server event.

`item_id` string
The ID of the user message item that will be created when speech stops.

`type` string
The event type, must be `input_audio_buffer.speech_started`.

input_audio_buffer.speech_stopped

Returned in `server_vad` mode when the server detects the end of speech in the audio buffer. The server will also send an `conversation.item.created` event with the user message item that is created from the audio buffer.

`audio_end_ms` integer

Milliseconds since the session started when speech stopped. This will correspond to the end of audio sent to the model, and thus includes the `min_silence_duration_ms` configured in the Session.

`event_id` string

The unique ID of the server event.

`item_id` string

The ID of the user message item that will be created.

`type` string

The event type, must be `input_audio_buffer.speech_stopped`.

OBJECT input_audio_buffer.speech_stopped

```
{
    "event_id": "event_1718",
    "type": "input_audio_buffer.speech_stop
    "audio_end_ms": 2000,
    "item_id": "msg_003"
}
```

response.created

Returned when a new Response is created. The first event of response creation, where the response is in an initial state of `in_progress`.

`event_id` string
The unique ID of the server event.

`response` object
The response resource.

Show properties

`type` string
The event type, must be `response.created`.

OBJECT response.created

```
{
    "event_id": "event_2930",
    "type": "response.created",
    "response": {
        "id": "resp_001",
        "object": "realtime.response",
        "status": "in_progress",
        "status_details": null,
        "output": [],
        "usage": null
    }
}
```

response.done

Returned when a Response is done streaming. Always emitted, no matter the final state. The Response object included in the `response.done` event will include all output

OBJECT response.done

```
{
    "event_id": "event_3132",
```

Items in the Response but will omit the raw audio data.

`event_id` string

The unique ID of the server event.

`response` object

The response resource.

Show properties

`type` string

The event type, must be `response.done`.

```
    "type": "response.done",
    "response": {
        "id": "resp_001",
        "object": "realtime.response",
        "status": "completed",
        "status_details": null,
        "output": [
            {
                "id": "msg_006",
                "object": "realtime.item",
                "type": "message",
                "status": "completed",
                "role": "assistant",
                "content": [
                    {
                        "type": "text",
                        "text": "Sure, how
                    }
                ]
            }
        ],
        "usage": {
            "total_tokens":275,
            "input_tokens":127,
            "output_tokens":148,
            "input_token_details": {
                "cached_tokens":384,
                "text_tokens":119,
                "audio_tokens":8,
                "cached_tokens_details": {
                    "text_tokens": 128,
```

response.output_item.added

Returned when a new Item is created during Response generation.

`event_id` string

The unique ID of the server event.

`item` object

The item to add to the conversation.

Show properties

`output_index` integer

The index of the output item in the Response.

`response_id` string

The ID of the Response to which the item belongs.

`type` string

The event type, must be `response.output_item.added`.

OBJECT response.output_item.added

```
{
    "event_id": "event_3334",
    "type": "response.output_item.added",
    "response_id": "resp_001",
    "output_index": 0,
    "item": {
        "id": "msg_007",
        "object": "realtime.item",
        "type": "message",
        "status": "in_progress",
        "role": "assistant",
        "content": []
    }
}
```

response.output_item.done

Returned when an Item is done streaming. Also emitted when a Response is interrupted, incomplete, or cancelled.

`event_id` string

The unique ID of the server event.

`item` object

The item to add to the conversation.

Show properties

`output_index` integer

The index of the output item in the Response.

`response_id` string

The ID of the Response to which the item belongs.

`type` string

The event type, must be `response.output_item.done`.

OBJECT response.output_item.done

```
{
    "event_id": "event_3536",
    "type": "response.output_item.done",
    "response_id": "resp_001",
    "output_index": 0,
    "item": {
        "id": "msg_007",
        "object": "realtime.item",
        "type": "message",
        "status": "completed",
        "role": "assistant",
        "content": [
            {
                "type": "text",
                "text": "Sure, I can help
            }
        ]
    }
}
```

response.content_part.added

Returned when a new content part is added to an assistant message item during response generation.

`content_index` integer
The index of the content part in the item's content array.

`event_id` string
The unique ID of the server event.

`item_id` string
The ID of the item to which the content part was added.

`output_index` integer
The index of the output item in the response.

`part` object
The content part that was added.

Show properties

`response_id` string
The ID of the response.

`type` string
The event type, must be `response.content_part.added`.

OBJECT response.content_part.added

```json
{
    "event_id": "event_3738",
    "type": "response.content_part.added",
    "response_id": "resp_001",
    "item_id": "msg_007",
    "output_index": 0,
    "content_index": 0,
    "part": {
        "type": "text",
        "text": ""
    }
}
```

response.content_part.done

Returned when a content part is done streaming in an assistant message item. Also emitted when a Response is interrupted, incomplete, or cancelled.

`content_index` integer
The index of the content part in the item's content array.

`event_id` string
The unique ID of the server event.

`item_id` string
The ID of the item.

`output_index` integer
The index of the output item in the response.

`part` object
The content part that is done.

Show properties

`response_id` string

OBJECT response.content_part.done

```json
{
    "event_id": "event_3940",
    "type": "response.content_part.done",
    "response_id": "resp_001",
    "item_id": "msg_007",
    "output_index": 0,
    "content_index": 0,
    "part": {
        "type": "text",
        "text": "Sure, I can help with tha
    }
}
```

The ID of the response.

`type` string

The event type, must be `response.content_part.done`.

response.text.delta

Returned when the text value of a "text" content part is updated.

content_index integer
The index of the content part in the item's content array.

delta string
The text delta.

event_id string
The unique ID of the server event.

item_id string
The ID of the item.

output_index integer
The index of the output item in the response.

response_id string
The ID of the response.

type string
The event type, must be `response.text.delta`.

OBJECT response.text.delta

```json
{
    "event_id": "event_4142",
    "type": "response.text.delta",
    "response_id": "resp_001",
    "item_id": "msg_007",
    "output_index": 0,
    "content_index": 0,
    "delta": "Sure, I can h"
}
```

response.text.done

Returned when the text value of a "text" content part is done streaming. Also emitted when a Response is interrupted, incomplete, or cancelled.

`content_index` integer

The index of the content part in the item's content array.

`event_id` string

The unique ID of the server event.

`item_id` string

The ID of the item.

`output_index` integer

The index of the output item in the response.

`response_id` string

The ID of the response.

`text` string

The final text content.

`type` string

The event type, must be `response.text.done`.

OBJECT response.text.done

```
{
    "event_id": "event_4344",
    "type": "response.text.done",
    "response_id": "resp_001",
    "item_id": "msg_007",
    "output_index": 0,
    "content_index": 0,
    "text": "Sure, I can help with that."
}
```

response.audio_transcript.delta

Returned when the model-generated transcription of audio output is updated.

`content_index` integer
The index of the content part in the item's content array.

`delta` string
The transcript delta.

`event_id` string
The unique ID of the server event.

`item_id` string
The ID of the item.

`output_index` integer
The index of the output item in the response.

`response_id` string
The ID of the response.

`type` string
The event type, must be `response.audio_transcript.delta`.

OBJECT response.audio_transcript.delta

```json
{
    "event_id": "event_4546",
    "type": "response.audio_transcript.delt
    "response_id": "resp_001",
    "item_id": "msg_008",
    "output_index": 0,
    "content_index": 0,
    "delta": "Hello, how can I a"
}
```

response.audio_transcript.done

Returned when the model-generated transcription of audio output is done streaming. Also emitted when a Response is interrupted, incomplete, or cancelled.

`content_index` integer

The index of the content part in the item's content array.

`event_id` string

The unique ID of the server event.

`item_id` string

The ID of the item.

`output_index` integer

The index of the output item in the response.

`response_id` string

The ID of the response.

`transcript` string

The final transcript of the audio.

`type` string

The event type, must be `response.audio_transcript.done` .

OBJECT response.audio_transcript.done

```
{
    "event_id": "event_4748",
    "type": "response.audio_transcript.done
    "response_id": "resp_001",
    "item_id": "msg_008",
    "output_index": 0,
    "content_index": 0,
    "transcript": "Hello, how can I assist
}
```

response.audio.delta

Returned when the model-generated audio is updated.

`content_index` integer

The index of the content part in the item's content array.

`delta` string

Base64-encoded audio data delta.

`event_id` string

The unique ID of the server event.

`item_id` string

The ID of the item.

`output_index` integer

The index of the output item in the response.

`response_id` string

The ID of the response.

`type` string

The event type, must be `response.audio.delta`.

OBJECT response.audio.delta

```
{
    "event_id": "event_4950",
    "type": "response.audio.delta",
    "response_id": "resp_001",
    "item_id": "msg_008",
    "output_index": 0,
    "content_index": 0,
    "delta": "Base64EncodedAudioDelta"
}
```

response.audio.done

Returned when the model-generated audio is done. Also emitted when a Response is interrupted, incomplete, or cancelled.

`content_index` integer
The index of the content part in the item's content array.

`event_id` string
The unique ID of the server event.

`item_id` string
The ID of the item.

`output_index` integer
The index of the output item in the response.

`response_id` string
The ID of the response.

`type` string
The event type, must be `response.audio.done` .

OBJECT response.audio.done

```
{
    "event_id": "event_5152",
    "type": "response.audio.done",
    "response_id": "resp_001",
    "item_id": "msg_008",
    "output_index": 0,
    "content_index": 0
}
```

response.function_call_arguments.delta

Returned when the model-generated function call arguments are updated.

call_id string
The ID of the function call.

delta string
The arguments delta as a JSON string.

event_id string
The unique ID of the server event.

item_id string
The ID of the function call item.

output_index integer
The index of the output item in the response.

response_id string
The ID of the response.

type string
The event type, must be `response.function_call_arguments.delta`.

OBJECT response.function_call_arguments.del…

```
{
    "event_id": "event_5354",
    "type": "response.function_call_argumen
    "response_id": "resp_002",
    "item_id": "fc_001",
    "output_index": 0,
    "call_id": "call_001",
    "delta": "{\"location\": \"San\""
}
```

response.function_call_arguments.done

Returned when the model-generated function call arguments are done streaming. Also emitted when a Response is interrupted, incomplete, or cancelled.

`arguments` string

The final arguments as a JSON string.

`call_id` string

The ID of the function call.

`event_id` string

The unique ID of the server event.

`item_id` string

The ID of the function call item.

`output_index` integer

The index of the output item in the response.

`response_id` string

The ID of the response.

`type` string

The event type, must be `response.function_call_arguments.done`.

OBJECT response.function_call_arguments.done

```
{
    "event_id": "event_5556",
    "type": "response.function_call_argumen
    "response_id": "resp_002",
    "item_id": "fc_001",
    "output_index": 0,
    "call_id": "call_001",
    "arguments": "{\"location\": \"San Fran
}
```

transcription_session.updated

Returned when a transcription session is updated with a `transcription_session.update` event, unless there is an error.

`event_id` string

The unique ID of the server event.

`session` object

A new Realtime transcription session configuration.

When a session is created on the server via REST API, the session object also contains an ephemeral key. Default TTL for keys is 10 minutes. This property is not present when a session is updated via the WebSocket API.

Show properties

`type` string

The event type, must be `transcription_session.updated`.

OBJECT transcription_session.updated

```
{
  "event_id": "event_5678",
  "type": "transcription_session.updated",
  "session": {
    "id": "sess_001",
    "object": "realtime.transcription_sess
    "input_audio_format": "pcm16",
    "input_audio_transcription": {
      "model": "gpt-4o-transcribe",
      "prompt": "",
      "language": ""
    },
    "turn_detection": {
      "type": "server_vad",
      "threshold": 0.5,
      "prefix_padding_ms": 300,
      "silence_duration_ms": 500,
      "create_response": true,
      // "interrupt_response": false  -- t
    },
    "input_audio_noise_reduction": {
      "type": "near_field"
    },
    "include": [
      "item.input_audio_transcription.avg_
    ],
  }
}
```

rate_limits.updated

Emitted at the beginning of a Response to indicate the updated rate limits. When a Response is created some tokens will be "reserved" for the output tokens, the rate limits shown here reflect that reservation, which is then adjusted accordingly once the Response is completed.

`event_id` string

The unique ID of the server event.

`rate_limits` array

List of rate limit information.

Show properties

`type` string

The event type, must be `rate_limits.updated`.

OBJECT rate_limits.updated

```
{
    "event_id": "event_5758",
    "type": "rate_limits.updated",
    "rate_limits": [
        {
            "name": "requests",
            "limit": 1000,
            "remaining": 999,
            "reset_seconds": 60
        },
        {
            "name": "tokens",
            "limit": 50000,
            "remaining": 49950,
            "reset_seconds": 60
        }
    ]
}
```

output_audio_buffer.started

WebRTC Only: Emitted when the server begins streaming audio to the client. This event is emitted after an audio content part has been added (`response.content_part.added`) to the response. Learn more.

`event_id` string

The unique ID of the server event.

`response_id` string

The unique ID of the response that produced the audio.

`type` string

The event type, must be `output_audio_buffer.started` .

OBJECT output_audio_buffer.started

```json
{
    "event_id": "event_abc123",
    "type": "output_audio_buffer.started",
    "response_id": "resp_abc123"
}
```

output_audio_buffer.stopped

WebRTC Only: Emitted when the output audio buffer has been completely drained on the server, and no more audio is forthcoming. This event is emitted after the full response data has been sent to the client (`response.done`). Learn more.

OBJECT output_audio_buffer.stopped

```
{
    "event_id": "event_abc123",
    "type": "output_audio_buffer.stopped",
    "response_id": "resp_abc123"
}
```

`event_id` string
The unique ID of the server event.

`response_id` string
The unique ID of the response that produced the audio.

`type` string
The event type, must be `output_audio_buffer.stopped` .

output_audio_buffer.cleared

WebRTC Only: Emitted when the output audio buffer is cleared. This happens either in VAD mode when the user has interrupted (`input_audio_buffer.speech_started`), or when the client has emitted the `output_audio_buffer.clear` event to manually cut off the current audio response. Learn more.

`event_id` string
The unique ID of the server event.

`response_id` string
The unique ID of the response that produced the audio.

`type` string
The event type, must be `output_audio_buffer.cleared`.

OBJECT output_audio_buffer.cleared

```
{
    "event_id": "event_abc123",
    "type": "output_audio_buffer.cleared",
    "response_id": "resp_abc123"
}
```

Chat Completions

The Chat Completions API endpoint will generate a model response from a list of messages comprising a conversation.

Related guides:

- Quickstart
- Text inputs and outputs
- Image inputs
- Audio inputs and outputs
- Structured Outputs
- Function calling
- Conversation state

Starting a new project? We recommend trying Responses to take advantage of the latest OpenAI platform features. Compare Chat Completions with Responses.

Create chat completion

```
POST https://api.openai.com/v1/chat/completions
```

Starting a new project? We recommend trying Responses to take advantage of the latest OpenAI platform features. Compare Chat Completions with Responses.

Creates a model response for the given chat conversation. Learn more in the text generation, vision, and audio guides.

Default | Image input | Streaming | Functions | Logp

Example request gpt-5 curl

```
curl https://api.openai.com/v1/chat/comple
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $OPENAI_API_KE
  -d '{
    "model": "gpt-5",
    "messages": [
```

Parameter support can differ depending on the model used to generate the response, particularly for newer reasoning models. Parameters that are only supported for reasoning models are noted below. For the current state of unsupported parameters in reasoning models, refer to the reasoning guide.

```
      {
        "role": "developer",
        "content": "You are a helpful assi
      },
      {
        "role": "user",
        "content": "Hello!"
      }
```

Request body

`messages` array Required

A list of messages comprising the conversation so far. Depending on the model you use, different message types (modalities) are supported, like text, images, and audio.

Show possible types

`model` string Required

Model ID used to generate the response, like `gpt-4o` or `o3`. OpenAI offers a wide range of models with different capabilities, performance characteristics, and price points. Refer to the model guide to browse and compare available models.

`audio` object or null Optional

Parameters for audio output. Required when audio output is requested with `modalities: ["audio"]`. Learn more.

Show properties

Response

```
{
  "id": "chatcmpl-B9MBs8CjcvOU2jLn4n570S5q
  "object": "chat.completion",
  "created": 1741569952,
  "model": "gpt-4.1-2025-04-14",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Hello! How can I assis
        "refusal": null,
        "annotations": []
      },
```

frequency_penalty number or null Optional Defaults to 0

Number between -2.0 and 2.0. Positive values penalize new tokens based on their existing frequency in the text so far, decreasing the model's likelihood to repeat the same line verbatim.

---

function_call Deprecated string or object Optional

Deprecated in favor of `tool_choice`.

Controls which (if any) function is called by the model.

`none` means the model will not call a function and instead generates a message.

`auto` means the model can pick between generating a message or calling a function.

Specifying a particular function via `{"name": "my_function"}` forces the model to call that function.

`none` is the default when no functions are present. `auto` is the default if functions are present.

Show possible types

---

functions Deprecated array Optional

Deprecated in favor of `tools`.

A list of functions the model may generate JSON inputs for.

Show properties

---

logit_bias map Optional Defaults to null

Modify the likelihood of specified tokens appearing in the completion.

Accepts a JSON object that maps tokens (specified by their token ID in the tokenizer) to an associated bias value from -100 to 100. Mathematically, the bias is added to the logits generated by the model prior to sampling. The exact effect will vary per model, but values between -1 and 1 should decrease or increase likelihood of selection; values like -100 or 100 should result in a ban or exclusive selection of the relevant token.

`logprobs` boolean or null Optional Defaults to false

Whether to return log probabilities of the output tokens or not. If true, returns the log probabilities of each output token returned in the `content` of `message` .

`max_completion_tokens` integer or null Optional

An upper bound for the number of tokens that can be generated for a completion, including visible output tokens and reasoning tokens.

`max_tokens` Deprecated integer or null Optional

The maximum number of tokens that can be generated in the chat completion. This value can be used to control costs for text generated via API.

This value is now deprecated in favor of `max_completion_tokens` , and is not compatible with o-series models.

`metadata` map Optional

Set of 16 key-value pairs that can be attached to an object. This can be useful for storing additional information about the object in a structured format, and querying for objects via API or the dashboard.

Keys are strings with a maximum length of 64 characters. Values are strings with a maximum length of 512 characters.

`modalities` array or null Optional

Output types that you would like the model to generate. Most models are capable of generating text, which is the default:

`["text"]`

The `gpt-4o-audio-preview` model can also be used to generate audio. To request that this model generate both text and audio responses, you can use:

`["text", "audio"]`

`n` integer or null Optional Defaults to 1

How many chat completion choices to generate for each input message. Note that you will be charged based on the number of generated tokens across all of the choices. Keep `n` as `1` to minimize costs.

`parallel_tool_calls` boolean Optional Defaults to true

Whether to enable parallel function calling during tool use.

`prediction` object Optional

Configuration for a Predicted Output, which can greatly improve response times when large parts of the model response are known ahead of time. This is most common when you are regenerating a file with only minor changes to most of the content.

Show possible types

`presence_penalty` number or null Optional Defaults to 0

Number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far, increasing the model's likelihood to talk about new topics.

`prompt_cache_key` string Optional

Used by OpenAI to cache responses for similar requests to optimize your cache hit rates. Replaces the `user` field. Learn more.

`reasoning_effort` string or null Optional Defaults to medium

Constrains effort on reasoning for reasoning models. Currently supported values are `minimal`, `low`, `medium`, and `high`. Reducing reasoning effort can result in faster responses and fewer tokens used on reasoning in a response.

`response_format` object Optional

An object specifying the format that the model must output.

Setting to `{ "type": "json_schema", "json_schema": {...} }` enables Structured Outputs which ensures the model will match your supplied JSON schema. Learn more in the Structured Outputs guide.

Setting to `{ "type": "json_object" }` enables the older JSON mode, which ensures the message the model generates is valid JSON.

Using `json_schema` is preferred for models that support it.

Show possible types

`safety_identifier` string Optional

A stable identifier used to help detect users of your application that may be violating OpenAI's usage policies. The IDs should be a string that uniquely identifies each user. We recommend hashing their username or email address, in order to avoid sending us any identifying information. Learn more.

`seed` integer or null Optional

This feature is in Beta. If specified, our system will make a best effort to sample deterministically, such that repeated requests with the same `seed` and parameters should return the same result. Determinism is not guaranteed, and you should refer to the `system_fingerprint` response parameter to monitor changes in the backend.

`service_tier` string or null Optional Defaults to auto

Specifies the processing type used for serving the request.

> If set to 'auto', then the request will be processed with the service tier configured in the Project settings. Unless otherwise configured, the Project will use 'default'.
>
> If set to 'default', then the request will be processed with the standard pricing and performance for the selected model.
>
> If set to 'flex' or 'priority', then the request will be processed with

the corresponding service tier. Contact sales to learn more about Priority processing.

When not set, the default behavior is 'auto'.

When the `service_tier` parameter is set, the response body will include the `service_tier` value based on the processing mode actually used to serve the request. This response value may be different from the value set in the parameter.

`stop` string / array / null Optional Defaults to null
Not supported with latest reasoning models `o3` and `o4-mini`.

Up to 4 sequences where the API will stop generating further tokens. The returned text will not contain the stop sequence.

`store` boolean or null Optional Defaults to false
Whether or not to store the output of this chat completion request for use in our model distillation or evals products.

Supports text and image inputs. Note: image inputs over 10MB will be dropped.

`stream` boolean or null Optional Defaults to false
If set to true, the model response data will be streamed to the client as it is generated using server-sent events. See the Streaming section below for more information, along with the streaming responses guide for more information on how to handle the streaming events.

`stream_options` object or null Optional Defaults to null

Options for streaming response. Only set this when you set `stream: true`.

Show properties

`temperature` number or null Optional Defaults to 1

What sampling temperature to use, between 0 and 2. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic. We generally recommend altering this or `top_p` but not both.

`tool_choice` string or object Optional

Controls which (if any) tool is called by the model. `none` means the model will not call any tool and instead generates a message. `auto` means the model can pick between generating a message or calling one or more tools. `required` means the model must call one or more tools. Specifying a particular tool via `{"type": "function", "function": {"name": "my_function"}}` forces the model to call that tool.

`none` is the default when no tools are present. `auto` is the default if tools are present.

Show possible types

`tools` array Optional

A list of tools the model may call. You can provide either custom tools or function tools.

Show possible types

`top_logprobs` integer or null Optional

An integer between 0 and 20 specifying the number of most likely tokens to return at each token position, each with an associated log probability. `logprobs` must be set to `true` if this parameter is used.

`top_p` number or null Optional Defaults to 1

An alternative to sampling with temperature, called nucleus sampling, where the model considers the results of the tokens with top_p probability mass. So 0.1 means only the tokens comprising the top 10% probability mass are considered.

We generally recommend altering this or `temperature` but not both.

`user` Deprecated string Optional

This field is being replaced by `safety_identifier` and `prompt_cache_key`. Use `prompt_cache_key` instead to maintain caching optimizations. A stable identifier for your end-users. Used to boost cache hit rates by better bucketing similar requests and to help OpenAI detect and prevent abuse. Learn more.

`verbosity` string or null Optional Defaults to medium

Constrains the verbosity of the model's response. Lower values will result in more concise responses, while higher values will result in more verbose responses. Currently supported values are `low`, `medium`, and `high`.

`web_search_options` object Optional

This tool searches the web for relevant results to use in a response. Learn more about the web search tool.

Show properties

Returns

Returns a chat completion object, or a streamed sequence of chat completion chunk objects if the request is streamed.

Get chat completion

```
GET https://api.openai.com/v1/chat/completions/{comp
    letion_id}
```

Get a stored chat completion. Only Chat Completions that have been created with the `store` parameter set to `true` will be returned.

Example request curl

```
curl https://api.openai.com/v1/chat/complet
  -H "Authorization: Bearer $OPENAI_API_KEY
  -H "Content-Type: application/json"
```

Response