Supermemory MCP

Handbuch für Claude

Persistentes Gedächtnis für KI-Assistenten

Universelles Memory über alle MCP-Clients hinweg

Version 1.0

Februar 2026

Inhaltsverzeichnis

1 Einführung

1.1 Was ist Supermemory?

Supermemory ist ein **Cloud-basierter Memory-Service** für Large Language Models (LLMs), der über das Model Context Protocol (MCP) angebunden wird. Der Dienst ermöglicht es KI-Assistenten wie Claude, Informationen über Sitzungen hinweg zu speichern und abzurufen.

Kernfunktionen:

- **Universelles Memory** – Gleiche Memories in allen MCP-Clients verfügbar
- **Geräteübergreifende Synchronisation** – Zugriff von Desktop, Web und Mobile
- **Automatische Deduplizierung** – Keine doppelten Einträge
- **Semantische Suche** – Findet relevante Memories basierend auf Bedeutung
- **User Profile Aggregation** – Erstellt automatisch strukturierte Profile
- **Projekt-Scoping** – Organisiert Memories nach Projekten

1.2 Anwendungsfälle

Anwendungsfall	Beschreibung
Entwickler-Workflow	Technische Präferenzen, Projekt-Kontexte und Code-Patterns über IDE-Wechsel hinweg beibehalten
Persönlicher Assistent	Termine, Präferenzen und persönliche Fakten speichern
Recherche	Erkenntnisse aus mehreren Sitzungen aggregieren
Team-Arbeit	Projekt-spezifische Informationen mit Team-Mitgliedern teilen

1.3 Architektur-Übersicht

MCP Client	⟷	**Supermemory API**
(Claude, Cursor)	OAuth/API Key	(api.supermemory.ai)

↓ MCP Protocol

Supermemory MCP Server
`mcp.supermemory.ai/mcp`

Cloudflare Durable Objects:
- Session State
- Memory Storage
- Profile Generation

2 Voraussetzungen

2.1 Systemanforderungen

Komponente	Anforderung
Node.js	Version 18 oder höher
npm/npx	Aktuell (mit Node.js installiert)
Internet	Stabile Verbindung (Cloud-Dienst)
Browser	Für OAuth-Authentifizierung

2.2 Unterstützte MCP-Clients

- Claude.ai (Web-Interface)
- Claude Desktop (macOS, Windows, Linux)
- Claude Code (CLI)
- Cursor IDE
- VS Code (mit Copilot)
- Windsurf
- Cline
- Gemini CLI

2.3 Account erstellen

1. Besuche `https://app.supermemory.ai`
2. Melde dich an mit Google, GitHub oder E-Mail
3. Du erhältst automatisch eine **persönliche URL** im Format:
   `https://mcp.supermemory.ai/[USER_ID]/sse`

Tipp

Die persönliche URL wird automatisch generiert und ist permanent. Du kannst sie jederzeit im Dashboard abrufen.

2.4 API-Key generieren (optional)

Für erweiterte Nutzung oder Self-Hosting:

1. Besuche `https://console.supermemory.ai`
2. Navigiere zu **API Keys**

3. Klicke auf **Create New Key**
4. Kopiere den Key (Format: `sm_xxxxxxxxxxxxxxxx`)

Wichtig

Der API-Key wird nur einmal angezeigt! Speichere ihn sicher ab.

3 Installation

3.1 One-Click Installation (Empfohlen)

Die einfachste Methode für die meisten Clients:

1. Besuche https://app.supermemory.ai
2. Melde dich an
3. Klicke auf **„Connect to your AI"**
4. Wähle deinen Client (z.B. Claude Desktop, Cursor)
5. Klicke auf den entsprechenden **„Add to [Client]"** Button
6. Bestätige im Client-Popup

3.2 Manuelle Installation via CLI

Universelle Methode für alle Clients:

```
# Allgemeines Format
npx -y install-mcp@latest https://mcp.supermemory.ai/mcp \
    --client [CLIENT] --oauth=yes

# Beispiele fuer verschiedene Clients
npx -y install-mcp@latest https://mcp.supermemory.ai/mcp --client claude --oauth=yes
npx -y install-mcp@latest https://mcp.supermemory.ai/mcp --client cursor --oauth=yes
npx -y install-mcp@latest https://mcp.supermemory.ai/mcp --client windsurf --oauth=yes
```

Info

Der `install-mcp` Befehl konfiguriert den Client automatisch und öffnet den Browser für die OAuth-Authentifizierung.

3.3 Claude.ai (Web-Interface)

Claude.ai unterstützt Remote-MCP-Server nativ:

1. Öffne Claude.ai → **Settings** → **Integrations**

2. Klicke auf **„Add Integration"**
3. Wähle **Supermemory** aus der Liste
4. Authentifiziere dich über OAuth (Browser-Popup)

3.4 Claude Desktop

3.4.1 Konfigurationsdatei finden

Betriebssystem	Pfad
macOS	`~/Library/Application Support/Claude/claude_desktop_config.json`
Windows	`%APPDATA%\Claude\claude_desktop_config.json`
Linux	`~/.config/Claude/claude_desktop_config.json`

3.4.2 Konfiguration mit URL (SSE Transport)

```
{
  "mcpServers": {
    "supermemory": {
      "transport": "sse",
      "url": "https://mcp.supermemory.ai/mcp"
    }
  }
}
```

3.4.3 Konfiguration mit API-Key

```
{
  "mcpServers": {
    "supermemory": {
      "url": "https://mcp.supermemory.ai/mcp",
      "headers": {
        "Authorization": "Bearer sm_dein_api_key_hier"
      }
    }
  }
}
```

3.4.4 Konfiguration mit Supergateway (npx)

```
{
  "mcpServers": {
    "supermemory": {
      "command": "npx",
      "args": ["-y", "supergateway", "--sse",
               "https://mcp.supermemory.ai/[USER_ID]/sse"]
```

```
    }
  }
}
```

3.5 Claude Code (CLI)

```
# Mit OAuth (empfohlen)
claude mcp add supermemory -- npx -y install-mcp \
    https://mcp.supermemory.ai/mcp --oauth=yes

# Alternativ: Direkte Konfiguration
claude mcp add supermemory --url https://mcp.supermemory.ai/mcp

# Installation verifizieren
claude mcp list
```

3.6 Cursor IDE

3.6.1 Globale Konfiguration

Bearbeite `~/.cursor/mcp.json`:

```
{
  "mcpServers": {
    "supermemory": {
      "command": "npx",
      "args": ["-y", "@supermemory/mcp"]
    }
  }
}
```

3.6.2 Projekt-spezifische Konfiguration

Erstelle `.cursor/mcp.json` im Projektordner:

```
{
  "mcpServers": {
    "supermemory": {
      "url": "https://mcp.supermemory.ai/mcp",
      "headers": {
        "x-sm-project": "mein-projekt"
      }
    }
  }
}
```

3.7 VS Code

Erstelle `.vscode/mcp.json` im Projektordner:

```
{
  "mcpServers": {
    "supermemory": {
      "command": "npx",
      "args": ["-y", "@supermemory/mcp"]
    }
  }
}
```

3.8 Nach der Installation

1. **Client neu starten** – Schließe und öffne den Client vollständig
2. **Tools prüfen** – Die Supermemory-Tools sollten verfügbar sein
3. **Test durchführen** – Frage Claude: *„Welche Memory-Tools hast du?"*

4 Verfügbare Tools

Supermemory stellt vier MCP-Tools bereit:

4.1 memory – Speichern und Vergessen

Speichert neue Informationen oder löscht bestehende Memories.

Parameter	Typ	Beschreibung
`action`	string	`"save"` oder `"forget"` (Standard: save)
`content`	string	Der Inhalt (max. 200.000 Zeichen)
`containerTag`	string	Optional: Projekt-Scope

Beispiel – Speichern:

```
{
  "action": "save",
  "content": "Nutzer bevorzugt Python und VS Code als IDE."
}
```

Beispiel – Mit Projekt-Scope:

```
{
  "action": "save",
  "content": "API-Endpoint: api.example.com/v2",
```

```
  "containerTag": "projekt_alpha"
}
```

Beispiel – Vergessen:

```
{
  "action": "forget",
  "content": "Veraltete Information die nicht mehr gilt."
}
```

4.2 recall – Suchen und Abrufen

Durchsucht die gespeicherten Memories semantisch.

Parameter	Typ	Beschreibung
`query`	string	Suchanfrage (max. 1.000 Zeichen)
`includeProfile`	boolean	User Profile einbeziehen (Standard: true)
`containerTag`	string	Optional: Nur in Projekt suchen

Beispiel:

```
{
  "query": "Programmiersprachen Praeferenzen",
  "includeProfile": true
}
```

Rückgabe-Struktur:

```
{
  "User Profile": {
    "Recent context": [
      "Arbeitet aktuell an Projekt X",
      "Bevorzugt TypeScript"
    ],
    "Persistent facts": [
      "Software-Entwickler",
      "Nutzt macOS"
    ]
  },
  "Relevant Memories": [
    {
      "id": "mem_abc123",
      "content": "Bevorzugt Python fuer Data Science",
      "match": "78%"
    },
    {
      "id": "mem_def456",
      "content": "IDE: VS Code mit Vim-Extension",
      "match": "65%"
    }
```

```
  ]
}
```

4.3 listProjects – Projekte auflisten

Zeigt alle verfügbaren Projekt-Container an.

Parameter	Typ	Beschreibung
`refresh`	boolean	Liste vom Server neu laden (Standard: true)

Beispiel-Rückgabe:

```
{
  "projects": [
    "sm_project_default",
    "projekt_alpha",
    "webapp_redesign"
  ]
}
```

4.4 whoAmI – Benutzerinfo

Gibt Informationen über den authentifizierten Benutzer zurück.

```
{
  "user_id": "usr_abc123",
  "email": "user@example.com",
  "plan": "pro",
  "memory_count": 142
}
```

5 Betrieb und Best Practices

5.1 Was sollte gespeichert werden?

Tipp

Speichere **Fakten und Präferenzen**, nicht Konversationen. Supermemory aggregiert automatisch ein strukturiertes User Profile.

Empfohlene Inhalte:

- Technische Präferenzen (Sprachen, Tools, Frameworks)

- Projekt-Kontexte und Zustände
- Wichtige Entscheidungen und deren Begründungen
- API-Endpoints und Konfigurationen
- Workflow-Beschreibungen
- Persönliche Präferenzen (Kommunikationsstil, Formatierung)

Nicht empfohlen:

- Passwörter, API-Keys, Secrets
- Vollständige Konversationen
- Temporäre oder einmalige Informationen
- Sensible persönliche Daten

5.2 Memory-Formulierung

Gut formulierte Memories:

```
"Nutzer ist Senior Backend-Entwickler mit 8 Jahren Erfahrung"
"Bevorzugt funktionale Programmierung ueber OOP"
"Projekt X: React-Frontend, FastAPI-Backend, PostgreSQL"
"Code-Review-Stil: Bevorzugt konstruktives Feedback mit Beispielen"
```

Schlecht formulierte Memories:

```
"Okay" (zu vage)
"User fragte nach Hilfe" (keine Information)
"Das Projekt" (zu unspezifisch)
```

5.3 Projekte effektiv nutzen

Organisiere Memories nach Projekten für bessere Übersicht:

```
# Projekt-spezifisches Memory speichern
{
  "action": "save",
  "content": "Sprint 42: Authentication-Modul fertig",
  "containerTag": "webapp_v2"
}

# Nur in Projekt suchen
{
  "query": "Sprint Status",
  "containerTag": "webapp_v2"
}
```

5.4 Automatisches Speichern aktivieren

Um Claude anzuweisen, wichtige Informationen automatisch zu speichern:

Option 1: User Preferences (Claude.ai)

1. Öffne **Settings** → **Profile** → **User Preferences**
2. Füge hinzu: *„Speichere wichtige Erkenntnisse aus Gesprächen automatisch in Supermemory."*

Option 2: Direkte Anweisung

Sage Claude:

> *„Bitte speichere ab jetzt alle wichtigen Informationen, die du über mich oder meine Projekte lernst, automatisch in Supermemory."*

5.5 Memory-Hygiene

Regelmäßige Pflege der Memories:

1. **Überprüfen** – Nutze `recall` mit breiten Queries um alle Memories zu sehen
2. **Aufräumen** – Lösche veraltete Informationen mit `forget`
3. **Aktualisieren** – Speichere korrigierte Versionen
4. **Konsolidieren** – Fasse ähnliche Memories zusammen

6 Troubleshooting

6.1 Häufige Probleme und Lösungen

Problem	Lösung
Tools nicht sichtbar	Client vollständig neu starten. MCP-Konfiguration auf JSON-Syntax prüfen.
OAuth-Fehler	Browser-Cookies löschen. Inkognito-Fenster für Auth nutzen.
„No memories found"	Breitere Suchbegriffe verwenden. `includeProfile: true` setzen.
Timeout bei Recall	Internetverbindung prüfen. Später erneut versuchen.
Duplikate gespeichert	Supermemory dedupliziert automatisch. Bei Bedarf `forget` nutzen.
API-Key ungültig	Neuen Key in Console generieren. Format `sm_...` prüfen.

6.2 Konfiguration validieren

```
# JSON-Syntax pruefen (macOS/Linux)
cat ~/.cursor/mcp.json | python3 -m json.tool

# Node.js Version pruefen
node --version  # Muss >= 18 sein

# npx testen
npx -y @supermemory/mcp --help

# MCP-Server-Liste anzeigen (Claude Code)
claude mcp list
```

6.3 Logs prüfen

Claude Desktop:

```
# macOS
tail -f ~/Library/Logs/Claude/mcp*.log

# Windows (PowerShell)
Get-Content "$env:APPDATA\Claude\logs\mcp*.log" -Wait

# Linux
tail -f ~/.config/Claude/logs/mcp*.log
```

Cursor:

```
# Developer Tools oeffnen:
# macOS: Cmd+Shift+I
# Windows/Linux: Ctrl+Shift+I
# Console-Tab fuer Fehler pruefen
```

6.4 Verbindung testen

```
# Server-Erreichbarkeit pruefen
curl -I https://mcp.supermemory.ai/mcp

# Mit Auth-Header
curl -H "Authorization: Bearer sm_your_key" \
     https://mcp.supermemory.ai/mcp
```

7 Datenschutz und Sicherheit

Achtung

Supermemory ist ein **Cloud-Dienst**. Alle Memories werden auf Supermemory-Servern gespeichert. Speichere **niemals**:

- Passwörter oder API-Keys
- Kreditkarten- oder Bankdaten
- Sozialversicherungsnummern oder Ausweisdaten
- Vertrauliche Geschäftsgeheimnisse
- Gesundheitsdaten

7.1 Daten löschen

Einzelne Memories:

```
{
  "action": "forget",
  "content": "Der exakte Inhalt der zu loeschenden Memory"
}
```

Alle Daten löschen:

1. Besuche https://console.supermemory.ai
2. Navigiere zu **Settings** → **Account**
3. Klicke auf **Delete All Data** oder **Delete Account**

7.2 Self-Hosting (Volle Datenkontrolle)

Für maximale Datenkontrolle kann Supermemory selbst gehostet werden:

```
# Repository klonen
git clone https://github.com/supermemoryai/supermemory-mcp.git
cd supermemory-mcp

# Umgebungsvariablen setzen
echo "SUPERMEMORY_API_KEY=sm_your_key" > .env

# Server starten
npm install
npm run dev
```

7.3 Alternative: Lokale Memory-Lösungen

Für vollständig lokale Speicherung ohne Cloud:

Lösung	Beschreibung
`@modelcontextprotocol/server-memory`	Offizieller Anthropic Memory Server mit SQLite
`mcp-memory-service`	Feature-reich mit Embeddings und Web-Dashboard
`memory-bank-mcp`	Projekt-spezifische Markdown-Notizen
`mcp-knowledge-graph`	Strukturierte Knowledge Graphs

8 Referenz

8.1 Alle MCP-Tools im Überblick

Tool	Parameter	Beschreibung
`memory`	action, content, containerTag	Speichert oder löscht Memories
`recall`	query, includeProfile, containerTag	Semantische Suche
`listProjects`	refresh	Zeigt verfügbare Projekte
`whoAmI`	–	Benutzerinformationen

8.2 Unterstützte Clients

Client	Plattform	Transport
Claude.ai	Web	Native Integration
Claude Desktop	macOS, Windows, Linux	SSE/stdio
Claude Code	CLI	stdio
Cursor	macOS, Windows, Linux	stdio
VS Code	macOS, Windows, Linux	stdio
Windsurf	macOS, Windows, Linux	stdio
Cline	VS Code Extension	stdio
Gemini CLI	CLI	stdio

8.3 Nützliche Links

- **Web-App:** `https://app.supermemory.ai`
- **Console:** `https://console.supermemory.ai`
- **Dokumentation:** `https://docs.supermemory.ai`
- **GitHub (MCP):** `https://github.com/supermemoryai/supermemory-mcp`
- **GitHub (Main):** `https://github.com/supermemoryai/supermemory`
- **Install-MCP CLI:** `https://github.com/supermemoryai/install-mcp`

- **MCP Protokoll:** https://modelcontextprotocol.io

8.4 Beispiel-Konfigurationen

Minimale Konfiguration (URL-basiert):

```
{
  "mcpServers": {
    "supermemory": {
      "url": "https://mcp.supermemory.ai/mcp"
    }
  }
}
```

Mit API-Key und Projekt:

```
{
  "mcpServers": {
    "supermemory": {
      "url": "https://mcp.supermemory.ai/mcp",
      "headers": {
        "Authorization": "Bearer sm_xxx",
        "x-sm-project": "mein-projekt"
      }
    }
  }
}
```

npx-basiert (Cursor/VS Code):

```
{
  "mcpServers": {
    "supermemory": {
      "command": "npx",
      "args": ["-y", "@supermemory/mcp"]
    }
  }
}
```

9 Changelog

Version	Datum	Änderungen
1.0	Februar 2026	Erstveröffentlichung