How to stream custom data

Prerequisites

This guide assumes familiarity with the following:

- Streaming
- astream_events API
- Chat Models
- Tools

The most common use case for streaming from inside a node is to stream LLM tokens, but you may also want to stream custom data.

For example, if you have a long-running tool call, you can dispatch custom events between the steps and use these custom events to monitor progress. You could also surface these custom events to an end user of your application to show them how the current task is progressing.

You can do so in two ways: using graph's `.stream` / `.astream` methods with `stream_mode="custom"` emitting custom events using adispatch_custom_events.

Below we'll see how to use both APIs.

Setup

First, let's install our required packages

```
%%capture --no-stderr
%pip install -U langgraph
```

Set up LangSmith for LangGraph development

Sign up for LangSmith to quickly spot issues and improve the performance of your LangGraph projects. LangSmith lets you use trace data to debug, test, and monitor your LLM apps built with LangGraph — read more about how to get started here.

Stream custom data using `.stream / .astream`

Define the graph

```
from langchain_core.messages import AIMessage
from langgraph.graph import START, StateGraph, MessagesState, END
from langgraph.types import StreamWriter


async def my_node(
    state: MessagesState,
    writer: StreamWriter,  # <-- provide StreamWriter to write chunks to be streamed
):
    chunks = [
        "Four",
        "score",
        "and",
        "seven",
        "years",
        "ago",
        "our",
        "fathers",
        "...",
    ]
    for chunk in chunks:
        # write the chunk to be streamed using stream_mode=custom
        writer(chunk)

    return {"messages": [AIMessage(content=" ".join(chunks))]}


# Define a new graph
workflow = StateGraph(MessagesState)

workflow.add_node("model", my_node)
workflow.add_edge(START, "model")
workflow.add_edge("model", END)

app = workflow.compile()
```

API Reference: AIMessage | START | StateGraph | END

Stream content

```
from langchain_core.messages import HumanMessage

inputs = [HumanMessage(content="What are you thinking about?")]
async for chunk in app.astream({"messages": inputs}, stream_mode="custom"):
    print(chunk, flush=True)
```

```
Four
score
and
seven
years
ago
our
fathers
...
```

API Reference: HumanMessage

You will likely need to use multiple streaming modes as you will want access to both the custom data and the state updates.

```
from langchain_core.messages import HumanMessage

inputs = [HumanMessage(content="What are you thinking about?")]
async for chunk in app.astream({"messages": inputs}, stream_mode=["custom", "updates"]):
    print(chunk, flush=True)
```

```
('custom', 'Four')
('custom', ' score')
('custom', ' and')
('custom', ' seven')
('custom', ' years')
('custom', ' ago')
('custom', ' our')
('custom', ' fathers')
('custom', ' ...')
('updates', {'model': {'messages': [AIMessage(content='Four score and seven years ago our fathers ...', addition
```

API Reference: HumanMessage

Stream custom data using `.astream_events`

If you are already using graph's `.astream_events` method in your workflow, you can also stream custom data by emitting custom events using `adispatch_custom_event`

ASYNC IN PYTHON<=3.10

LangChain cannot automatically propagate configuration, including callbacks necessary for `astream_events()`, to child runnables if you are running async code in python<=3.10. This is a common reason why you may fail to see events being emitted from custom runnables or tools. If you are running python<=3.10, you will need to manually propagate the `RunnableConfig` object to the child runnable in async environments. For an example of how to manually propagate the config, see the implementation of the node below with `adispatch_custom_event`. If you are running python>=3.11, the `RunnableConfig` will automatically propagate to child runnables in async environment. However, it is still a good idea to propagate the `RunnableConfig` manually if your code may run in other Python versions.

Define the graph

```
from langchain_core.runnables import RunnableConfig, RunnableLambda
from langchain_core.callbacks.manager import adispatch_custom_event


async def my_node(state: MessagesState, config: RunnableConfig):
    chunks = [
        "Four",
        "score",
        "and",
        "seven",
        "years",
        "ago",
        "our",
        "fathers",
        "...",
    ]
    for chunk in chunks:
        await adispatch_custom_event(
            "my_custom_event",
            {"chunk": chunk},
            config=config,  # <-- propagate config
        )

    return {"messages": [AIMessage(content=" ".join(chunks))]}


# Define a new graph
workflow = StateGraph(MessagesState)

workflow.add_node("model", my_node)
workflow.add_edge(START, "model")
workflow.add_edge("model", END)

app = workflow.compile()
```

API Reference: RunnableConfig | RunnableLambda | adispatch_custom_event

Stream content

```
from langchain_core.messages import HumanMessage

inputs = [HumanMessage(content="What are you thinking about?")]
async for event in app.astream_events({"messages": inputs}, version="v2"):
    tags = event.get("tags", [])
    if event["event"] == "on_custom_event" and event["name"] == "my_custom_event":
        data = event["data"]
        if data:
            print(data["chunk"], end="|", flush=True)
```

```
Four|score|and|seven|years|ago|our|fathers|...|
```

API Reference: HumanMessage

Comments

2 reactions

2 comments · 6 replies — powered by giscus

Oldest Newest

saitharunsai Oct 4

required positional argument: 'writer'"

1 reply

vbarda Oct 4 Collaborator

could you try updating to the latest version of langgraph and see if it resolves the issue?

lnnary Oct 7

Argument `writer: StreamWriter` doesn't seem to be assigned directly in the code.
How does it work?
Doesn't it need to be bound to `my_node` using partial?

5 replies

vbarda Oct 7 Collaborator

@lnnary it is automatically injected in langgraph, you don't need to manually specify it anywhere except the node function signature

lnnary Oct 8 edited

> @lnnary it is automatically injected in langgraph, you don't need to manually specify it anywhere except the node function signature

Can I customize it? If so, how do I customize? Thank you

vbarda Oct 8 Collaborator

@lnnary could you please elaborate? what are you looking to customize?

rfouyang Oct 9

Updated to langgraph==0.2.35 and it still say missing 1 required positional argument: 'writer'

rfouyang Oct 9

I found the reason:

Previous, can mix non-stream and stream node together as the code in create_react_agent function did:

workflow.add_node("agent", RunnableCallable(call_model, acall_model))

But if doing this way, cannot use custom stream writer

Directly use workflow.add_node("agent", acall_model) is working

Any suggestion if can implement non-stream and stream together

Write Preview Aa

Sign in to comment

Sign in with GitHub

Table of contents