Aurora AI API Reference

Base URL

```
https://api.aurora.ai/v1
```

All endpoints are served over HTTPS. Plain HTTP is not supported.

---

1. Authentication

Aurora uses **API keys** for authentication.

1.1 API Key Format

API keys are opaque strings, e.g.:

```
aur_sk_live_xxxxyyyyzzzz...
```

-
- Keys are **tenant-scoped** and can have:
  - `read`
  - `write`
  - `admin`
    scopes.

1.2 Authorization Header

Include the API key in the `Authorization` header:

```
Authorization: Bearer aur_sk_live_xxxxyyyyzzzz
```

1.3 Error Responses for Auth

- `401 Unauthorized` – missing or invalid key
- `403 Forbidden` – key does not have permission for requested action

Example error payload:

```
{
  "error": {
    "code": "unauthorized",
    "message": "Invalid or missing API key.",
    "details": null
  }
}
```

---

2. Models

2.1 List Models

GET `/models`

Returns metadata about available Aurora models.

Request

```
GET /v1/models HTTP/1.1
Host: api.aurora.ai
Authorization: Bearer aur_sk_live_xxx
```

Response

```
{
  "data": [
    {
      "id": "aurora-72b-agent-v3",
      "type": "text-completion",
      "context_window": 256000,
      "max_output_tokens": 8192,
      "capabilities": {
        "tool_use": true,
```

```json
        "long_context": true,
        "code": true
      }
    },
    {
      "id": "aurora-16b-assistant",
      "type": "text-completion",
      "context_window": 128000,
      "max_output_tokens": 4096,
      "capabilities": {
        "tool_use": true,
        "long_context": true,
        "code": true
      }
    },
    {
      "id": "aurora-3b-edge",
      "type": "text-completion",
      "context_window": 16000,
      "max_output_tokens": 2048,
      "capabilities": {
        "tool_use": false,
        "long_context": false,
        "code": true
      }
    }
  ]
}
```

---

3. Chat Completions (Primary Endpoint)

Multi-turn conversation + tools + long context.

3.1 Create Chat Completion

POST `/chat/completions`

Request Body

```json
{
  "model": "aurora-72b-agent-v3",
  "messages": [
    {
      "role": "system",
      "content": "You are Aurora AI, a helpful technical assistant."
    },
    {
      "role": "user",
      "content": "Explain KV cache in transformers."
    }
  ],
  "tools": [
    {
      "name": "get_weather",
      "description": "Fetch the weather for a given city.",
      "input_schema": {
        "type": "object",
        "properties": {
          "location": { "type": "string" },
          "date": { "type": "string", "format": "date" }
        },
        "required": ["location"]
      }
    }
  ],
  "tool_choice": "auto",
  "max_tokens": 1024,
  "temperature": 0.2,
  "top_p": 0.9,
  "n": 1,
  "stream": false,
  "metadata": {
    "tenant_id": "acme-co",
    "request_id": "abc-123"
  }
}
```

Parameters

- `model` (string, required)
    - e.g., `aurora-72b-agent-v3`

`messages` (array, required)
Array of message objects:

```
{
  "role": "system" | "user" | "assistant" | "tool",
  "name": "optional_tool_or_user_name",
  "content": "text or tool result JSON-encoded as string"
}
```

-
- `tools` (array, optional)
  Tool definitions the model can call. See **§4** for full schema.
- `tool_choice` (string | object, optional)
    - `"auto"` – model decides when to call tools
    - `"none"` – tools disabled
    - `{"type": "tool", "name": "get_weather"}` – force a specific tool
- `max_tokens` (int, optional)
  Max number of tokens in the completion.
- `temperature` (float, optional, default `0.7`)
- `top_p` (float, optional, default `1.0`)
- `n` (int, optional)
  Number of completions to generate.
- `stream` (bool, optional)
  Whether to stream tokens over SSE.

- `metadata` (object, optional)
  Arbitrary key/value payload echoed back in logs.

Response

```
{
  "id": "chatcmpl_01JABCDXYZ",
  "object": "chat.completion",
  "created": 1769097600,
  "model": "aurora-72b-agent-v3",
  "usage": {
    "prompt_tokens": 134,
    "completion_tokens": 256,
    "total_tokens": 390
  },
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Key-value (KV) cache is a memory structure used
during transformer inference..."
      },
      "finish_reason": "stop",
      "logprobs": null
    }
  ]
}
```

Streaming Responses

If `stream: true`, Aurora returns an **SSE** stream:

```
POST /v1/chat/completions
Accept: text/event-stream
```

Events:

```
data:
{"id":"chatcmpl_...","object":"chat.completion.chunk","choices":[{"del
ta":{"role":"assistant","content":"KV "},"index":0}]}
data:
{"id":"chatcmpl_...","object":"chat.completion.chunk","choices":[{"del
ta":{"content":"cache "},"index":0}]}
...
data:
{"id":"chatcmpl_...","object":"chat.completion.chunk","choices":[{"del
ta":{},"finish_reason":"stop","index":0}]}
data: [DONE]
```

---

4. Tool Use & Agent Protocol

Aurora is trained to call tools using a structured schema similar to MCP/OpenAPI.

4.1 Tool Definition Format

Each tool object:

```
{
  "name": "get_weather",
  "description": "Fetches weather information for a city.",
  "input_schema": {
    "type": "object",
    "properties": {
      "location": {
        "type": "string",
        "description": "City name, e.g. 'Denver, CO'."
      },
      "date": {
        "type": "string",
        "format": "date",
        "description": "Optional ISO 8601 date."
      }
    },
    "required": ["location"]
  }
}
```

Constraints:

- `name` – must be a valid identifier: `[a-zA-Z0-9_-]+`
- `input_schema` follows JSON Schema (Draft 7-ish subset).

4.2 Tool Call Messages

When Aurora decides to call a tool, it responds with a message containing a `tool_calls` array.

Example assistant message:

```
{
  "role": "assistant",
  "content": null,
  "tool_calls": [
    {
      "id": "toolcall_01ABC",
      "type": "function",
      "name": "get_weather",
      "arguments": {
        "location": "Denver, CO",
        "date": "2025-11-22"
      }
    }
  ]
}
```

Notes:

- `arguments` is always a **JSON object**, already parsed for you.
- `content` is typically `null` for tool calls.

4.3 Tool Result Messages

Your application executes the tool and then **feeds the result back** as a new message:

```
{
  "role": "tool",
  "name": "get_weather",
  "tool_call_id": "toolcall_01ABC",
  "content": "{\"temperature_c\": 3.5, \"conditions\": \"snow\"}"
}
```

Aurora then uses this tool result to continue the conversation.

4.4 Example End-to-End Tool Call

1. Client request:

```
{
  "model": "aurora-72b-agent-v3",
  "messages": [
    { "role": "user", "content": "What's the weather in Denver today?"
}
  ],
  "tools": [
    {
      "name": "get_weather",
      "description": "Fetches today's weather.",
      "input_schema": {
        "type": "object",
        "properties": {
          "location": { "type": "string" }
        },
        "required": ["location"]
      }
    }
  ],
  "tool_choice": "auto"
}
```

2. Aurora responds (tool call):

```
{
  "choices": [
    {
      "message": {
        "role": "assistant",
        "tool_calls": [
          {
            "id": "toolcall_01ABC",
            "type": "function",
            "name": "get_weather",
            "arguments": {
              "location": "Denver, CO"
            }
          }
        ]
      }
    }
  ]
}
```

3. Your app executes `get_weather` and sends another API call:

```
{
  "model": "aurora-72b-agent-v3",
  "messages": [
    { "role": "user", "content": "What's the weather in Denver today?"
},
    {
      "role": "assistant",
      "tool_calls": [
        {
          "id": "toolcall_01ABC",
          "type": "function",
          "name": "get_weather",
          "arguments": { "location": "Denver, CO" }
        }
      ]
    },
    {
      "role": "tool",
      "name": "get_weather",
      "tool_call_id": "toolcall_01ABC",
      "content": "{\"temperature_c\": 3.5, \"conditions\": \"snow\"}"
    }
  ]
}
```

4. Aurora final answer:

```
{
  "choices": [
    {
      "message": {
        "role": "assistant",
        "content": "In Denver today it's around 3.5°C with snow."
      }
    }
  ]
}
```

---

5. Plain Completions (Non-Chat)

For raw text-in, text-out tasks (no roles/messages).

5.1 Create Completion

POST `/completions`

Request

```
{
  "model": "aurora-16b-assistant",
  "prompt": "Write a function in Python that computes factorial of
n.",
  "max_tokens": 256,
  "temperature": 0.3,
  "top_p": 1.0,
  "n": 1,
  "stop": ["```"],
  "stream": false
}
```

Response

```
{
  "id": "cmpl_01ABA",
  "object": "text.completion",
  "created": 1769097680,
  "model": "aurora-16b-assistant",
  "usage": {
    "prompt_tokens": 24,
    "completion_tokens": 78,
    "total_tokens": 102
  },
  "choices": [
    {
      "index": 0,
      "text": "def factorial(n):\n    if n < 0:\n        raise ValueError(\"n must be non-negative\")\n    result = 1\n    for i in range(2, n+1):\n        result *= i\n    return result\n",
      "finish_reason": "stop",
      "logprobs": null
    }
  ]
}
```

---

6. Embeddings

Generate vector embeddings for text (e.g., RAG, semantic search).

6.1 Create Embeddings

POST `/embeddings`

Request

```
{
  "model": "aurora-embeddings-1024-v1",
  "input": [
    "Explain rotary position embeddings.",
    "KV cache stores key and value tensors across layers."
  ],
  "encoding_format": "float"
}
```

Response

```
{
  "object": "list",
  "model": "aurora-embeddings-1024-v1",
  "data": [
    {
      "index": 0,
      "embedding": [0.0231, -0.1043, 0.9932, ...]
    },
    {
      "index": 1,
      "embedding": [0.1189, 0.0021, 0.4023, ...]
    }
  ],
  "usage": {
    "prompt_tokens": 35,
    "total_tokens": 35
  }
}
```

Parameters

- `model` – e.g. `aurora-embeddings-1024-v1`
- `input` – string or array of strings
- `encoding_format` – `"float"` or `"base64"`

---

7. Agents (High-Level Orchestrated Workflows)

A convenience layer that lets Aurora manage planning, tool calls, and final answers in a single endpoint.

7.1 Execute Agent

POST `/agents/execute`

Request

```
{
  "agent_id": "default-aurora-agent",
  "input": "Summarize this repo and generate a migration plan.",
  "context_documents": [
    {
      "id": "doc_1",
      "title": "README.md",
      "content": "..."
    },
    {
      "id": "doc_2",
      "title": "MIGRATION_GUIDE.md",
      "content": "..."
    }
  ],
  "max_steps": 8,
  "return_plan": true,
  "stream": false
}
```

Response

```
{
  "id": "agt_01XYZ",
  "object": "agent.run",
  "agent_id": "default-aurora-agent",
  "created": 1769097800,
  "steps": [
    {
      "index": 0,
      "type": "plan",
      "content": "<agent_plan>1) Read docs. 2) Extract current stack.
3) Propose migration plan...</agent_plan>"
    },
    {
      "index": 1,
      "type": "tool_call",
      "tool_name": "search_repo",
      "arguments": { "query": "architecture overview" },
      "result": { "matches": ["doc_1", "doc_2"] }
    }
  ],
  "output": {
    "role": "assistant",
    "content": "Here's a summary of the repo and a suggested migration
plan..."
  },
  "usage": {
    "prompt_tokens": 300,
    "completion_tokens": 700,
    "total_tokens": 1000
  }
}
```

---

8. Rate Limiting & Quotas

8.1 Headers

Every response includes rate-limit headers:

```
X-Aurora-Limit-Requests: 3000
X-Aurora-Remaining-Requests: 2789
X-Aurora-Reset-Requests: 1769101200

X-Aurora-Limit-Tokens: 300000
```

```
X-Aurora-Remaining-Tokens: 284523
X-Aurora-Reset-Tokens: 1769101200
```

- `*Limit-*` – quota per rolling window
- `*Remaining-*` – remaining allowance
- `*Reset-*` – UNIX timestamp when quota resets

8.2 Rate-Limit Error

```
{
  "error": {
    "code": "rate_limit_exceeded",
    "message": "Token quota exceeded for this window.",
    "details": {
      "retry_after_seconds": 30
    }
  }
}
```

---

9. Errors

All error responses use a common envelope:

```
{
  "error": {
    "code": "invalid_request",
    "message": "The 'model' field is required.",
    "details": {
      "field": "model",
      "location": "body"
    }
  }
}
```

9.1 Common Error Codes

- `invalid_request` – malformed JSON, missing fields, invalid parameter range
- `unauthorized` – invalid API key
- `forbidden` – insufficient permissions
- `not_found` – unknown model or resource
- `rate_limit_exceeded` – quotas exceeded
- `server_error` – internal Aurora error
- `overloaded` – temporary capacity issue, retry with backoff

---

10. Safety Controls

10.1 Safe-Mode Parameters (Chat & Completions)

You can control safety behavior:

```
{
  "safety": {
    "level": "standard",
    "allow_deep_technical": true,
    "blocked_categories": ["self_harm", "weapons"]
  }
}
```

- `level` – `"strict" | "standard" | "relaxed"` (tenant-configurable defaults)
- `blocked_categories` – override to explicitly block certain content domains

10.2 Safety Metadata in Responses

```
{
  "choices": [
    {
      "message": { "role": "assistant", "content": "..." },
      "safety": {
        "policy_applied": true,
        "categories": {
          "self_harm": "none",
          "harassment": "none",
          "hate": "none"
        }
      }
    }
  ]
}
```