ChatGPT

MCP-connector voor ChatGPT en Topdesk: opzet en gebruik

Wat is het Model Context Protocol (MCP)?

Definitie: MCP is een open protocol dat standaardiseert hoe applicaties context (zoals tools of data) aanbieden aan Large Language Models (LLM's). Denk aan MCP als een soort *"USB-C-poort"* voor AI-toepassingen: een uniforme manier om AI-modellen te verbinden met verschillende databronnen en tools [1]. Met MCP kunnen ontwikkelaars functies en gegevens van externe systemen aanbieden aan een taalmodel op een gestandaardiseerde wijze.

ChatGPT en connectors: OpenAI's ChatGPT ondersteunt dit protocol om externe diensten te koppelen via zogeheten **connectors**. Veel van de ingebouwde integraties in ChatGPT (bijv. voor Outlook, Teams, Gmail, Google Drive, Dropbox, etc.) zijn eigenlijk MCP-gebaseerde connectors die door OpenAI worden onderhouden [2] [3]. Voor zakelijke teams (Enterprise/Edu) en individuele Pro-gebruikers is het bovendien mogelijk om **eigen MCP-servers (custom connectors)** toe te voegen in ChatGPT [3]. Dit betekent dat een organisatie haar eigen systeem (zoals Topdesk) via een MCP-interface kan aanbieden, zodat ChatGPT daarop kan "zoeken, redeneren en acties uitvoeren" binnen de toegestane context van die organisatie [3].

Hoe ChatGPT omgaat met een MCP-connector

Tool-lijst en oproepen: Wanneer een MCP-server als custom connector is toegevoegd, zal ChatGPT (of nauwkeuriger: het achterliggende OpenAI model) eerst de lijst van beschikbare *tools* opvragen bij de MCP-server. Elke tool komt overeen met een bepaalde functie of actie die de server aanbiedt (bijvoorbeeld "haal incident op", "maak nieuw incident aan", etc.). ChatGPT voegt deze lijst van tools, inclusief naam, beschrijving en parameters, toe aan zijn context. Vervolgens kan het model tijdens een gesprek besluiten een tool te gebruiken: het genereert dan een functie-aanroep naar de MCP-server met de benodigde parameters. De MCP-server voert de actie uit en retourneert het resultaat, dat ChatGPT weer in het gesprek verwerkt als antwoord of als vervolgactie.

Vereisten en best practices: In de praktijk is gebleken dat ChatGPT vooral goed overweg kan met custom connectors als er een generieke zoekfunctionaliteit aanwezig is. Vaak implementeren connectors een tool genaamd `search` voor algemene zoekvragen. De Topdesk MCP voorziet inderdaad in zo'n `search`**-functie** die zoekt naar incidenten op basis van (deel van) de titel [4]. Daarnaast is er een kortere `fetch`**-functie** toegevoegd om direct een incident op ID of nummer op te halen [5]. Deze extra tools dienen als gebruiksvriendelijke shortcuts naast de specifiek benoemde Topdesk-tools – ze komen overeen met respectievelijk *incidenten zoeken* en *incident ophalen*, en helpen het model om de juiste actie te kiezen. (Er zijn meldingen dat de ChatGPT interface custom connectors zonder een `search`-tool soms niet toont of minder gemakkelijk activeert, vandaar dat het verstandig is zo'n tool te voorzien als ingang voor het model.)

MCP-configuratie in ChatGPT: Het toevoegen van de Topdesk MCP aan ChatGPT vereist doorgaans dat een beheerder de **server-URL** en een label invoert in de ChatGPT instellingen (beschikbaar voor ondersteunende abonnementen/edities). ChatGPT verwacht dat de MCP-server luistert op een bepaald

eindpunt (meestal `/mcp`) en communiceert volgens het protocol. Een configuratievoorbeeld (zoals gebruikt in de OpenAI Agents SDK) ziet er als volgt uit:

```
{
  "type": "mcp",
  "server_label": "topdesk-mcp",
  "server_url": "https://jouwdomein.example.com/mcp",
  "require_approval": "never"
}
```

Hierbij is `server_label` een naam voor de server, en `server_url` de publiek bereikbare MCP-endpoint URL. In ChatGPT zal een toegevoegde custom connector doorgaans verschijnen onder de opgegeven labelnaam, en het model kan de bijbehorende tools dan gebruiken zonder extra tussenkomst.

De Topdesk MCP-server: opzet en functies

De **Topdesk MCP-server** is een Python-gebaseerde toepassing die het MCP-protocol implementeert voor de TOPdesk servicedesk-omgeving. Het project maakt gebruik van de officiële TOPdesk API (via de bestaande **TOPdeskPy SDK** [6] met enkele uitbreidingen) om acties uit te voeren in TOPdesk, en van de **FastMCP** library [7] om MCP-communicatie te verzorgen.

Configuratie en credentials: Om verbinding te maken met de TOPdesk API verwacht de server een aantal omgevingsvariabelen. Belangrijk zijn: - `TOPDESK_URL` – de base URL van de TOPdesk-instance (bijv. `https://jouwbedrijf.topdesk.net`) [8]. - `TOPDESK_USERNAME` – de gebruikersnaam waaronder het API-token is aangemaakt (meestal een integration account of de eigen username) [8]. - `TOPDESK_PASSWORD` – het API-token zelf, dat hier als wachtwoord wordt gebruikt [8].

Deze credentials worden bij opstart ingelezen; de code zal stoppen met een fout als ze niet gezet zijn [9]. Zodra ze aanwezig zijn, wordt een verbinding gemaakt via de SDK (er wordt een `topdesk_client` geïnitialiseerd) [9].

Transportmodi: De Topdesk MCP kan draaien als: - **Stdio-process** – direct als een lokaal proces dat via stdin/stdout met het model communiceert (handig voor lokale Agents setups) [10]. - **HTTP-server (streamable)** – een eigen webservice die HTTP-verzoeken en -antwoorden (evt. streaming) afhandelt op een host/poort (standaard host `0.0.0.0` en poort `3030`) [11]. Hierbij wordt het MCP-protocol over HTTP uitgevoerd (de server exposeert onder andere een `/mcp` endpoint). - **HTTP-server (SSE)** – een variant waarbij gebruik wordt gemaakt van Server-Sent Events (event streaming) [11].

Men kan de modus kiezen via de env var `TOPDESK_MCP_TRANSPORT` (`'stdio'`, `'streamable-http'` of `'sse'`) [11]. In een gehoste situatie (zoals op Render) is een HTTP-modus gebruikelijk. De testpagina (`/test`) die genoemd is, maakt vermoedelijk deel uit van de webserver modus en stelt je in staat de tools handmatig uit te proberen via een eenvoudige UI.

Interne werking: Bij opstart registreert de server al zijn functies als **tools** bij FastMCP. In de code gebeurt dit via decorators `@mcp.tool()` boven elke functie. Hierdoor krijgt elke Python-functie een toolnaam (standaard de functienaam) en omschrijving mee, die in de MCP **list_tools** output verschijnt [12] [13]. De Topdesk MCP gebruikt grotendeels de functienamen als toolnamen, bijvoorbeeld een Python-functie `topdesk_get_incident(...)` wordt een tool `"topdesk_get_incident"` met de

docstring als beschrijving. Bij sommige tools is extra logica toegevoegd – zo zijn er anonieme functies `search` en `fetch` in de code die respectievelijk incidenten zoeken en ophalen, om het modelgebruik te vergemakkelijken [4] [5], hoewel deze niet expliciet in de README werden genoemd.

Beschikbare Topdesk-tools (functies)

De Topdesk MCP-server stelt een breed scala aan functies beschikbaar, vergelijkbaar met wat je via de TOPdesk REST API kunt doen. Hieronder een overzicht van alle tools (functies) zoals beschreven in de documentatie, gegroepeerd op categorie, met korte toelichting:

Algemene hulpmiddelen: - `topdesk_get_fiql_query_howto` – Geeft uitleg en voorbeelden over het construeren van FIQL-query's (filter queries) voor TOPdesk [14]. Handig om gebruikers te informeren hoe ze zoekfilters kunnen opbouwen.
- `topdesk_get_object_schemas` – Haalt de volledige objectschema's op voor TOPdesk-incidenten en alle bijbehorende subvelden [15]. Dit biedt inzicht in de datavelden/structuur die TOPdesk hanteert.

Incident-gerelateerde tools: - `topdesk_get_incident` – Haalt een TOPdesk-incident op aan de hand van een *incident-ID (UUID)* óf *incidentnummer* (formaat `I-xxxxxx-xxx`). Beide vormen worden geaccepteerd [16]. (De functie detecteert het formaat en gebruikt het juiste API-eindpunt voor ID of nummer.)
- `topdesk_get_incidents_by_fiql_query` – Zoekt incidenten via een gegeven FIQL-query en retourneert de lijst van overeenkomende incidenten [17]. Hiermee kun je op allerlei velden filteren (zoals status, prioriteit, etc.).
- `topdesk_get_incident_user_requests` – Geeft alle *user requests* (gebruikersverzoeken/meldingen) terug die aan een bepaald incident gekoppeld zijn [18]. Zo kun je bijv. zien welke eindgebruikerverzoeken onder één incident vallen.
- `topdesk_create_incident` – Maakt een nieuw incident aan in TOPdesk [19]. Vereist doorgaans een `caller_id` (de melder) en een set velden (`incident_fields` dictionary) om het incident te beschrijven (zoals titel, omschrijving, categorie, etc.).
- `topdesk_archive_incident` – Archiveert het opgegeven incident (op ID of nummer) [20]. Een gearchiveerd incident wordt historisch gemaakt en verdwijnt uit actieve overzichten.
- `topdesk_unarchive_incident` – Haalt een gearchiveerd incident weer uit het archief (maakt het weer actief) [21].
- `topdesk_get_timespent_on_incident` – Haalt alle geregistreerde bestede-tijd entries op voor een gegeven incident [22]. Je krijgt een lijst van tijdregistraties (bijv. wie heeft hoeveel minuten geboekt).
- `topdesk_register_timespent_on_incident` – Registreert nieuwe bestede tijd op een incident (met gegeven duur in minuten) [22]. Dit voegt een tijdboekings-entry toe aan het incident.
- `topdesk_escalate_incident` – Escaleert een incident (markeert het als geëscaleerd) met een meegegeven *reden-ID* van de escalatiereden [23]. (Je moet meestal eerst de mogelijke redenen opvragen – zie volgende tool.)
- `topdesk_get_available_escalation_reasons` – Haalt alle beschikbare *escalatie*-redenen op die in TOPdesk zijn gedefinieerd [24]. Deze lijst van redenen kan gebruikt worden bij het escaleren van een incident.
- `topdesk_get_available_deescalation_reasons` – Haalt alle beschikbare *de-escalatie*-redenen op voor incidenten [25]. (Bij het terugdraaien van een escalatie kan een reden worden meegegeven.)
- `topdesk_deescalate_incident` – Draagt de escalatie van een incident weer af (*de-escalate*), eventueel met een mee te geven de-escalatiereden-ID [26].
- `topdesk_get_progress_trail` – Haalt het *progress trail* (het volledige logboek van statuswijzigingen en opmerkingen) van een incident op [27]. Optioneel kunnen hierin inline afbeeldingen meegenomen worden of juist als aparte data (base64) worden toegevoegd, afhankelijk

van parameters in de implementatie.
- `topdesk_get_incident_attachments` – Downloadt alle bijlagen van een incident en retourneert deze als base64-gecodeerde data [28]. Zo kun je de files (PDF's, afbeeldingen, etc.) ophalen die bij een incident horen.
- `topdesk_get_incident_attachments_as_markdown` – Haalt ook alle bijlagen op, maar probeert deze te converteren naar leesbare *Markdown*-tekst [29]. Dit is handig voor bijvoorbeeld PDF-rapporten of Word-documenten; de MCP-server probeert via AI (OpenAI API of Docling) de inhoud uit de bijlagen te extraheren en als tekst terug te geven (valt terug op een simpele converter als AI niet lukt).
- `topdesk_get_complete_incident_overview` – Geeft een *volledig overzicht* van een incident terug, inclusief de basisgegevens, alle progress-trail items *en* de inhoud van alle bijlagen (als markdown) [30] [31]. Deze tool combineert feitelijk de resultaten van meerdere andere tools – het is een alles-in-één voor als je een incident in zijn geheel wilt bekijken zonder apart de bijlagen en voortgang op te vragen.
- `topdesk_add_action_to_incident` – Voegt een nieuwe *actie* (bijv. een reactie of opmerking) toe aan een incident [32]. De inhoud (`text`) moet in HTML-formaat aangeleverd worden, conform de TOPdesk API (die HTML gebruikt voor opmaak in reacties). Hiermee kan bijvoorbeeld ChatGPT (met toestemming) een reply toevoegen aan een ticket.
- `topdesk_get_incident_actions` – Haalt alle acties (reacties/commentaar) op die aan een incident verbonden zijn [32]. Je krijgt een lijst van alle opmerkingen, inclusief hun ID's, auteurs, inhoud, etc.
- `topdesk_delete_incident_action` – Verwijdert een specifieke actie/reactie van een incident op basis van incident-ID en actie-ID [33]. Hiermee kun je bijvoorbeeld een foutieve opmerking weghalen.

Operator-gerelateerde tools: (Operators zijn de gebruikers binnen TOPdesk die tickets afhandelen, vaak servicedesk-medewerkers.) - `topdesk_get_operatorgroups_of_operator` – Geeft de lijst van operatorgroepen terug waarvan een bepaalde operator lid is [34]. Hiermee kun je bijvoorbeeld zien tot welke teams of afdelingen een medewerker behoort. (Je kunt ook een FIQL-query meegeven om te filteren op groepsnaam, of leeg laten om alle groepen op te vragen.)
- `topdesk_get_operator` – Haalt één TOPdesk-operator (medewerker) op aan de hand van zijn/haar unieke ID [35]. Je krijgt de details van de medewerker terug (naam, contactinfo, etc., voor zover beschikbaar via de API).
- `topdesk_get_operators_by_fiql_query` – Zoekt naar operators (medewerkers) via een FIQL-query en geeft de overeenkomende resultaten terug [36]. Bijvoorbeeld, je zou kunnen zoeken naar alle operators met een bepaalde rol of in een bepaalde locatie.

Personen (eindgebruiker) tools: (Personen zijn vaak de klanten/melders in TOPdesk.) - `topdesk_get_person_by_query` – Zoekt personen (eindgebruikers) via een query (FIQL) en retourneert de lijst van resultaten [37]. Dit kan gebruikt worden om bijvoorbeeld een gebruiker op naam of email te zoeken.
- `topdesk_get_person` – Haalt de gegevens van een specifieke persoon op ID op [38]. Je krijgt dan de persoonskaart-informatie (zoals naam, afdeling, contactgegevens) van die eindgebruiker.
- `topdesk_create_person` – Maakt een nieuw persoon-record aan in TOPdesk met de meegegeven velden [39]. Hiermee kun je via ChatGPT bijvoorbeeld direct een nieuwe gebruiker invoeren (als die nog niet bestaat) door benodigde info te geven.
- `topdesk_update_person` – Werkt een bestaand personenrecord bij (geïdentificeerd door ID) met de verstrekte gewijzigde velden [40]. Zo kun je bv. iemands telefoonnummer of afdeling updaten via de connector.
- `topdesk_archive_person` – Archiveert een persoon (bijv. wanneer iemand uit dienst gaat) op basis van ID [41]. In TOPdesk worden personen vaak gearchiveerd i.p.v. verwijderd. Deze tool voert dat uit; sommige TOPdesk-instellingen vereisen mogelijk een reden, die kan meegegeven worden (reason_id).

- `topdesk_unarchive_person` – Maakt een gearchiveerde persoon weer actief (de-archiveert) [42]. Dit kan bijvoorbeeld als iemand terug in dienst komt of per abuis gearchiveerd was.

(Noot: Bovenstaande opsomming is gebaseerd op de projectdocumentatie. "FIQL" verwijst naar Flexible Interview Query Language, een filtertaal die TOPdesk API gebruikt voor zoekopdrachten. Veel van deze tools roepen onder water direct een TOPdesk REST API endpoint aan. Zo kiest `topdesk_get_incident` *intern of het* `/incidents/id/{id}` *of* `/incidents/number/{nr}` *endpoint moet gebruiken op basis van het patroon van de invoer [43].)*

Foutdiagnose: *"Bad Request"* bij gebruik via ChatGPT

Situatie: In de vraag werd vermeld dat de Topdesk MCP testpagina (`/test`) prima functioneert – via die weg konden TOPdesk-acties uitgevoerd worden – maar dat vanuit een ChatGPT-prompt met de MCP als custom connector er "niets" werd teruggekregen. In de serverlogs op Render verscheen de foutmelding:

```
topdesk_mcp._utils - ERROR - Bad Request: The request was invalid or cannot
be served
```

Deze melding komt rechtstreeks uit de MCP-server code. De `_utils.py` module van de Topdesk MCP controleert de HTTP-status van elke API-respons. Bij status 400 (Bad Request) logt hij expliciet de boodschap *"Bad Request: The request was invalid or cannot be served."* [44]. Dit betekent dat de TOPdesk API de aanvraag niet kon verwerken omdat er iets mis was met het verzoek (bijv. ontbrekende of onjuiste parameters).

Analyse van oorzaken: Enkele potentiële redenen voor deze 400 *Bad Request* in context van de MCP-connector: - *Ongeldige of ontbrekende input:* De meest waarschijnlijke oorzaak is dat ChatGPT een tool aanriep zonder de juiste argumenten. Bijvoorbeeld, als `topdesk_get_incident` werd aangeroepen zonder een incidentnummer of ID (of met een verkeerd formaat), zou de MCP-server een URL construeren die eindigt op `/incidents/number/` zonder geldig nummer, wat door TOPdesk wordt afgewezen als een ongeldig verzoek (400). In de implementatie zien we dat als de meegegeven `incident` geen geldige UUID is, er van uitgegaan wordt dat het een *nummer* is en die waarde in de URL geplakt wordt [43]. Een lege string of een onvolledig nummer resulteert dan in een foutieve API-URL. Kortom, **een ontbrekende of verkeerd geformatteerde parameter (zoals incident-ID)** leidt vrijwel zeker tot een *Bad Request* vanuit TOPdesk.
- *Foutief gebruik van de functie:* Mogelijk heeft het model een verkeerde tool gekozen of het verkeerde veld gebruikt. Stel dat de gebruiker in ChatGPT iets vraagt als "zoek incident 12345", zou het model de `search` tool kunnen gebruiken die zoekt op titels. Als "12345" geen onderdeel van titels is, of als het model de output daarvan niet goed interpreteert en vervolgens *fetch* aanroept zonder geldige ID, kan dat misgaan. Een ander voorbeeld: het aanroepen van `topdesk_create_incident` zonder verplichte velden (zoals `caller_id`) zou ook een 400 kunnen opleveren vanuit TOPdesk (omdat de API het verzoek afkeurt als incompleet).
- *Randgevallen in de data:* Hoewel minder waarschijnlijk hier, zou het kunnen dat bepaalde tekens of encoderingen in de input een rol spelen. De `search` tool bijvoorbeeld bouwt een FIQL-query string en escape't quotes in de titel [4]. Als de gebruiker een vreemd karakter invoerde dat niet goed verwerkt werd, kán dat tot een ongeldige query geleid hebben. De implementatie lijkt hier echter rekening mee te houden (quotes worden ge-escaped en whitespace genormaliseerd), dus dit is minder waarschijnlijk dan een ontbrekende parameter.

Waarom geen response in ChatGPT UI: Wanneer de MCP-tool een error-string retourneert (zoals “Bad Request: …”), moet het model beslissen hoe dit aan de gebruiker te presenteren. Mogelijk “verstond” ChatGPT dat dit een fout was die niet direct bruikbaar is voor de gebruiker, en heeft het ervoor gekozen geen antwoord te formuleren (of slechts iets zeer algemeens). In sommige gevallen zal het model alsnog iets zeggen als “Er ging iets mis bij het ophalen van de gegevens.”, maar als het volledig stilviel kan dat komen doordat het twijfelde hoe te reageren op de fout.

Oplossingen/aanbevelingen: 1. **Zorg voor geldige input in de prompt:** Controleer bij gebruik van de connector of je ChatGPT alle benodigde info geeft. Bijvoorbeeld, gebruik een volledige incidentreferentie: *“Haal incident I-20230123-0001 op”* in plaats van alleen *“Haal incident 123 op”*. Dit vergroot de kans dat het model de juiste tool (`fetch` / `get_incident`) met correcte parameter aanroept. Als het incidentnummer niet bekend is, kun je eerst de `search` tool expliciet laten gebruiken: *“Zoek naar incidenten met ‘XYZ’ in de titel.”*
2. **Let op volgorde bij complexe acties:** Voor acties die meerdere gegevens vereisen (zoals incident aanmaken of tijd schrijven) kun je het gesprek stap voor stap voeren. Vraag eerst om een voorbeeld van de vereiste velden of controleer of het model de structuur weet. Je kunt ook de **FIQL how-to** tool benutten om te kijken hoe je filters kunt opstellen.
3. **Verbetering aan de MCP-kant:** Als je zelf controle hebt over de MCP-server, kun je overwegen extra validatie in te bouwen. De Topdesk MCP heeft bijvoorbeeld al een check in de `fetch` tool: die gooit een ValueError als het incident_id leeg is [5]. Zo’n check ontbreekt in `topdesk_get_incident` – je zou vergelijkbare inputcontrole kunnen toevoegen om een lege input af te vangen vóórdat de API wordt aangeroepen. Je zou dan een duidelijke foutmelding terug kunnen geven (die het model eventueel kan tonen). Ook logging van *welke URL* is aangeroepen (in debug mode) kan helpen om de oorzaak te vinden.
4. **Controleer de logs en reproduceer indien mogelijk:** De foutmelding geeft aan dát er iets mis ging, maar niet meteen wát. Probeer hetzelfde verzoek dat je ChatGPT gaf, na te bootsen via de `/test` pagina of een andere client. Zie je daar ook een fout of krijg je meer informatie? TOPdesk API’s geven bij 400 vaak een response body met meer detail. In `_utils.handle_topdesk_response` wordt geprobeerd error details uit JSON te halen, maar bij een kale 400 zonder JSON pakt hij alleen de status en tekst [44] [45]. Mogelijk kun je in de testpagina zien welke call misging. Bijvoorbeeld, als het een *create* betrof, kreeg je wellicht te horen welke veld ontbrak. Die kennis kun je dan weer in de prompt aan ChatGPT geven.

Samengevat duidt de *“Bad Request”* fout erop dat de **MCP-connector zelf technisch goed bereikbaar is**, maar dat **de inhoud van een verzoek onjuist was voor de TOPdesk API**. Door de invoer/aanroep te corrigeren (ofwel via betere instructies aan ChatGPT, of via verbeterde validatie in de MCP-server), zou het probleem verholpen moeten worden. Zodra ChatGPT de functies met geldige parameters aanroept, zal de Topdesk MCP naar verwachting correcte resultaten teruggeven – net zoals je al op de testpagina zag. Met andere woorden: de “USB-C-verbinding” (MCP) staat, je moet alleen even zorgen dat de juiste stekker op de juiste manier wordt ingestoken .

Bronnen: De beschrijving van MCP is gebaseerd op OpenAI’s documentatie [1]. De lijst van Topdesk-tools en hun uitleg is samengevat uit de projectdocumentatie van de Topdesk MCP [14] [17]. De foutmelding en analyse zijn afgeleid van de MCP-server code en logoutput [44], alsmede de TOPdesk API-verwachtingen. Zie de aangehaalde bronnen voor meer details.

---

[1] Model context protocol (MCP) - OpenAI Agents SDK
https://openai.github.io/openai-agents-python/mcp/

2 3 Thread by @OpenAI on Thread Reader App – Thread Reader App
https://threadreaderapp.com/thread/1968361701784568200.html

4 5 9 12 13 TOPdesk MCP Server | Glama
https://glama.ai/mcp/servers/@GerbenRoebersen/Topdesk_MCP_python/blob/6bd14dece60ae42171ea97a84055e02c1fd61844/topdesk_mcp/main.py

6 7 8 10 11 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 topdesk-mcp · PyPI
https://pypi.org/project/topdesk-mcp/

43 TOPdesk MCP Server | Glama
https://glama.ai/mcp/servers/@GerbenRoebersen/Topdesk_MCP_python/blob/6bd14dece60ae42171ea97a84055e02c1fd61844/topdesk_mcp/_incident.py

44 45 TOPdesk MCP Server | Glama
https://glama.ai/mcp/servers/@GerbenRoebersen/Topdesk_MCP_python/blob/6bd14dece60ae42171ea97a84055e02c1fd61844/topdesk_mcp/_utils.py