
object string

amount object

The monetary value in its associated currency.

▼ Show properties

line_item string or null

When `group_by=line_item`, this field provides the line item of the grouped costs result.

project_id string or null

When `group_by=project_id`, this field provides the project ID of the grouped costs result.

OBJECT Costs object



```
1 {
2   "object": "organization.costs.result",
3   "amount": {
4     "value": 0.06,
5     "currency": "usd"
6   },
7   "line_item": "Image models",
8   "project_id": "proj_abc"
9 }
```

Realtime Beta

Communicate with a GPT-4o class model in real time using WebRTC or WebSockets. Supports text and audio inputs and outputs, 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.

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](#) 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.

⌶ Hide properties

type string Optional
The type of the tool, i.e. `function` .

name string Optional
The name of the function.

description string Optional
The description of the function, including guidance on when and how to call it, and guidance about what to tell the user when calling (if anything).

parameters object Optional
Parameters of the function in JSON Schema.

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 ⌵ 

```
1 curl -X POST https://api.openai.com/v1/realtime/sessions \
2   -H "Authorization: Bearer $OPENAI_API_KEY" \
3   -H "Content-Type: application/json" \
4   -d '{
5     "model": "gpt-4o-realtime-preview-2024-12-17",
```

```
6     "modalities": ["audio", "text"],
7     "instructions": "You are a friendly assistant."
8   }'
```

Response



```
1  {
2    "id": "sess_001",
3    "object": "realtime.session",
4    "model": "gpt-4o-realtime-preview-2024-12-17",
5    "modalities": ["audio", "text"],
6    "instructions": "You are a friendly assistant.",
7    "voice": "alloy",
8    "input_audio_format": "pcm16",
9    "output_audio_format": "pcm16",
10   "input_audio_transcription": {
11     "model": "whisper-1"
12   },
13   "turn_detection": null,
14   "tools": [],
15   "tool_choice": "none",
16   "temperature": 0.7,
17   "max_response_output_tokens": 200,
18   "client_secret": {
19     "value": "ek_abc123",
20     "expires_at": 1234567890
21   }
22 }
```

The session object

A new Realtime session configuration, with an ephemeral 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`.

```
1  {
2    "id": "sess_001",
3    "object": "realtime.session",
4    "model": "gpt-4o-realtime-preview-2024-12-17",
5    "modalities": ["audio", "text"],
6    "instructions": "You are a friendly assistant.",
7    "voice": "alloy",
8    "input_audio_format": "pcm16",
9    "output_audio_format": "pcm16",
10   "input_audio_transcription": {
11     "model": "whisper-1"
12   },
13   "turn_detection": null,
14   "tools": [],
15   "tool_choice": "none",
16   "temperature": 0.7,
17   "max_response_output_tokens": 200,
18   "client_secret": {
19     "value": "ek_abc123",
20     "expires_at": 1234567890
21   }
22 }
```

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.

^ Hide properties

modalities

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

model string

The Realtime model used for this session.

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` . 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

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

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](#) 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.

v 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 session.update



```
1  {
2    "event_id": "event_123",
3    "type": "session.update",
4    "session": {
5      "modalities": ["text", "audio"],
6      "instructions": "You are a helpful assistant.",
7      "voice": "sage",
8      "input_audio_format": "pcm16",
9      "output_audio_format": "pcm16",
10     "input_audio_transcription": {
11       "model": "whisper-1"
12     },
13     "turn_detection": {
14       "type": "server_vad",
15       "threshold": 0.5,
16       "prefix_padding_ms": 300,
17       "silence_duration_ms": 500,
18       "create_response": true
19     },
20     "tools": [
21       {
22         "type": "function",
23         "name": "get_weather",
24         "description": "Get the current weather...",
25         "parameters": {
26           "type": "object",
27           "properties": {
```



```

28         "location": { "type": "string" }
29     },
30     "required": ["location"]
31 }
32 }
33 ],
34 "tool_choice": "auto",
35 "temperature": 0.8,
36 "max_response_output_tokens": "inf"
37 }
38 }

```

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`



```

1 {
2     "event_id": "event_456",
3     "type": "input_audio_buffer.append",
4     "audio": "Base64EncodedAudioData"
5 }

```

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`



```
1 {  
2   "event_id": "event_789",  
3   "type": "input_audio_buffer.commit"  
4 }
```

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`



```
1 {  
2   "event_id": "event_012",  
3   "type": "input_audio_buffer.clear"  
4 }
```

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`



```
1  {
2    "event_id": "event_345",
3    "type": "conversation.item.create",
4    "previous_item_id": null,
5    "item": {
6      "id": "msg_001",
7      "type": "message",
8      "role": "user",
9      "content": [
10         {
11           "type": "input_text",
12           "text": "Hello, how are you?"
13         }
14       ]
15     }
16  }
```

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`



```
1 {  
2   "event_id": "event_678",  
3   "type": "conversation.item.truncate",  
4   "item_id": "msg_002",  
5   "content_index": 0,  
6   "audio_end_ms": 1500  
7 }
```

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`



```
1 {  
2   "event_id": "event_901",  
3   "type": "conversation.item.delete",  
4   "item_id": "msg_003"  
5 }
```

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



```
1  {
2    "event_id": "event_234",
3    "type": "response.create",
4    "response": {
5      "modalities": ["text", "audio"],
6      "instructions": "Please assist the user.",
7      "voice": "sage",
8      "output_audio_format": "pcm16",
9      "tools": [
10       {
11         "type": "function",
12         "name": "calculate_sum",
13         "description": "Calculates the sum of two numbers.",
14         "parameters": {
15           "type": "object",
16           "properties": {
17             "a": { "type": "number" },
18             "b": { "type": "number" }
19           },
20           "required": ["a", "b"]
21         }
22       }
23     ],
24     "tool_choice": "auto",
25     "temperature": 0.8,
26     "max_output_tokens": 1024
27   }
28 }
```

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`



```
1 {
2   "event_id": "event_567",
3   "type": "response.cancel"
4 }
```

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`



```
1 {
2   "event_id": "event_890",
3   "type": "error",
4   "error": {
5     "type": "invalid_request_error",
6     "code": "invalid_event",
7     "message": "The 'type' field is missing.",
8     "param": null,
9     "event_id": "event_567"
```

```
10 }
11 }
```

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



```
1 {
2   "event_id": "event_1234",
3   "type": "session.created",
4   "session": {
5     "id": "sess_001",
6     "object": "realtime.session",
7     "model": "gpt-4o-realtime-preview-2024-12-17",
8     "modalities": ["text", "audio"],
9     "instructions": "...model instructions here...",
10    "voice": "sage",
11    "input_audio_format": "pcm16",
12    "output_audio_format": "pcm16",
13    "input_audio_transcription": null,
14    "turn_detection": {
15      "type": "server_vad",
16      "threshold": 0.5,
17      "prefix_padding_ms": 300,
18      "silence_duration_ms": 200
19    },
20    "tools": [],
21    "tool_choice": "auto",
22    "temperature": 0.8,
23    "max_response_output_tokens": "inf"
24  }
25 }
```


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`



```
1 {
2   "event_id": "event_5678",
3   "type": "session.updated",
4   "session": {
5     "id": "sess_001",
6     "object": "realtime.session",
7     "model": "gpt-4o-realtime-preview-2024-12-17",
8     "modalities": ["text"],
9     "instructions": "New instructions",
10    "voice": "sage",
11    "input_audio_format": "pcm16",
12    "output_audio_format": "pcm16",
13    "input_audio_transcription": {
14      "model": "whisper-1"
15    },
16    "turn_detection": null,
17    "tools": [],
18    "tool_choice": "none",
19    "temperature": 0.7,
20    "max_response_output_tokens": 200
21  }
22 }
```

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`



```
1 {  
2   "event_id": "event_9101",  
3   "type": "conversation.created",  
4   "conversation": {  
5     "id": "conv_001",  
6     "object": "realtime.conversation"  
7   }  
8 }
```

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



```
1 {
2   "event_id": "event_1920",
3   "type": "conversation.item.created",
4   "previous_item_id": "msg_002",
5   "item": {
6     "id": "msg_003",
7     "object": "realtime.item",
8     "type": "message",
9     "status": "completed",
10    "role": "user",
11    "content": [
12      {
13        "type": "input_audio",
14        "transcript": "hello how are you",
15        "audio": "base64encodedaudio=="
16      }
17    ]
18  }
19 }
```

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`



```
1 {
2   "event_id": "event_2122",
3   "type": "conversation.item.input_audio_transcription.completed",
4   "item_id": "msg_003",
5   "content_index": 0,
6   "transcript": "Hello, how are you?"
7 }
```

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



```
1  {
2    "event_id": "event_2324",
3    "type": "conversation.item.input_audio_transcription.failed",
4    "item_id": "msg_003",
5    "content_index": 0,
6    "error": {
7      "type": "transcription_error",
8      "code": "audio_unintelligible",
9      "message": "The audio could not be transcribed.",
10     "param": null
11   }
12 }
```

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



```
1 {
2   "event_id": "event_2526",
3   "type": "conversation.item.truncated",
4   "item_id": "msg_004",
5   "content_index": 0,
6   "audio_end_ms": 1500
7 }
```

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`



```
1 {
2   "event_id": "event_2728",
3   "type": "conversation.item.deleted",
4   "item_id": "msg_005"
5 }
```

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`



```
1 {  
2   "event_id": "event_1121",  
3   "type": "input_audio_buffer.committed",  
4   "previous_item_id": "msg_001",  
5   "item_id": "msg_002"  
6 }
```

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`



```
1 {  
2   "event_id": "event_1314",  
3   "type": "input_audio_buffer.cleared"  
4 }
```

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`



```
1 {  
2   "event_id": "event_1516",  
3   "type": "input_audio_buffer.speech_started",  
4   "audio_start_ms": 1000,  
5   "item_id": "msg_003"  
6 }
```

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`



```
1 {
2   "event_id": "event_1718",
3   "type": "input_audio_buffer.speech_stopped",
4   "audio_end_ms": 2000,
5   "item_id": "msg_003"
6 }
```

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`



```
1 {
2   "event_id": "event_2930",
3   "type": "response.created",
4   "response": {
5     "id": "resp_001",
6     "object": "realtime.response",
7     "status": "in_progress",
8     "status_details": null,
9     "output": [],
10    "usage": null
11  }
12 }
```

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`



```
1  {
2    "event_id": "event_3132",
3    "type": "response.done",
4    "response": {
5      "id": "resp_001",
6      "object": "realtime.response",
7      "status": "completed",
8      "status_details": null,
9      "output": [
10       {
11         "id": "msg_006",
12         "object": "realtime.item",
13         "type": "message",
14         "status": "completed",
15         "role": "assistant",
16         "content": [
17           {
18             "type": "text",
19             "text": "Sure, how can I assist you today?"
20           }
21         ]
22       }
23     ],
24     "usage": {
25       "total_tokens": 275,
26       "input_tokens": 127,
27       "output_tokens": 148,
28       "input_token_details": {
```

```

29         "cached_tokens":384,
30         "text_tokens":119,
31         "audio_tokens":8,
32         "cached_tokens_details": {
33             "text_tokens": 128,
34             "audio_tokens": 256
35         }
36     },
37     "output_token_details": {
38         "text_tokens":36,
39         "audio_tokens":112
40     }
41 }
42 }
43 }

```

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



```

1  {
2      "event_id": "event_3334",
3      "type": "response.output_item.added",
4      "response_id": "resp_001",
5      "output_index": 0,
6      "item": {
7          "id": "msg_007",
8          "object": "realtime.item",

```

```
9      "type": "message",
10     "status": "in_progress",
11     "role": "assistant",
12     "content": []
13   }
14 }
```

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



```
1  {
2    "event_id": "event_3536",
3    "type": "response.output_item.done",
4    "response_id": "resp_001",
5    "output_index": 0,
6    "item": {
7      "id": "msg_007",
8      "object": "realtime.item",
9      "type": "message",
10     "status": "completed",
11     "role": "assistant",
12     "content": [
13       {
14         "type": "text",
15         "text": "Sure, I can help with that."
```

```
16         }
17     ]
18 }
19 }
```

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`



```
1  {
2      "event_id": "event_3738",
3      "type": "response.content_part.added",
4      "response_id": "resp_001",
5      "item_id": "msg_007",
6      "output_index": 0,
7      "content_index": 0,
8      "part": {
9          "type": "text",
10         "text": ""
```

```
11     }
12 }
```

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`



```
1  {
2    "event_id": "event_3940",
3    "type": "response.content_part.done",
4    "response_id": "resp_001",
5    "item_id": "msg_007",
6    "output_index": 0,
7    "content_index": 0,
8    "part": {
9      "type": "text",
10     "text": "Sure, I can help with that."
11   }
12 }
```

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



```
1 {
2   "event_id": "event_4142",
3   "type": "response.text.delta",
4   "response_id": "resp_001",
5   "item_id": "msg_007",
6   "output_index": 0,
7   "content_index": 0,
8   "delta": "Sure, I can h"
9 }
```

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`



```
1 {  
2   "event_id": "event_4344",  
3   "type": "response.text.done",  
4   "response_id": "resp_001",  
5   "item_id": "msg_007",  
6   "output_index": 0,  
7   "content_index": 0,  
8   "text": "Sure, I can help with that."  
9 }
```

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



```
1 {
2   "event_id": "event_4546",
3   "type": "response.audio_transcript.delta",
4   "response_id": "resp_001",
5   "item_id": "msg_008",
6   "output_index": 0,
7   "content_index": 0,
8   "delta": "Hello, how can I a"
9 }
```

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



```
1 {  
2   "event_id": "event_4748",  
3   "type": "response.audio_transcript.done",  
4   "response_id": "resp_001",  
5   "item_id": "msg_008",  
6   "output_index": 0,  
7   "content_index": 0,  
8   "transcript": "Hello, how can I assist you today?"  
9 }
```

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

```
1 {
2   "event_id": "event_4950",
3   "type": "response.audio.delta",
4   "response_id": "resp_001",
5   "item_id": "msg_008",
6   "output_index": 0,
7   "content_index": 0,
8   "delta": "Base64EncodedAudioDelta"
9 }
```

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

```
1 {
2   "event_id": "event_5152",
3   "type": "response.audio.done",
4   "response_id": "resp_001",
5   "item_id": "msg_008",
6   "output_index": 0,
7   "content_index": 0
8 }
```

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



```
1 {
2   "event_id": "event_5354",
3   "type": "response.function_call_arguments.delta",
4   "response_id": "resp_002",
5   "item_id": "fc_001",
6   "output_index": 0,
7   "call_id": "call_001",
```

```
8   "delta": "{\\"location\\": \\"San\\""}
9 }
```

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



```
1 {
2   "event_id": "event_5556",
3   "type": "response.function_call_arguments.done",
4   "response_id": "resp_002",
5   "item_id": "fc_001",
6   "output_index": 0,
7   "call_id": "call_001",
8   "arguments": "{\\"location\\": \\"San Francisco\\"}"
9 }
```

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

OBJECT `rate_limits.updated`



```
1  {
2    "event_id": "event_5758",
3    "type": "rate_limits.updated",
4    "rate_limits": [
5      {
6        "name": "requests",
7        "limit": 1000,
8        "remaining": 999,
9        "reset_seconds": 60
10     },
11     {
12       "name": "tokens",
13       "limit": 50000,
14       "remaining": 49950,
15       "reset_seconds": 60
16     }
17   ]
18 }
```

Completions Legacy

Given a prompt, the model will return one or more predicted completions along with the probabilities of alternative tokens at each position. Most developer should use our [Chat Completions API](#) to leverage our best and newest models.

Create completion Legacy