Python - Agent Development Kit

Source URL: https://google.github.io/adk-docs/get-started/streaming/quickstart-streaming/

Quickstart (Streaming / Python)¶

With this quickstart, you'll learn to create a simple agent and use ADK Streaming to enable voice and video communication with it that is low-latency and bidirectional. We will install ADK, set up a basic "Google Search" agent, try running the agent with Streaming with `adk web` tool, and then explain how to build a simple asynchronous web app by yourself using ADK Streaming and FastAPI.

Note: This guide assumes you have experience using a terminal in Windows, Mac, and Linux environments.

Supported models for voice/video streaming¶

In order to use voice/video streaming in ADK, you will need to use Gemini models that support the Live API. You can find the **model ID(s)** that supports the Gemini Live API in the documentation:

- Google AI Studio: Gemini Live API
- Vertex AI: Gemini Live API

1. Setup Environment & Install ADK¶

Create & Activate Virtual Environment (Recommended):

```
 # Create
python -m venv .venv
# Activate (each new terminal)
# macOS/Linux: source .venv/bin/activate
# Windows CMD: .venv\Scripts\activate.bat
# Windows PowerShell: .venv\Scripts\Activate.ps1
```

Install ADK:

```
pip install google-adk
```

2. Project Structure¶

Create the following folder structure with empty files:

```
adk-streaming/  # Project folder
└── app/ # the web app folder
    ├── .env # Gemini API key
    └── google_search_agent/ # Agent folder
        ├── __init__.py # Python package
        └── agent.py # Agent definition
```

agent.py¶

Copy-paste the following code block to the `agent.py`.

For `model`, please double check the model ID as described earlier in the Models section.

```
from google.adk.agents import Agent
from google.adk.tools import google_search  # Import the tool

root_agent = Agent(
   # A unique name for the agent.
   name="basic_search_agent",
   # The Large Language Model (LLM) that agent will use.
   model="gemini-2.0-flash-exp",
   # model="gemini-2.0-flash-live-001",  # New streaming model versior
   # A short description of the agent's purpose.
```

```
    description="Agent to answer questions using Google Search.",
    # Instructions to set the agent's behavior.
    instruction="You are an expert researcher. You always stick to the
    # Add google_search tool to perform grounding with Google search.
    tools=[google_search]
)
```

Note: To enable both text and audio/video input, the model must support the generateContent (for text) and bidiGenerateContent methods. Verify these capabilities by referring to the List Models Documentation. This quickstart utilizes the gemini-2.0-flash-exp model for demonstration purposes.

`agent.py` is where all your agent(s)' logic will be stored, and you must have a `root_agent` defined.

Notice how easily you integrated grounding with Google Search capabilities. The `Agent` class and the `google_search` tool handle the complex interactions with the LLM and grounding with the search API, allowing you to focus on the agent's *purpose* and *behavior*.

intro_components.png

Copy-paste the following code block to `__init__.py` file.

__init__.py

```
from . import agent
```

3. Set up the platform¶

To run the agent, choose a platform from either Google AI Studio or Google Cloud Vertex AI:

Gemini - Google AI StudioGemini - Google Cloud Vertex AI

1. Get an API key from Google AI Studio.

2. Open the `.env` file located inside (`app/`) and copy-paste the following code.

.env

``` GOOGLE_GENAI_USE_VERTEXAI=FALSE
GOOGLE_API_KEY=PASTE_YOUR_ACTUAL_API_KEY_HERE

`` 3. Replace`PASTE_YOUR_ACTUAL_API_KEY_HERE`with your actual`API KEY`.

1. You need an existing Google Cloud account and a project.
2. Set up a Google Cloud project
3. Set up the gcloud CLI
4. Authenticate to Google Cloud, from the terminal by running `gcloud auth login`.
5. Enable the Vertex AI API.
6. Open the `.env` file located inside (`app/`). Copy-paste the following code and update the project ID and location.

.env

``` GOOGLE_GENAI_USE_VERTEXAI=TRUE
GOOGLE_CLOUD_PROJECT=PASTE_YOUR_ACTUAL_PROJECT_ID
GOOGLE_CLOUD_LOCATION=us-central1

```

4. Try the agent with `adk web`¶

Now it's ready to try the agent. Run the following command to launch the **dev UI**. First, make sure to set the current directory to `app`:

```
cd app
```

Also, set `SSL_CERT_FILE` variable with the following command. This is required for the voice and video tests later.
```

```
export SSL_CERT_FILE=$(python -m certifi)
```

Then, run the dev UI:

```
adk web
```

Note for Windows users

When hitting the `_make_subprocess_transport NotImplementedError`, consider using `adk web --no-reload` instead.

Open the URL provided (usually `http://localhost:8000` or `http://127.0.0.1:8000`) **directly in your browser**. This connection stays entirely on your local machine. Select `google_search_agent`.

Try with text¶

Try the following prompts by typing them in the UI.

- What is the weather in New York?
- What is the time in New York?
- What is the weather in Paris?
- What is the time in Paris?

The agent will use the google_search tool to get the latest information to answer those questions.

Try with voice and video¶

To try with voice, reload the web browser, click the microphone button to enable the voice input, and ask the same question in voice. You will hear the answer in voice in real-time.

To try with video, reload the web browser, click the camera button to enable the video input, and ask questions like "What do you see?". The agent will answer what they see in the video input.

Stop the tool¶

Stop `adk web` by pressing `Ctrl-C` on the console.

Note on ADK Streaming¶

The following features will be supported in the future versions of the ADK Streaming: Callback, LongRunningTool, ExampleTool, and Shell agent (e.g. SequentialAgent).

Congratulations! You've successfully created and interacted with your first Streaming agent using ADK!

Next steps: build custom streaming app¶

In Custom Audio Streaming app tutorial, it overviews the server and client code for a custom asynchronous web app built with ADK Streaming and FastAPI, enabling real-time, bidirectional audio and text communication.