OCTUS

Octus MCP Connector

Developer Guide

This article provides information for developers integrating the Octus MCP Connector into their own MCP client or agent framework. For non-technical setup against existing clients (Claude.ai), see the integrations guide.

MCP Server Endpoint

Octus MCP Server is a remote MCP server that uses Streamable HTTP. Any compatible client app — including agent frameworks like LangChain, LangGraph, Semantic Kernel, and the official MCP SDKs — can connect.

```
https://mcp.octus.com/mcp
```

Common format to configure MCP servers in integrated development environments:

```
{
 "octus": {
  "type": "http",
  "url": "https://mcp.octus.com/mcp"
 }
}
```

Although the Octus MCP Connector provides a public endpoint, it is not an Application Programming Interface (API) in the traditional sense. Developers should integrate through an agent framework or MCP-aware client, not access the endpoint directly. The interface may change dynamically — including the set of available tools and their request and response formats. Using it requires Octus subscription and login credentials in order for Octus MCP Server to authenticate

Best practices

- **Use an agent framework**
  Don't hand-roll JSON-RPC against the endpoint; rely on an MCP-aware client or SDK so tool schemas and OAuth state are handled correctly.

- **Refresh the tool list per session**
  Tools are entitlement-filtered and may change — read tools/list at session start rather than caching tool names in code.

- **Surface the citation claims**
  Responses include claim-level source bindings; render them as inline links so users can trace each statement back to the underlying Octus document.

- **Let the model route**
  Each search_<source> tool ships with a description tuned for natural-language routing. If a question is ambiguous, fall back to list_available_tools and let the user pick.

Authentication

The server implements the MCP OAuth Specification. On first connection, the user is redirected to an Octus authentication page and, after a successful sign-in, returned to the client. Subsequent tool calls reuse the established session — no API keys or bearer tokens to manage manually. See the integrations guide for the user-facing flow.

Input and output shape

Every search tool takes a single natural-language question and returns a structured response containing the synthesized answer, source-bound claims for inline citations, and suggested follow-up questions. The exact JSON schema is published by the server itself — read it from the live tools/list response your client receives at connect time, so your integration stays aligned with the current contract.

Related

Integrations guide — end-user setup for Claude.ai and other MCP clients.

Tools

We encourage developers to integrate with the Octus MCP Connector through an agent framework. Every time the client initializes the server, it should call the standard tools/list to retrieve the available tools — the set is filtered per user based on their Octus entitlements and may evolve over time, so we deliberately don't enumerate request and response details here.

The current tools fall into three groups:

- **search_<source>** — one tool per Octus content source (intel, SEC filings, transcripts, court filings, fundamentals, covenants, deal-term analytics, data rooms, uploads, private-company analysis).
- **search_across_all_sources** — searches every source the user is entitled to in one call.
- **list_available_tools** — returns the tools the user can access plus a short description for each. Useful for client-side disambiguation pickers.

For human-readable descriptions of every tool, see Available tools in the integrations guide.