

BookStack MCP Server – Installationsanleitung

Kontext

Diese Anleitung dokumentiert die Einrichtung des `bookstack-mcp-server` (npm-Paket von `pnocera`), der Claude über das Model Context Protocol direkten Zugriff auf diese BookStack-Instanz gibt.

Voraussetzungen

- Laufende BookStack-Instanz mit Zugriff auf einen Benutzer-Account
- Die Rolle des Accounts benötigt die Berechtigung **"Access system API"** (BookStack: Settings → Roles → <Rolle> → Bereich "System")
- **Bun-Runtime ≥ 1.1.0** installiert und im PATH
 - Das npm-Paket liefert kein kompiliertes JavaScript aus, sondern startet die TypeScript-Quelle direkt über das Shebang `#!/usr/bin/env bun`. **Node/npx allein reicht nicht aus** – Bun muss auf der Maschine vorhanden sein.
 - Prüfen: `bun --version`
 - Installation z. B. via winget: `winget install Oven-sh.Bun`
- Claude Code CLI installiert

Schritt 1: API-Token in BookStack erzeugen

1. In BookStack einloggen.
2. Oben rechts auf das Profil-Icon klicken → **Edit Profile**.
3. Tab **API Tokens** öffnen → **Create Token**.
4. Namen vergeben, Token erzeugen.
5. **Token-ID** und **Token-Secret** notieren (Secret wird nur einmal angezeigt).
6. Für die Konfiguration werden beide Teile mit Doppelpunkt kombiniert: `token_id:token_secret`.

Schritt 2: MCP-Server zu Claude Code hinzufügen

```
claude mcp add bookstack --scope user \  
  --env BOOKSTACK_BASE_URL=https://<deine-bookstack-domain>/api \  
  --env BOOKSTACK_API_TOKEN=<token_id>:<token_secret> \  
  --env MCP_TRANSPORT=stdio \  
  -- npx -y bookstack-mcp-server
```

Wichtige Punkte:

- `BOOKSTACK_BASE_URL` muss zwingend auf `/api` enden – nicht nur die reine Domain angeben.
- `MCP_TRANSPORT=stdio` ist notwendig. Ohne diese Variable startet der Server im HTTP-Modus, verlangt zusätzlich einen `MCP_AUTH_TOKEN` (separates Bearer-Secret für eingehende Verbindungen, z. B. via `openssl rand -hex 32`) und einen freien Port (Default `3000`).
- `--scope user` legt den Server user-weit an, nicht nur projektbezogen.

Schritt 3: Verbindung prüfen

- In Claude Code `/mcp` ausführen → Server `bookstack` sollte als verbunden angezeigt werden.
- Testaufruf, z. B. `bookstack_search` oder `bookstack_books_list` → sollte JSON-Daten liefern, **kein** HTML.

Claude Desktop (statt/zusätzlich zu Claude Code)

Grundsätzlich derselbe JSON-Block unter `mcpServers`, aber ein paar Desktop-spezifische Fallstricke:

- **Config-Pfad bei paketierter Installation** (App-Installer/MSIX-Variante): nicht der klassische Pfad `%APPDATA%\Claude\claude_desktop_config.json`, sondern `%LOCALAPPDATA%\Packages\Claude_pzs8srxjxfjc\LocalCache\Roaming\Claude\claude_desktop_config.json`. Am zuverlässigsten über die App selbst finden: **Settings → Developer → "Edit Config"**.

- **MCP_TRANSPORT: "stdio" nicht vergessen** – ohne diese Env-Variable startet `bookstack-mcp-server` im HTTP-Modus (Port 3000, zusätzlich `MCP_AUTH_TOKEN` nötig) und bricht beim Start sofort ab, weil dieser Token fehlt. Claude Desktop meldet dann nur "kann nicht verbinden", ohne den eigentlichen Grund zu nennen.
- **Die App läuft nach dem Schließen des Fensters im Hintergrund weiter** (Tray-Icon) und schreibt die Config-Datei bei jedem Öffnen/Schließen neu (v. a. den `preferences`-Block). Der `mcpServers`-Block wurde dabei im Test nicht überschrieben – trotzdem zur Sicherheit: Datei nur bearbeiten, wenn die App über **Tray-Icon → Beenden** vollständig beendet wurde, danach neu starten. Nur ein Fenster zu schließen reicht nicht.
- Minimales `mcpServers`-Beispiel für Claude Desktop:

```
{
  "mcpServers": {
    "bookstack": {
      "command": "npx",
      "args": ["-y", "bookstack-mcp-server"],
      "env": {
        "BOOKSTACK_BASE_URL": "https://<domain>/api",
        "BOOKSTACK_API_TOKEN": "<token_id>:<token_secret>",
        "MCP_TRANSPORT": "stdio"
      }
    }
  }
}
```

- Verifikation: im Chat-Eingabefeld erscheint ein Stecker-/Werkzeug-Icon, darunter sollte `bookstack` als verbunden aufgeführt sein.

Troubleshooting

Symptom	Ursache	Fix
Tool liefert rohes HTML (Login-Seite) statt JSON	<code>BOOKSTACK_BASE_URL</code> ohne <code>/api</code> -Suffix → Anfrage landet auf normaler Web-Route statt REST-API	URL um <code>/api</code> ergänzen
<code>MCP error -32600: Requested resource not found</code> bei <code>bookstack_system_info</code>	Einzelnes Tool/Endpoint-Problem, nicht grundsätzlich	Mit anderem Tool (z. B. <code>bookstack_search</code>) erneut testen
401 / Auth-Fehler	<code>BOOKSTACK_API_TOKEN</code> nicht im Format <code>token_id:token_secret</code>	Token in BookStack neu erzeugen, beide Teile mit <code>:</code> kombinieren
Änderung an <code>~/.claude.json</code> wirkt nicht	Laufender stdio-Prozess wurde mit alten Env-Variablen gestartet	Claude-Code-Session neu starten

Symptom	Ursache	Fix
Server startet gar nicht (nur im HTTP-Modus relevant)	<code>BOOKSTACK_API_TOKEN</code> fehlt komplett	Env-Variable setzen, Prozess neu starten

Sicherheitshinweise

- Der BookStack-API-Token gewährt API-Zugriff im vollen Umfang der Rolle des Accounts, für den er erzeugt wurde (ggf. inkl. Create/Update/Delete auf Books/Pages/Users) – als Secret behandeln, nicht in Klartext-Notizen, Tickets oder Chats teilen.
- `~/.claude.json` liegt lokal und unverschlüsselt auf der Maschine; Zugriff entsprechend einschränken.
- Falls statt Stdio der HTTP-Transport genutzt wird: zusätzlich `MCP_AUTH_TOKEN` generieren (z. B. `openssl rand -hex 32`). Dieser ist **nicht** identisch mit dem BookStack-Token und schützt den eingehenden `/message`-Endpoint.

Alternative: offizielle Bun-Variante

Das Paket-README empfiehlt eigentlich `bunx` statt `npx`:

```
claude mcp add bookstack --scope user \  
  --env BOOKSTACK_BASE_URL=https://<domain>/api \  
  --env BOOKSTACK_API_TOKEN=<token_id>:<token_secret> \  
  --env MCP_TRANSPORT=stdio \  
  -- bunx bookstack-mcp-server
```

In der bestehenden Konfiguration funktioniert auch `npx -y bookstack-mcp-server`, weil npm auf Windows das Shebang (`#!/usr/bin/env bun`) im generierten Wrapper-Skript berücksichtigt und automatisch `bun` aufruft, sofern es im PATH gefunden wird.

Andere MCP-Server-Implementierungen für BookStack (falls Alternative gewünscht)

- [pnocera/bookstack-mcp-server](#) (aktuell genutzt; npm: `bookstack-mcp-server`; 59 Tools; benötigt Bun) — <https://github.com/pnocera/bookstack-mcp-server>
- [h4sht/bookstack-mcp](#) (25 Tools; reines Node.js, kein Bun nötig; Installation: `claude mcp add bookstack -- npx github:h4sht/bookstack-mcp`) — <https://github.com/h4sht/bookstack-mcp>
- [oculairmedia/Bookstack-MCP](#) (Python/FastMCP, Fokus u. a. auf Bild-Galerie-Workflows)
- weitere Optionen: [yellowgg2/mcp-bookstack](#), [pacyworld/bookstack-mcp](#) (Multi-Instanz), [Derron-Knox/bookstack-mcp-server](#)

Quellen

- [pnocera/bookstack-mcp-server - README](#)
- [h4sht/bookstack-mcp - README](#)
- [BookStack - Hacking BookStack / API-Doku](#)

Erstellt von Claude Sonnet 5 (claude-sonnet-5) via API, ausgehend von der Obsidian-Notiz
[Themes/BookStack MCP/2026-09-17 BookStack MCP Installationsanleitung.md](#).

Revision #2

Created 17 September 2026 07:00:18 by Stefan Mechler

Updated 17 September 2026 09:46:27 by Stefan Mechler