OBJECT The container file object


```json
{
    "id": "cfile_682e0e8a43c88191a7978f477a09bdf5",
    "object": "container.file",
    "created_at": 1747848842,
    "bytes": 880,
    "container_id": "cntr_682e0e7318108198aa783fd921ff305e08e78805b9fdbb04",
    "path": "/mnt/data/88e12fa445d32636f190a0b33daed6cb-tsconfig.json",
    "source": "user"
}
```


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.

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 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 conjunction 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

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

Response


```
{
  "id": "sess_001",
  "object": "realtime.session",
  "model": "gpt-4o-realtime-preview",
  "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,
  "speed": 1.1,
  "tracing": "auto",
  "client_secret": {
    "value": "ek_abc123",
    "expires_at": 1234567890
  }
}
```


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.

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

`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 conjunction 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

Example request curl

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

Response

```
{
  "id": "sess_BBwZc7cFV3XizEyKGDCGL",
  "object": "realtime.transcription_session",
  "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
}
```

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

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. 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"].

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`.

OBJECT The session object

```json
{
  "id": "sess_001",
  "object": "realtime.session",
  "model": "gpt-4o-realtime-preview",
  "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,
  "speed": 1.1,
  "tracing": "auto",
  "max_response_output_tokens": 200,
  "client_secret": {
    "value": "ek_abc123",
    "expires_at": 1234567890
  }
}
```

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 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

```
OBJECT The transcription session object
```

```json
{
  "id": "sess_BBwZc7cFV3XizEyKGDCGL",
  "object": "realtime.transcription_session",
  "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.

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`.

OBJECT session.update


```json
{
    "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",
        "speed": 1.1,
        "tracing": "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

```
{
    "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.

`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`.

OBJECT conversation.item.create

```json
{
    "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.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.

`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`.

OBJECT conversation.item.retrieve

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

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.

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`.

OBJECT conversation.item.truncate

```json
{
    "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.

item_id string

The ID of the item to delete.

type string

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

response object

Create a new Realtime response with these parameters

Show properties

type string

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

OBJECT response.create

```json
{
    "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.done` event with a status of `response.status=cancelled`. If there is no response to cancel, the server will respond with an error.

`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.logprobs",
    ]
  }
}
```

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

```
{
    "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 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.

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 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",
        "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.

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`.

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": []
    }
}
```

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 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. 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

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

OBJECT conversation.item.input_audio_transcription.completed


```json
{
    "event_id": "event_2122",
    "type": "conversation.item.input_audio_transcription.completed",
    "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,
    }
}
```


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_transcription.delta

```
{
  "type": "conversation.item.input_audio_transcription.delta",
  "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_transcription.failed


```json
{
    "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.

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

```
{
    "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

```json
{
    "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).

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`.

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.

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_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.

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 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`.

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.

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 with that."
            }
```

```
        ]
    }
}
```

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

```
{
    "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
The ID of the response.

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

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.

`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


```
{
    "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

```
{
    "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.

`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 you today?"
}
```

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.delta

```json
{
    "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.

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


```json
{
    "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\"}"
}
```


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_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,
      // "interrupt_response": false  -- this will NOT be returned
    },
    "input_audio_noise_reduction": {
      "type": "near_field"
    },
    "include": [
      "item.input_audio_transcription.avg_logprob",
    ],
  }
}
```

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


```json
{
    "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

```
{
    "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.

`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` .

OBJECT output_audio_buffer.stopped

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

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