ChatGPT

Search Tools

Agentic search is a core capability of xAI's agentic tool calling. The `grok-4-fast` model is trained to use search tools to iteratively query sources and follow up on responses, allowing the agent to navigate web pages and X posts and uncover information that might otherwise be hard to find [1]. To use the agentic tool calling API via the xAI Python SDK you must use version 1.3.1 or later [2].

Available Search Tools

You can enable the following server-side search tools:

- **Web Search** – allows the agent to search the web and browse pages [3].
- **X Search** – performs keyword, semantic and user search on X and can fetch threads [4].

Specify which tools you need by listing them in the `tools` parameter of your request [5]. The mapping between tool names in the xAI SDK and in the OpenAI responses API is shown below.

Tool	xAI SDK	OpenAI Responses API
Web Search	`web_search`	`web_search` [6]
X Search	`x_search`	`x_search` [6]

Retrieving Citations

Citations provide traceability for sources used during agentic search. In the xAI Python SDK you can access them from the response object. The example below shows how to stream a conversation and then print the citations, usage and server-side tool calls:

```python
import os

from xai_sdk import Client
from xai_sdk.chat import user
from xai_sdk.tools import web_search

client = Client(api_key=os.getenv("XAI_API_KEY"))
chat = client.chat.create(
    model="grok-4-fast",  # reasoning model
    tools=[web_search()],
)
```

```python
chat.append(user("What is xAI?"))

is_thinking = True
for response, chunk in chat.stream():
    # View the server-side tool calls as they are being made in real-time
    for tool_call in chunk.tool_calls:
        print(f"\nCalling tool: {tool_call.function.name} with arguments: "
              f"{tool_call.function.arguments}")
    if response.usage.reasoning_tokens and is_thinking:
        print(f"\rThinking... ({response.usage.reasoning_tokens} tokens)",
end="", flush=True)
    if chunk.content and is_thinking:
        print("\n\nFinal Response:")
        is_thinking = False
    if chunk.content and not is_thinking:
        print(chunk.content, end="", flush=True)

print("\n\nCitations:")
print(response.citations)
print("\n\nUsage:")
print(response.usage)
print(response.server_side_tool_usage)
print("\n\nServer Side Tool Calls:")
print(response.tool_calls)
```

The `citations` array contains URLs of all sources encountered during the search. Not every URL will be relevant to the final answer because the agent may review a source and decide it is not useful [7]. See the overview page for more details on citations [8].

Applying Search Filters to Control Agentic Search

Both search tools support optional parameters to narrow the search space and limit the sources or data considered [9]. Supported filter parameters are shown below:

Tool	Supported filter parameters
Web Search	`allowed_domains`, `excluded_domains`, `enable_image_understanding` [10]
X Search	`allowed_x_handles`, `excluded_x_handles`, `from_date`, `to_date`, `enable_image_understanding`, `enable_video_understanding` [10]

Web Search Parameters

Only Search in Specific Domains

Use `allowed_domains` to restrict web search to pages within specified domains [11]. You can include up to five domains and cannot combine this parameter with `excluded_domains` [12].

Example using the xAI SDK:

```
import os

from xai_sdk import Client
from xai_sdk.chat import user
from xai_sdk.tools import web_search

client = Client(api_key=os.getenv("XAI_API_KEY"))
chat = client.chat.create(
    model="grok-4-fast",
    tools=[
        web_search(allowed_domains=["wikipedia.org"]),
    ],
)

chat.append(user("What is xAI?"))

# stream or sample the response...
```

Exclude Specific Domains

Use `excluded_domains` to prevent the agent from searching or browsing pages on specified domains [13]. You can exclude up to five domains. It cannot be used together with `allowed_domains` [14].

Example:

```
import os

from xai_sdk import Client
from xai_sdk.chat import user
from xai_sdk.tools import web_search

client = Client(api_key=os.getenv("XAI_API_KEY"))
chat = client.chat.create(
    model="grok-4-fast",
    tools=[
        web_search(excluded_domains=["wikipedia.org"]),
    ],
)

chat.append(user("What is xAI?"))

# stream or sample the response...
```

Enable Image Understanding

Setting `enable_image_understanding` to `True` equips the agent with a `view_image` tool so that it can fetch and analyze images encountered during search [15]. When this tool is invoked you will see entries in `chunk.tool_calls` and `response.tool_calls` and the server-side tool usage will include `SERVER_SIDE_TOOL_VIEW_IMAGE` [16]. Enabling image understanding increases token usage because images are processed and represented as tokens [17]. Enabling it for web search will also enable image understanding for X Search if both tools are included in your request [18].

Example:

```
import os

from xai_sdk import Client
from xai_sdk.chat import user
from xai_sdk.tools import web_search

client = Client(api_key=os.getenv("XAI_API_KEY"))
chat = client.chat.create(
    model="grok-4-fast",
    tools=[
        web_search(enable_image_understanding=True),
    ],
)

chat.append(user("What is included in the image in xAI's official website?"))

# stream or sample the response...
```

X Search Parameters

Only Consider X Posts from Specific Handles

Use `allowed_x_handles` to only consider posts from specific X handles [19]. You can specify up to 10 handles and cannot combine this with `excluded_x_handles` [20].

Example:

```python
import os

from xai_sdk import Client
from xai_sdk.chat import user
from xai_sdk.tools import x_search

client = Client(api_key=os.getenv("XAI_API_KEY"))
chat = client.chat.create(
    model="grok-4-fast",
    tools=[
        x_search(allowed_x_handles=["elonmusk"]),
    ],
)

chat.append(user("What is the current status of xAI?"))

# stream or sample the response...
```

Exclude X Posts from Specific Handles

Use `excluded_x_handles` to prevent the agent from including posts from specified X handles [21]. Up to 10 handles can be excluded. You cannot use `excluded_x_handles` together with `allowed_x_handles` [22].

Example:

```python
import os

from xai_sdk import Client
from xai_sdk.chat import user
from xai_sdk.tools import x_search

client = Client(api_key=os.getenv("XAI_API_KEY"))
chat = client.chat.create(
    model="grok-4-fast",
    tools=[
        x_search(excluded_x_handles=["elonmusk"]),
    ],
)

chat.append(user("What is the current status of xAI?"))

# stream or sample the response...
```

Date Range

You can restrict the date range of data used by specifying `from_date` and/or `to_date` [23]. Both fields should be in ISO 8601 date format (YYYY-MM-DD). When using the xAI SDK you may also pass `datetime.datetime` objects for these parameters. If only `from_date` is specified, the agent uses data from that date until today; if only `to_date` is specified, it uses data up to the specified date [23].

Example:

```python
import os
from datetime import datetime

from xai_sdk import Client
from xai_sdk.chat import user
from xai_sdk.tools import x_search

client = Client(api_key=os.getenv("XAI_API_KEY"))
chat = client.chat.create(
    model="grok-4-fast",
    tools=[
        x_search(
            from_date=datetime(2025, 10, 1),
            to_date=datetime(2025, 10, 10),
        ),
    ],
)

chat.append(user("What is the current status of xAI?"))

# stream or sample the response...
```

Enable Image Understanding

Setting `enable_image_understanding` to `True` for the X Search tool gives the agent access to the `view_image` tool, allowing it to fetch and analyze images in X posts [24]. Like web search, using image understanding increases token usage and will cause `SERVER_SIDE_TOOL_VIEW_IMAGE` to appear in the server-side tool usage [25]. Enabling this parameter for X Search also enables image understanding for Web Search if it is included [26].

Example:

```python
import os

from xai_sdk import Client
from xai_sdk.chat import user
from xai_sdk.tools import x_search

client = Client(api_key=os.getenv("XAI_API_KEY"))
chat = client.chat.create(
    model="grok-4-fast",
    tools=[
        x_search(enable_image_understanding=True),
    ],
)

chat.append(user("What images are being shared in recent xAI posts?"))

# stream or sample the response...
```

Enable Video Understanding

Setting `enable_video_understanding` to `True` equips the agent with a `view_x_video` tool so it can fetch and analyze video content from X posts [27]. When invoked, this tool appears in `chunk.tool_calls` and `response.tool_calls` with a `video_url` parameter, and `SERVER_SIDE_TOOL_VIEW_X_VIDEO` will appear in `response.server_side_tool_usage` [28]. Processing video content uses additional tokens [29].

Example:

```
import os

from xai_sdk import Client
from xai_sdk.chat import user
from xai_sdk.tools import x_search

client = Client(api_key=os.getenv("XAI_API_KEY"))
chat = client.chat.create(
    model="grok-4-fast",
    tools=[
        x_search(enable_video_understanding=True),
    ],
)

chat.append(user("What is the latest video talking about from the xAI official X
account?"))

# stream or sample the response...
```

The sections above describe how to configure xAI’s search tools and control the agent’s behavior through filter parameters and optional features. Use these parameters to focus the agent’s search on specific domains, users or time periods and to enable or disable image and video understanding features as needed.

---

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29

Search Tools

https://docs.x.ai/docs/guides/tools/search-tools