FROM THE WORKBENCH · NOVEMBER 28, 2026

How to Wire WordPress MCP for Grok, Claude, and Coding Terminals

By Will Tygart · Tygart Media · Canonical guide & .md twin at https://tygartmedia.com/how-to-wire-wordpress-mcp-grok-claude-session-teardown/

Direct Answer: To wire WordPress to Grok, Claude, or terminal MCP clients cleanly, connect to the Streamable HTTP endpoint /wp-json/mcp/v1/mcp under the 2025-03-26 MCP specification, advertise RFC 8414 OAuth metadata or WordPress Application Passwords, handle HTTP 405 GET SSE probes cleanly, and close every session via DELETE returning HTTP 204.

Before connecting an AI assistant to WordPress, most developers fall back on bulk REST scraping — pulling dozens of pages of paginated post JSON just to answer a single question. In our 24-hour origin audit in Grok Knocked on Our WordPress Door 86 Times Using MCP, bulk scrapers transferred 84.45 MB over `/wp-json/wp/v2/posts` while `xAI-Grok-Connectors` ran 86 clean MCP sessions at a fraction of the bandwidth.

Under the Streamable HTTP MCP transport spec, WordPress does not need long-lived background daemons or open WebSockets. Every step — authentication, capability discovery, JSON-RPC tool calls, optional SSE checks, and explicit teardown — rides standard stateless or session-header HTTP verbs you can test in twenty lines of Python.

The Five-Verb WordPress MCP Lifecycle

Inspecting real connector logs shows why partial client wrappers fail. Clients that skip the `notifications/initialized` step or forget DELETE leave orphaned state, while naive WAF rules break OAuth metadata discovery before authentication even begins. Expect these exact HTTP codes:

Session Stage	HTTP Method & Endpoint	Expected Status	Protocol Purpose
1. Auth / Token Refresh	POST OAuth token URL or Basic Auth header	HTTP 200 OK	Issues scoped bearer token (or validates WordPress Application Password)
2. Initialize Handshake	POST /wp-json/mcp/v1/mcp (initialize)	HTTP 200 OK	Returns server capabilities plus Mcp-Session-Id response header
3. Client Ready Notice	POST /wp-json/mcp/v1/mcp (notifications/initialized)	HTTP 202 Accepted	Confirms capability agreement before issuing tools/list or tools/call
4. Optional Stream Probe	GET /wp-json/mcp/v1/mcp (Accept: text/event-stream)	HTTP 405 Method Not Allowed	Stateless WordPress handlers decline optional SSE cleanly per spec
5. Clean Session Teardown	DELETE /wp-json/mcp/v1/mcp (Mcp-Session-Id)	HTTP 204 No Content	Frees session state cleanly on exit inside try/finally blocks

When publishing an externally discoverable connector for xAI Grok or Claude Desktop, pair your MCP endpoint with an RFC 8414 OAuth metadata document and publish a plain Agent Handshake specification so visiting bots read your supported endpoints rather than brute-forcing guesswork paths.

Client Configuration and OAuth Discovery Endpoints

Most modern agent hosts accept a compact Streamable HTTP block pointing directly at `/wp-json/mcp/v1/mcp`. Avoid older deprecated `serverUrl` or SSE-only paths unless routing through an explicit stdio bridge:

mcp_config.json + WordPress OAuth & Application Password Endpoints

```
{
"mcpServers": {
"wordpress-production": {
"url": "https://example.com/wp-json/mcp/v1/mcp",
"headers": {
"Accept": "application/json, text/event-stream",
"User-Agent": "Mozilla/5.0 (compatible; Studio-MCP-Client/1.0)"
}
}
}
}
```

Self-Testing Handshake Harness and Real-World Traps

WordPress MCP Blocker	What Breaks Live	Why It Tricks Builders	Clean Production Fix
Edge Firewall 403 / 1010	Python-urllib or default Curl UAs rejected at edge proxy	Local curl works from allowlisted shell while remote agents get 403	Send standard User-Agent and exempt signed /wp-json/mcp/v1/ calls
UTF-8 BOM in JSON Credential Files	Windows PowerShell prepends 0xEFBBBF BOM to JSON secrets	json.loads fails or base64 auth token includes corrupt leading bytes	Always read secret JSON files with encoding='utf-8-sig'
Blanket /wp-json/ Firewall Block	Security rule blocks entire REST prefix to stop user scans	Breaks oEmbed, Application Passwords, and all MCP discovery together	Block only /wp/v2/users enumeration and query author=N scans cleanly
Orphaned Sessions Without DELETE	Client scripts exit after tools/call without cleanup	Test harness works 10 times before hitting server session limits	Always wrap session calls in try/finally issuing HTTP DELETE

Stdlib Python Streamable HTTP MCP Probe With Clean HTTP 204 DELETE Teardown

```
import base64, json, pathlib, urllib.request

def load_wp_basic_header(secret_file: pathlib.Path) -> tuple[str, str]:
data = json.loads(secret_file.read_text(encoding="utf-8-sig"))
token = base64.b64encode(f"{data['username']}:{data['application_password']}".encode()).decode()
return data["site"].rstrip("/") + "/wp-json/mcp/v1/mcp", f"Basic {token}"

def mcp_post(url: str, auth: str, payload: dict, session_id: str | None = None):
hdrs = {
"Content-Type": "application/json",
"Accept": "application/json, text/event-stream",
"Authorization": auth,
"User-Agent": "Mozilla/5.0 Chrome/131.0.0.0 Safari/537.36",
}
if session_id:
hdrs["Mcp-Session-Id"] = session_id
req = urllib.request.Request(url, data=json.dumps(payload).encode("utf-8"), headers=hdrs, method="POST")
with urllib.request.urlopen(req, timeout=20) as resp:
raw = resp.read().decode("utf-8", errors="replace")
return resp.status, resp.headers.get("Mcp-Session-Id", session_id), (json.loads(raw) if raw else {})

def smoke_test_mcp(secret_file: pathlib.Path) -> None:
url, auth = load_wp_basic_header(secret_file)
st, sid, _ = mcp_post(url, auth, {"jsonrpc": "2.0", "id": 1, "method": "initialize", "params": {}})
assert st == 200 and sid
try:
st2, _, _ = mcp_post(url, auth, {"jsonrpc": "2.0", "method": "notifications/initialized"}, sid)
assert st2 in (200, 202)
st3, _, tools = mcp_post(url, auth, {"jsonrpc": "2.0", "id": 2, "method": "tools/list", "params": {}}, sid)
assert st3 == 200 and "result" in tools
finally:
del_req = urllib.request.Request(url, headers={"Authorization": auth, "Mcp-Session-Id": sid}, method="DELETE")
with urllib.request.urlopen(del_req, timeout=15) as r_del:
assert r_del.status in (200, 204)
```

Treat Your API Door Like Your Storefront

As we explored in Skills vs. MCP vs. Connectors vs. Plugins, exposing clean, typed tools lets external AI systems verify facts and run workflows without downloading gigabytes of noisy HTML. Run the teardown probe regularly so your endpoint stays as clean on call ten thousand as on day one.

> **Workbench Next Step:** Clean WordPress MCP connectors follow a strict 5-verb handshake capped by HTTP 204 DELETE teardown. Continue at https://tygartmedia.com/how-to-wire-voice-notes-metamuse-notion-markdown-inbox/