**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.

**Request body**

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.

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.

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.

input\_audio\_format

string

Optional

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.

output\_audio\_format

string

Optional

The format of output audio. Options are pcm16, g711\_ulaw, or g711\_alaw. For pcm16, output audio is sampled at a rate of 24kHz.

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 [OpenAI Whisper transcription](https://platform.openai.com/docs/api-reference/audio/createTranscription) and should be treated as rough guidance rather than the representation understood by the model. The client can optionally set the language and prompt for transcription, these fields will be passed to the Whisper API.

Show properties

turn\_detection

object

Optional

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

tools

array

Optional

Tools (functions) available to the model.

Show properties

tool\_choice

string

Optional

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

temperature

number

Optional

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

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.

**Returns**

The created Realtime session object, plus an ephemeral key

Example request

curl

curl -X POST https://api.openai.com/v1/realtime/sessions \

-H "Authorization: Bearer $OPENAI\_API\_KEY" \

-H "Content-Type: application/json" \

-d '{

"model": "gpt-4o-realtime-preview-2024-12-17",

"modalities": ["audio", "text"],

"instructions": "You are a friendly assistant."

}'

Response

{

"id": "sess\_001",

"object": "realtime.session",

"model": "gpt-4o-realtime-preview-2024-12-17",

"modalities": ["audio", "text"],

"instructions": "You are a friendly assistant.",

"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,

"client\_secret": {

"value": "ek\_abc123",

"expires\_at": 1234567890

}

}

**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

modalities

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

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.

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.

input\_audio\_format

string

The format of input audio. Options are pcm16, g711\_ulaw, or g711\_alaw.

output\_audio\_format

string

The format of output 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. Transcription runs asynchronously through Whisper and should be treated as rough guidance rather than the representation understood by the model.

Show properties

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

tools

array

Tools (functions) available to the model.

Show properties

tool\_choice

string

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

temperature

number

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

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.

OBJECT The session object

{

"id": "sess\_001",

"object": "realtime.session",

"model": "gpt-4o-realtime-preview-2024-12-17",

"modalities": ["audio", "text"],

"instructions": "You are a friendly assistant.",

"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,

"client\_secret": {

"value": "ek\_abc123",

"expires\_at": 1234567890

}

}

**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.

event\_id

string

Optional client-generated ID used to identify this event.

type

string

The event type, must be session.update.

session

object

Realtime session object configuration.

Show properties

OBJECT session.update

{

"event\_id": "event\_123",

"type": "session.update",

"session": {

"modalities": ["text", "audio"],

"instructions": "You are a helpful assistant.",

"voice": "sage",

"input\_audio\_format": "pcm16",

"output\_audio\_format": "pcm16",

"input\_audio\_transcription": {

"model": "whisper-1"

},

"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 current weather...",

"parameters": {

"type": "object",

"properties": {

"location": { "type": "string" }

},

"required": ["location"]

}

}

],

"tool\_choice": "auto",

"temperature": 0.8,

"max\_response\_output\_tokens": "inf"

}

}

**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.

event\_id

string

Optional client-generated ID used to identify this event.

type

string

The event type, must be input\_audio\_buffer.append.

audio

string

Base64-encoded audio bytes. This must be in the format specified by the input\_audio\_format field in the session configuration.

OBJECT input\_audio\_buffer.append

{

"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

{

"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 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.

type

string

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

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.

item

object

The item to add to the conversation.

Show properties

OBJECT conversation.item.create

{

"event\_id": "event\_345",

"type": "conversation.item.create",

"previous\_item\_id": null,

"item": {

"id": "msg\_001",

"type": "message",

"role": "user",

"content": [

{

"type": "input\_text",

"text": "Hello, how are you?"

}

]

}

}

**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.

event\_id

string

Optional client-generated ID used to identify this event.

type

string

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

item\_id

string

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

content\_index

integer

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

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.

OBJECT conversation.item.truncate

{

"event\_id": "event\_678",

"type": "conversation.item.truncate",

"item\_id": "msg\_002",

"content\_index": 0,

"audio\_end\_ms": 1500

}

**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.

event\_id

string

Optional client-generated ID used to identify this event.

type

string

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

item\_id

string

The ID of the item to delete.

OBJECT conversation.item.delete

{

"event\_id": "event\_901",

"type": "conversation.item.delete",

"item\_id": "msg\_003"

}

**response.create**

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

type

string

The event type, must be response.create.

response

object

Create a new Realtime response with these parameters

Show properties

OBJECT response.create

{

"event\_id": "event\_234",

"type": "response.create",

"response": {

"modalities": ["text", "audio"],

"instructions": "Please assist the user.",

"voice": "sage",

"output\_audio\_format": "pcm16",

"tools": [

{

"type": "function",

"name": "calculate\_sum",

"description": "Calculates the sum of two numbers.",

"parameters": {

"type": "object",

"properties": {

"a": { "type": "number" },

"b": { "type": "number" }

},

"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.

type

string

The event type, must be response.cancel.

response\_id

string

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

OBJECT response.cancel

{

"event\_id": "event\_567",

"type": "response.cancel"

}

**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.

event\_id

string

The unique ID of the server event.

type

string

The event type, must be error.

error

object

Details of the error.

Show properties

OBJECT error

{

"event\_id": "event\_890",

"type": "error",

"error": {

"type": "invalid\_request\_error",

"code": "invalid\_event",

"message": "The 'type' field is missing.",

"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.

type

string

The event type, must be session.created.

session

object

Realtime session object configuration.

Show properties

OBJECT session.created

{

"event\_id": "event\_1234",

"type": "session.created",

"session": {

"id": "sess\_001",

"object": "realtime.session",

"model": "gpt-4o-realtime-preview-2024-12-17",

"modalities": ["text", "audio"],

"instructions": "...model instructions here...",

"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"

}

}

**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.

type

string

The event type, must be session.updated.

session

object

Realtime session object configuration.

Show properties

OBJECT session.updated

{

"event\_id": "event\_5678",

"type": "session.updated",

"session": {

"id": "sess\_001",

"object": "realtime.session",

"model": "gpt-4o-realtime-preview-2024-12-17",

"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

}

}

**conversation.created**

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

event\_id

string

The unique ID of the server event.

type

string

The event type, must be conversation.created.

conversation

object

The conversation resource.

Show properties

OBJECT conversation.created

{

"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.

event\_id

string

The unique ID of the server event.

type

string

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

previous\_item\_id

string

The ID of the preceding item in the Conversation context, allows the client to understand the order of the conversation.

item

object

The item to add to the conversation.

Show properties

OBJECT conversation.item.created

{

"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 are you",

"audio": "base64encodedaudio=="

}

]

}

}

**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, currently always whisper-1. Thus the transcript may diverge somewhat from the model's interpretation, and should be treated as a rough guide.

event\_id

string

The unique ID of the server event.

type

string

The event type, must be conversation.item.input\_audio\_transcription.completed.

item\_id

string

The ID of the user message item containing the audio.

content\_index

integer

The index of the content part containing the audio.

transcript

string

The transcribed text.

OBJECT conversation.item.input\_audio\_transcription.completed

{

"event\_id": "event\_2122",

"type": "conversation.item.input\_audio\_transcription.completed",

"item\_id": "msg\_003",

"content\_index": 0,

"transcript": "Hello, how are you?"

}

**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.

event\_id

string

The unique ID of the server event.

type

string

The event type, must be conversation.item.input\_audio\_transcription.failed.

item\_id

string

The ID of the user message item.

content\_index

integer

The index of the content part containing the audio.

error

object

Details of the transcription error.

Show properties

OBJECT conversation.item.input\_audio\_transcription.failed

{

"event\_id": "event\_2324",

"type": "conversation.item.input\_audio\_transcription.failed",

"item\_id": "msg\_003",

"content\_index": 0,

"error": {

"type": "transcription\_error",

"code": "audio\_unintelligible",

"message": "The audio could not be transcribed.",

"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.

event\_id

string

The unique ID of the server event.

type

string

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

item\_id

string

The ID of the assistant message item that was truncated.

content\_index

integer

The index of the content part that was truncated.

audio\_end\_ms

integer

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

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.

type

string

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

item\_id

string

The ID of the item that was 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.

type

string

The event type, must be input\_audio\_buffer.committed.

previous\_item\_id

string

The ID of the preceding item after which the new item will be inserted.

item\_id

string

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

OBJECT input\_audio\_buffer.committed

{

"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 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).

event\_id

string

The unique ID of the server event.

type

string

The event type, must be input\_audio\_buffer.speech\_started.

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.

item\_id

string

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

OBJECT input\_audio\_buffer.speech\_started

{

"event\_id": "event\_1516",

"type": "input\_audio\_buffer.speech\_started",

"audio\_start\_ms": 1000,

"item\_id": "msg\_003"

}

**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.

event\_id

string

The unique ID of the server event.

type

string

The event type, must be input\_audio\_buffer.speech\_stopped.

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.

item\_id

string

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

OBJECT input\_audio\_buffer.speech\_stopped

{

"event\_id": "event\_1718",

"type": "input\_audio\_buffer.speech\_stopped",

"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.

type

string

The event type, must be response.created.

response

object

The response resource.

Show properties

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

event\_id

string

The unique ID of the server event.

type

string

The event type, must be response.done.

response

object

The response resource.

Show properties

OBJECT response.done

{

"event\_id": "event\_3132",

"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 can I assist you today?"

}

]

}

],

"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,

"audio\_tokens": 256

}

},

"output\_token\_details": {

"text\_tokens":36,

"audio\_tokens":112

}

}

}

}

**response.output\_item.added**

Returned when a new Item is created during Response generation.

event\_id

string

The unique ID of the server event.

type

string

The event type, must be response.output\_item.added.

response\_id

string

The ID of the Response to which the item belongs.

output\_index

integer

The index of the output item in the Response.

item

object

The item to add to the conversation.

Show properties

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.

type

string

The event type, must be response.output\_item.done.

response\_id

string

The ID of the Response to which the item belongs.

output\_index

integer

The index of the output item in the Response.

item

object

The item to add to the conversation.

Show properties

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 with that."

}

]

}

}

**response.content\_part.added**

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

event\_id

string

The unique ID of the server event.

type

string

The event type, must be response.content\_part.added.

response\_id

string

The ID of the response.

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.

content\_index

integer

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

part

object

The content part that was added.

Show properties

OBJECT response.content\_part.added

{

"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.

event\_id

string

The unique ID of the server event.

type

string

The event type, must be response.content\_part.done.

response\_id

string

The ID of the response.

item\_id

string

The ID of the item.

output\_index

integer

The index of the output item in the response.

content\_index

integer

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

part

object

The content part that is done.

Show properties

OBJECT response.content\_part.done

{

"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 that."

}

}

**response.text.delta**

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

event\_id

string

The unique ID of the server event.

type

string

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

response\_id

string

The ID of the response.

item\_id

string

The ID of the item.

output\_index

integer

The index of the output item in the response.

content\_index

integer

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

delta

string

The text delta.

OBJECT response.text.delta

{

"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.

event\_id

string

The unique ID of the server event.

type

string

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

response\_id

string

The ID of the response.

item\_id

string

The ID of the item.

output\_index

integer

The index of the output item in the response.

content\_index

integer

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

text

string

The final text content.

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.

event\_id

string

The unique ID of the server event.

type

string

The event type, must be response.audio\_transcript.delta.

response\_id

string

The ID of the response.

item\_id

string

The ID of the item.

output\_index

integer

The index of the output item in the response.

content\_index

integer

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

delta

string

The transcript delta.

OBJECT response.audio\_transcript.delta

{

"event\_id": "event\_4546",

"type": "response.audio\_transcript.delta",

"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.

event\_id

string

The unique ID of the server event.

type

string

The event type, must be response.audio\_transcript.done.

response\_id

string

The ID of the response.

item\_id

string

The ID of the item.

output\_index

integer

The index of the output item in the response.

content\_index

integer

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

transcript

string

The final transcript of the audio.

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 you today?"

}

**response.audio.delta**

Returned when the model-generated audio is updated.

event\_id

string

The unique ID of the server event.

type

string

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

response\_id

string

The ID of the response.

item\_id

string

The ID of the item.

output\_index

integer

The index of the output item in the response.

content\_index

integer

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

delta

string

Base64-encoded audio data 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.

event\_id

string

The unique ID of the server event.

type

string

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

response\_id

string

The ID of the response.

item\_id

string

The ID of the item.

output\_index

integer

The index of the output item in the response.

content\_index

integer

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

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.

event\_id

string

The unique ID of the server event.

type

string

The event type, must be response.function\_call\_arguments.delta.

response\_id

string

The ID of the response.

item\_id

string

The ID of the function call item.

output\_index

integer

The index of the output item in the response.

call\_id

string

The ID of the function call.

delta

string

The arguments delta as a JSON string.

OBJECT response.function\_call\_arguments.delta

{

"event\_id": "event\_5354",

"type": "response.function\_call\_arguments.delta",

"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.

event\_id

string

The unique ID of the server event.

type

string

The event type, must be response.function\_call\_arguments.done.

response\_id

string

The ID of the response.

item\_id

string

The ID of the function call item.

output\_index

integer

The index of the output item in the response.

call\_id

string

The ID of the function call.

arguments

string

The final arguments as a JSON string.

OBJECT response.function\_call\_arguments.done

{

"event\_id": "event\_5556",

"type": "response.function\_call\_arguments.done",

"response\_id": "resp\_002",

"item\_id": "fc\_001",

"output\_index": 0,

"call\_id": "call\_001",

"arguments": "{\"location\": \"San Francisco\"}"

}

**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.

type

string

The event type, must be rate\_limits.updated.

rate\_limits

array

List of rate limit information.

Show properties

cancelAssistantSpeech

THere is a bunch of different events transmitted between the openai realtime api and the application, which you could find in this file.   
For each output from the agent, you will always find a pair of "output audio buffer.started" and "output audio buffer.stopped" at the beginning and end of this audio output. it's a buffer that holds the audio data as it is being generated.   
  
I intend to add something triggered by "output audio buffer.stopped" only when the application is running the agent [speaking2and3].  
Here is what I need your help with:  
When the event "output audio buffer.stopped", you should get the transcript of that output from the agent and check whether it ends with "Your preparation time starts now", and if it does, the app should create a countdown bar for one minute somewhere on the page and start it (you can decide where to put this bar as long as you believe it is easy to implement).   
  
Then when the countdown reaches the end(which means one minute passes), the app should