

# Streaming

Streaming lets you subscribe to updates of the agent run as it proceeds. This can be useful for showing the end-user progress updates and partial responses.

To stream, you can call `Runner.run_streamed()`, which will give you a `RunResultStreaming`. Calling `result.stream_events()` gives you an async stream of `StreamEvent` objects, which are described below.

## Raw response events

`RawResponsesStreamEvent` are raw events passed directly from the LLM. They are in OpenAI Responses API format, which means each event has a type (like `response.created`, `response.output_text.delta`, etc) and data. These events are useful if you want to stream response messages to the user as soon as they are generated.

For example, this will output the text generated by the LLM token-by-token.

```
import asyncio
from openai.types.responses import ResponseTextDeltaEvent
from agents import Agent, Runner

async def main():
    agent = Agent(
        name="Joker",
        instructions="You are a helpful assistant.",
    )

    result = Runner.run_streamed(agent, input="Please tell me 5 jokes.")
    async for event in result.stream_events():
        if event.type == "raw_response_event" and isinstance(event.data,
            ResponseTextDeltaEvent):
            print(event.data.delta, end="", flush=True)

if __name__ == "__main__":
    asyncio.run(main())
```

## Run item events and agent events

`RunItemStreamEvent`s are higher level events. They inform you when an item has been fully generated. This allows you to push progress updates at the level of "message generated", "tool ran",

etc, instead of each token. Similarly, `AgentUpdatedStreamEvent` gives you updates when the current agent changes (e.g. as the result of a handoff).

For example, this will ignore raw events and stream updates to the user.

```
import asyncio
import random
from agents import Agent, ItemHelpers, Runner, function_tool

@function_tool
def how_many_jokes() -> int:
    return random.randint(1, 10)

async def main():
    agent = Agent(
        name="Joker",
        instructions="First call the `how_many_jokes` tool, then tell that many jokes.",
        tools=[how_many_jokes],
    )

    result = Runner.run_streamed(
        agent,
        input="Hello",
    )
    print("=== Run starting ===")

    async for event in result.stream_events():
        # We'll ignore the raw responses event deltas
        if event.type == "raw_response_event":
            continue
        # When the agent updates, print that
        elif event.type == "agent_updated_stream_event":
            print(f"Agent updated: {event.new_agent.name}")
            continue
        # When items are generated, print them
        elif event.type == "run_item_stream_event":
            if event.item.type == "tool_call_item":
                print("-- Tool was called")
            elif event.item.type == "tool_call_output_item":
                print(f"-- Tool output: {event.item.output}")
            elif event.item.type == "message_output_item":
                print(f"-- Message output:\n{ItemHelpers.text_message_output(event.item)}")
            else:
                pass # Ignore other event types

    print("=== Run complete ===")

if __name__ == "__main__":
```

```
asyncio.run(main())
```