CSCE 631 — TAMU API Setup Guide

Dr. Alan Kuhnle
Texas A&M University

Summer I 2026

Key Takeaways

This guide walks you through setting up access to the TAMU API gateway for PA2. You will need to repeat the cookie extraction step (Step 2) each coding session, as cookies expire approximately every 24 hours.

1 What is the TAMU API?

Texas A&M provides access to several large language models (Claude, GPT, Gemini) through **chat.tamu.ai**, an OpenAI-compatible API gateway. For PA2, you will use this API to build multi-agent debate systems with real LLMs.

Access requires two credentials:

1. A **shared API key** (provided below — same for all students in the course). This key identifies the *course application* to the gateway, not any individual student.
2. A **Cloudflare Access cookie** from your browser session (personal to your NetID, expires every ~24 hours). This authenticates *you* and enforces your per-student usage quota.

Your usage is capped at **$5/day**, enforced per-student via the cookie. This is more than enough for PA2 if you follow the budgeting guidance in Section 7.

2 Step 1: Log into chat.tamu.ai

1. Open your browser and navigate to https://chat.tamu.ai
2. Click **"Sign in with TAMU SSO"**
3. Authenticate with your NetID and Duo MFA
4. You should see the chat interface — this confirms your session is active

You need an active browser session before proceeding. The session sets a Cloudflare Access cookie that your Python code will use.

3 Step 2: Get the Cloudflare Cookie

After logging in, you need to extract the `CF_Authorization` cookie from your browser. This cookie is marked **HttpOnly**, which means JavaScript in the console cannot access it via `document.cookie`. You must use one of the two DevTools methods below.

Method A: Application Tab

1. Open **DevTools** (press `F12`, or right-click anywhere and select "Inspect")
2. Go to the **Application** tab (in Chrome/Edge) or **Storage** tab (in Firefox)
3. In the left sidebar, expand **Cookies** and click on `https://chat.tamu.ai`
4. Find the cookie named `CF_Authorization`
5. Click on it and **copy the Value** — it is a long JWT string starting with `eyJ...`

Method B: Network Tab

1. Open **DevTools** (press `F12`) and go to the **Network** tab
2. **Refresh the page** (`Ctrl+R` or `F5`)
3. Click the **first request** to `chat.tamu.ai` in the list (usually the document request at the top)
4. In the panel that opens on the right, scroll down to **Request Headers**
5. Find the line starting with `Cookie:` — it contains multiple `name=value` pairs separated by semicolons
6. Copy everything from `CF_Authorization=eyJ...` through the end of that cookie value (it ends before the next `;`)

This method has the advantage that the raw cookie header shows the exact format you need to paste into your Python code.

Common Pitfalls

- This cookie expires after approximately **24 hours** — you will need to repeat Steps 1–2 each coding session.
- The cookie is personal to your NetID — do not share it.

4 Step 3: Install the OpenAI Python Library

The TAMU API is OpenAI-compatible, so we use the standard OpenAI Python client:

```
pip install openai
```

If you are using a virtual environment or conda, activate it first.

5 Step 4: Make Your First API Call

Create a Python file or Jupyter cell with the following code:

```
import openai

# Paste your CF_Authorization cookie value here (the eyJ... string)
CF_COOKIE = "CF_Authorization=eyJ..."  # <-- replace with your actual
    cookie

client = openai.OpenAI(
    base_url="https://chat.tamu.ai/api",
    api_key="sk-0183ed7c1c8e47c0a8f4d1e75161aa6d",
    default_headers={"Cookie": CF_COOKIE}
)
```

```
response = client.chat.completions.create(
    model="protected.Claude Sonnet 4.5",
    messages=[{"role": "user",
               "content": "What is a Nash equilibrium? "
                          "Answer in one sentence."}],
    temperature=1,        # Required for Claude thinking models on TAMU
    max_tokens=16384      # Must exceed the thinking budget (see below)
)

print(response.choices[0].message.content)
print(f"Tokens used: {response.usage.total_tokens}")
```

If you see a coherent answer about Nash equilibrium and a token count, your setup is working.

Common Pitfalls

Thinking Mode on Claude Models. The TAMU gateway enables *extended thinking* on Claude Sonnet and Opus models. This has three consequences:

1. `temperature` must be exactly `1` — other values produce an error.
2. `max_tokens` must be large (at least 16384) — the model uses some of these tokens for internal reasoning before producing the answer. A small value like 256 will fail.
3. The response includes reasoning content: in addition to `response.choices[0].message.content` (the actual answer), the response also contains `response.choices[0].message.reasoning_content` (the model's chain of thought). You can ignore the reasoning content for PA2.

Non-Claude models (GPT, Gemini) do **not** have this constraint and work with any `temperature` and small `max_tokens` values.

6 Available Models

The gateway provides access to 30+ models. The table below lists the most useful ones for PA2; use `client.models.list()` to see the full list.

Model identifier	Provider	Notes
`protected.Claude Sonnet 4.5`	Anthropic	**Recommended default.** Thinking mode: `temperature=1`, `max_tokens`≥16384.
`protected.Claude-Haiku-4.5`	Anthropic	Fastest Claude. No thinking constraints — works with any `temperature` and small `max_tokens`. Good for high-volume experiments.
`protected.Claude Opus 4.5`	Anthropic	Strongest reasoning, slower. Thinking mode.
`protected.gpt-5-mini`	OpenAI	Fast, cheap. Standard parameters (`temperature=0.7`, `max_tokens=256`).
`protected.gpt-5.2`	OpenAI	Strong reasoning model.
`protected.o3`	OpenAI	Reasoning specialist (chain-of-thought).
`protected.gemini-2.5-pro`	Google	Alternative for comparison experiments.
`protected.gemini-2.5-flash`	Google	Fast and cheap Google model.

Table 1: Selected models available through the TAMU API gateway.

Connection: PA2 Experiment 3.3

Use `protected.Claude Sonnet 4.5` as your default — best balance of quality and cost. For high-volume experiments, `protected.Claude-Haiku-4.5` or `protected.gpt-5-mini` are cheaper. Use a different provider (GPT or Gemini) for Experiment 3.3 (model comparison).

7 Budget Awareness

Your daily cap is **$5/day**. Here is a rough guide to costs:

- **Claude Sonnet 4.5:** approximately $3/M input tokens, $15/M output tokens
- A typical debate turn: ~200 tokens input (system prompt + history), ~100 tokens output (the agent's argument)
- One full debate game (2 agents, 3 rounds) = 6 API calls, roughly 1,800 total tokens
- **You can run 100+ debate games per day** on Sonnet without exceeding $5

Track your usage by checking the `usage` field in each API response:

```
print(f"Prompt tokens: {response.usage.prompt_tokens}")
print(f"Completion tokens: {response.usage.completion_tokens}")
print(f"Total tokens: {response.usage.total_tokens}")
```

Key Takeaways

Cost-saving tips:

- For Claude thinking models, `max_tokens` must be large (16384), but you are only billed for tokens actually produced, not the cap.
- For non-thinking models (Haiku, GPT, Gemini), keep `max_tokens` low (256 or less).
- Use `protected.Claude-Haiku-4.5` or `protected.gpt-5-mini` for high-volume experiments — significantly cheaper per token.

- Cache results: save API responses to a JSON file so you don't repeat identical calls.
- Start with small experiments (50 games) before scaling up.

8 Troubleshooting

Error	Cause	Fix
`401 Unauthorized` / `403 Forbidden`	Cookie expired	Log into chat.tamu.ai again and copy a fresh `CF_Authorization` cookie
`"model not found"` / `"invalid model"`	Wrong model name	Check that the model name exactly matches the table above, including the `protected.` prefix
Empty or nonsensical response	Content filtering	Rephrase your prompt; avoid adversarial or sensitive content
`429 Too Many Requests`	Rate limiting	Add `time.sleep(1)` between API calls
`openai.APIConnectionError`	Network issue	Check your internet connection; retry after a moment
No `usage` field in response	Some models omit usage stats	Wrap in `getattr(response.usage, 'total_tokens', 0)`

Table 2: Common errors and their solutions.

9 Important Notes

Common Pitfalls

- **Do NOT share the API key** outside this course.
- The `CF_Authorization` cookie is **personal to your NetID** — do not share it.
- Always include **both** the `Cookie` header and the `Authorization` header (the `api_key` parameter handles the latter).
- **Add error handling** in your debate code — API calls can fail intermittently.
- If you exhaust your $5/day budget, you will need to wait until the next day to continue.
- All API calls are logged — usage that violates TAMU's acceptable use policy may result in access revocation.