Sisukord

AI solution Ask Directo: just ask, and the AI gives you answers based on your Directo data 3
Directo MCP Server Setup - Claude Desktop and Gemini CLI 3
1. Introduction - why MCP? 3
What is MCP? 3
What's the benefit? 3
Architecture 4
2. Checking prerequisites 4
2.1 Required for everyone 5
2.2 Additionally for Claude Desktop users 5
2.3 Additionally for Gemini CLI users 5
2.4 Admin rights — when are they needed? 6
3. Shared steps (do once) 6
Step 3.1 — Installing Node.js (required) 6
Choose your path 7
PATH A — Quick install with scripts (recommended) 7
PATH B — Manual, step by step (alternative to Path A) 9
Step B1 — Setting the PowerShell execution policy 9
Step B2 — CredentialManager module 10
Step B3 — Creating the MCP server folder 10
Step B4 — Placing directo-mcp.js 11
Step B5 — Creating package.json 11
Step B6 — Installing dependencies 12
Step B7 — Securely saving the API key 12
4. Setting up Claude Desktop 13
Step 4.1 — Configuration file location 13
Step 4.2 — Configuration file 14
Step 4.3 — Restarting Claude Desktop 15
Step 4.4 — Testing 15
5. Setting up Gemini CLI 16
Step 5.1 — Installing Gemini CLI 16
Step 5.2 — Setting up the Google API key 17
Step 5.3 — Configuration file location 17
Step 5.4 — Adding the MCP server 18
Step 5.5 — Trusting the folder (Trusted Folders) 18
Step 5.6 — Testing 19
6. Troubleshooting 20
Common errors (either client) 20
Gemini CLI-specific errors 21
7. MCP tools 21
Good to know 22
8. Security 22
9. Updating and versions 23
Which version am I running? 23
How to update 23
10. References 23

AI solution Ask Directo: just ask, and the AI gives you answers based on your Directo data

The Ask Directo solution is built on an MCP (Model Context Protocol) server, which, simply put, is a small local technical helper program running on your computer. By installing the MCP on your computer, you allow an AI assistant (Claude, Gemini) to securely read data directly from your Directo system. All you need to do is ask your questions in plain language.

Read more about what Ask Directo can do: https://directo.ee/ask-directo.

Directo MCP Server Setup - Claude Desktop and Gemini CLI

This guide explains how to connect the Directo ERP system to an AI assistant (Claude or Gemini) through a local MCP (Model Context Protocol) server. The guide is intended for both Claude Desktop and Gemini CLI users - the shared steps only need to be done once, after which you pick the section for your AI client.

1. Introduction - why MCP?

What is MCP?

MCP (Model Context Protocol) is an open standard that lets AI assistants communicate directly with external systems — databases, APIs (Application Programming Interfaces), and applications.

Without MCP, working with an AI assistant looks like this:

1. You copy data out of Directo
2. You paste it into the AI conversation
3. The AI answers
4. You copy the answer back into Directo

With MCP, it looks like this:

1. You ask in natural language: „Which items are below minimum stock level?“
2. The AI queries the Directo API (Application Programming Interface) itself
3. The AI analyzes the data and answers immediately

What's the benefit?

Situation	Without MCP	With MCP
Stock level check	Open Directo → search → copy → paste into AI	„Which items have fewer than 10 units in stock?“

Situation	Without MCP	With MCP
Invoice analysis	Export CSV → upload to AI → ask	„Which customers haven't paid in the last 90 days?“
Sales report	Build manually → copy → analyze	„Compare this month's sales to last year“
Order tracking	Open several views → copy info	„Which orders are more than 5 days overdue?“

Architecture

```
You (natural language)
        ↕
AI client (Claude Desktop or Gemini CLI)
        ↕  MCP protocol (local)
MCP Server (on your computer, Node.js)
        ↕  HTTPS
Directo API (cloud)
```

The MCP server runs **on your computer**. The Directo API key never leaves your machine. The data retrieved from Directo does, however, travel onward to the AI provider's cloud so the model can read it and formulate a natural-language answer - the same as any other message content in the conversation.



Local install — important to understand:

- The MCP server is installed **only on your own computer**
- It is **not shared** — colleagues cannot access your MCP server
- Every user must do all the steps **themselves**, on their own computer
- The MCP server only works while **your computer is on** and the AI client is running
- If you want to share an MCP server across multiple users, you need a separate hosted-server solution (out of scope for this guide)
- One API key belongs to **one Directo database**. This MCP server is tied to exactly the database the key was created in and cannot query another database (for example between a production and a test database). The AI itself does not know which database the server is connected to: if it claims to be querying another database, that is not true. The Directo address bar shows the database name (`login.directo.ee/DATABASE/…`): make sure the API key was created in the right database.

2. Checking prerequisites

Check every prerequisite before installing. A missing prerequisite is the most common cause of failure.

2.1 Required for everyone

Prerequisite	How to check	Fix
Windows 10 or 11 (64-bit)	Start → Settings → System → About	—
Node.js 22 or newer (LTS)	`node –version`	nodejs.org → LTS
npm	`npm –version` (only works after execution policy is set, see below)	Comes bundled with Node.js
PowerShell execution policy	`Get-ExecutionPolicy` → should return RemoteSigned	See below
CredentialManager module	`Get-Module -ListAvailable CredentialManager`	See below
Directo API key	Directo → employee card → Settings → Integration (visible to users with SUPER rights)	If your company has no key yet, ask your Directo project manager — only they can create it

2.2 Additionally for Claude Desktop users

Prerequisite	How to check	Link
Claude Desktop installed	Claude appears in the Start menu	https://claude.ai/download
Anthropic account	Sign-in works	https://claude.ai

Official documentation: https://docs.claude.ai

2.3 Additionally for Gemini CLI users



⚠ **Gemini CLI requires a paid Gemini API key.** Since June 18, 2026 Google no longer lets you sign in to Gemini CLI with a Google account (free, Google AI Pro and Google AI Ultra users were moved to Antigravity CLI). New Gemini Code Assist Standard / Enterprise subscriptions are no longer sold; an existing licence still works until its contract ends.

Use a **Gemini API key from Google AI Studio with billing enabled** (paid tier). A free (unpaid) API key may still work with limits, but Google no longer guarantees it for Gemini CLI — we do not recommend it for Ask Directo. This guide does not cover Antigravity CLI.

Prerequisite	How to check	Link
Google account	Sign-in works	https://accounts.google.com
Paid Gemini API key (billing enabled)	Google AI Studio → API keys → the key's project shows a paid tier	https://aistudio.google.com/apikey
Gemini CLI installed	`gemini –version`	`npm install -g @google/gemini-cli`

Official documentation: https://geminicli.com/docs

2.4 Admin rights — when are they needed?



Good news: almost every step can be done **without** administrator rights. You genuinely need admin rights at only one point — installing Node.js.

Step	Admin rights?	Reason
Installing Node.js	**Yes** — the installer requires it	System-level install (Program Files)
PowerShell execution policy	No	`-Scope CurrentUser` writes to the HKCU registry
CredentialManager module	No	`-Scope CurrentUser` installs under the user's profile
npm install (MCP server)	No	User-level install
Claude Desktop config	No	User folders
Gemini CLI config	No	User folders
Saving the API key	No	Credential Manager, user-level entry

Risks and an important warning:

- Node.js installer — low risk, official software
- **Do not open any of the other steps as administrator**, even if an earlier version of this guide or examples found online suggest it — `-Scope CurrentUser` exists specifically to avoid needing admin rights
- If your everyday account and your admin rights come from **separate Windows accounts** (common in corporate/domain environments), opening something as administrator runs the action under the *other* account's profile — the module, setting, or key then „disappears“ from your regular user's point of view. See the troubleshooting chapter for details.
- Don't run `npm install` as administrator — this creates permission conflicts
- `npm install -g` (e.g. Gemini CLI, Step 5.1) only needs admin rights if npm's global prefix points at the `Program Files` folder — Step 5.1 includes a fix that avoids this

3. Shared steps (do once)



You do these steps **on your own computer**. If a colleague wants to use the same setup, they need to do these steps **on their own computer** independently.

Step 3.1 — Installing Node.js (required)



This is the only step that is **required on both paths** below and cannot be automated (it needs admin rights). Do this before moving on, regardless of which path you choose.

1. Go to https://nodejs.org and download the **LTS** version (22 or newer)
2. Run the downloaded `.msi` file as administrator
3. Install with the default settings — don't change the install location

Verify the install (regular PowerShell, not admin):

```
node --version
```

Expected result:

```
v22.11.0 or newer
```



We deliberately don't check `npm –version` here yet — npm is a PowerShell script (`npm.ps1`) and will give a `running scripts is disabled` error before the execution policy is set (Path A script, or Step B1), even if Node.js and npm themselves are installed correctly. This is not a bug, it's expected behavior at this stage. `npm –version` will work after completing Path A/B.



If PowerShell says „not recognized“ — close the window and reopen it. Node.js only adds itself to PATH in new PowerShell sessions.

Choose your path



Below are **two alternative paths** to reach the same result — a working MCP server, with the API key and AI client set up. **Do only ONE of them, not both.**

	Path A — Scripts	Path B — Manual
Time	~5 minutes	~20 minutes
Best for	Most users	You want to understand every step yourself, or Path A doesn't work in your environment
Covers steps	A1—A2	B1—B7

If you choose Path A and it succeeds, **skip Path B (B1—B7) entirely** — those steps describe, manually, exactly what the scripts already did for you. Path B is troubleshooting reference material, not an additional required step.

PATH A — Quick install with scripts (recommended)

Step A1 — Downloading the files

Download `directo-mcp-files.zip` and extract it:

directo-mcp-files.zip



The files are attached to the wiki as a **ZIP archive**, not as individual links — DokuWiki blocks `.js` and `.ps1` file uploads by default (for security, see `conf/mime.conf`), but `.zip` is allowed by default. Don't rename the files to a different extension (e.g. `.txt`) as a „fix“ — the wrong extension is exactly what causes `MODULE_NOT_FOUND`-type errors later.



The scripts print all messages in English: `[OK]` = succeeded, `[!]` = needs your attention, `[ERROR]` = failed.



The scripts are updated from time to time. If you already downloaded the ZIP earlier (for example an older copy sits in your Downloads folder), **always download a fresh version**. When you run a script, its first line shows a version number (for example `v1.5`). If you don't see a version number, you have an old copy.

Extract it (right-click the ZIP → **Extract All**, or in PowerShell):

```
Expand-Archive "$env:USERPROFILE\Downloads\directo-mcp-files.zip" -
DestinationPath "$env:USERPROFILE\Downloads\directo-mcp-files"
cd "$env:USERPROFILE\Downloads\directo-mcp-files"
```

Before running the scripts, two Windows default security restrictions must be cleared — both are required, not optional:

1) Allow running scripts in PowerShell at all:

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```



Without this, every following script will fail with `running scripts is disabled on this system` / `UnauthorizedAccess`. `-Scope CurrentUser` does not require admin rights.

2) Remove Windows' „downloaded from the internet“ (Mark of the Web) flag:

```
Get-ChildItem "$env:USERPROFILE\Downloads\directo-mcp-files" -Recurse |
Unblock-File
```



This is also required, not optional — files downloaded through a browser are blocked by default **even if the execution policy is already correct**. Windows tags internet-downloaded files with a separate security marker, which also carries over to files extracted from a ZIP archive — `RemoteSigned` requires a digital signature for such files, which our scripts don't have. Without this step you'll see `is not digitally signed` / `UnauthorizedAccess`, which looks confusingly similar to the error from step 1, but has a different cause and fix.

The archive contains four files: `directo-mcp.js` (the MCP server itself), `set-directo-apikey.ps1`, `install-claude-mcp.ps1`, `install-gemini-mcp.ps1`.

Step A2 — Running the scripts

In the same PowerShell window (not as administrator), in the folder you just extracted:
A2.1 API key (shared by both clients, do this first)

```
.\set-directo-apikey.ps1
```

A2.2 Pick your AI client (or both):
Claude

```
.\install-claude-mcp.ps1
```

Gemini

```
.\install-gemini-mcp.ps1
```

Each script prints a summary at the end: `[OK]` = done, `[!]` = still needs your action (for example restart Claude Desktop completely), `[ERROR]` = failed, with a message saying what to do.



If the scripts succeeded, you're done with Chapter 3. Go straight to Chapter 4.3 (Claude Desktop) and/or Chapter 5 (Gemini CLI) — Path B (below) is not needed.

PATH B — Manual, step by step (alternative to Path A)



This is an **alternative to Path A**, not an addition to it. Only do this if the Path A scripts didn't work, or if you want to go through every step yourself. If Path A already succeeded, skip this section.

Step B1 — Setting the PowerShell execution policy

Check the current setting:

```
Get-ExecutionPolicy
```

If the answer isn't `RemoteSigned`, fix it in a **regular** PowerShell window (not as administrator):

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```



`-Scope CurrentUser` writes the setting only to the HKEY_CURRENT_USER registry hive, not a machine-wide location — that's why it doesn't need admin rights. Admin

rights would only be needed for `-Scope LocalMachine`, which this guide does not use.

It will ask for confirmation → press Y → Enter.

Step B2 — CredentialManager module

Open a **regular** PowerShell window (not as administrator) — under exactly the same user account the MCP server will later run under:

```
Install-Module -Name CredentialManager -Force -Scope CurrentUser
```

`-Scope CurrentUser` exists specifically to avoid needing admin rights — the module installs under your own profile (`Documents\WindowsPowerShell\Modules`). If the command asks to confirm the NuGet provider or an „untrusted repository," answer Y — these steps don't need admin rights either.

Verify the install succeeded (in the same regular PowerShell window):

```
Get-Module -ListAvailable CredentialManager
```

Should show the module's name and version.

⚠ If your everyday work account and your admin rights come from **separate Windows accounts** (common in corporate/domain setups), make sure this step and the later Step B7 run under **the same account** that `node directo-mcp.js` will later run under. A PowerShell window opened with elevation may actually run under a different account's identity, which would leave the module or API key under the wrong profile, and the later step will fail invisibly.

Step B3 — Creating the MCP server folder

Open a regular PowerShell window (not admin) and run:

```
New-Item -ItemType Directory -Force -Path "$env:USERPROFILE\directo-mcp"
```

Always use the `$env:USERPROFILE` variable, rather than manually typing `C:\Users\YOURNAME`. On domain-joined computers (corporate/AD environments) your actual profile folder may differ from your sign-in name — for example it may include a domain suffix (`YOURNAME.DOMAIN`). `$env:USERPROFILE` always gives the real, correct path, regardless.

Step B4 — Placing directo-mcp.js



This file contains all 41 tools (40 Directo API interfaces and the access-overview tool `get_api_access`), and each tool's parameters have been checked against Directo's actual specifications. Don't copy this file manually from Notepad or any other source — always use a fresh, official copy to avoid ending up with an outdated or incorrectly-parameterized version.

Download the `directo-mcp.js` file — it's included in the `directo-mcp-files.zip` archive linked under Path A (extract it, see Step A1) — and place it in this folder:

```
$env:USERPROFILE\directo-mcp\directo-mcp.js
```

The simplest way: if the file is already in the extracted Downloads subfolder, copy it in PowerShell:

```
Copy-Item "$env:USERPROFILE\Downloads\directo-mcp-files\directo-mcp.js"
"$env:USERPROFILE\directo-mcp\directo-mcp.js"
```



Make sure the file name is exactly `directo-mcp.js` — not `directo-mcp.js.txt`. Check:

```
Test-Path "$env:USERPROFILE\directo-mcp\directo-mcp.js"
```

Should return `True`. If `False`, check the exact file name:

```
Get-ChildItem "$env:USERPROFILE\directo-mcp"
```

Step B5 — Creating package.json

Open a new Notepad window and paste:

```
{
  "name": "directo-mcp",
  "version": "1.0.0",
  "type": "module",
  "description": "Directo ERP MCP Server",
  "main": "directo-mcp.js",
  "scripts": {
    "start": "node directo-mcp.js"
  },
  "dependencies": {
    "@modelcontextprotocol/sdk": "^1.0.0",
    "zod": "^3.22.0"
  }
```

```
}
```

Save it in the same folder, named `package.json` (file type: All Files).

Step B6 — Installing dependencies

Regular PowerShell (not admin):

```
cd "$env:USERPROFILE\directo-mcp"
npm install
```

A successful install creates a `node_modules` folder and a `package-lock.json` file in this folder.



This is a **local** npm install (goes under the `directo-mcp` folder's own `node_modules`), not global — it never needs admin rights or any special npm-prefix configuration, regardless of how Node.js was originally installed.

Step B7 — Securely saving the API key

The Directo API key is saved to Windows Credential Manager — encrypted, under your user account only. No one else on the computer can see this key.

```
New-StoredCredential -Target DIRECTO_API_KEY -UserName directo -Password
YOUR_REAL_KEY_HERE -Persist LocalMachine
```

Verify the save succeeded:

```
(Get-StoredCredential -Target
DIRECTO_API_KEY).GetNetworkCredential().Password
```

Should display your API key.

Rotating the API key in the future:

```
Remove-StoredCredential -Target DIRECTO_API_KEY
New-StoredCredential -Target DIRECTO_API_KEY -UserName directo -Password
NEW_KEY -Persist LocalMachine
```



Run this step under **exactly the same user account** that `node directo-mcp.js` will later run under. See the warning under Step B2 about the separate-admin-account risk.



The Path A script (`set-directo-apikey.ps1`) does the same thing more securely (asks for the key without echoing it to the screen) and also works for rotation — use it

if you can.

4. Setting up Claude Desktop

First complete Chapter 3 (Step 3.1 + either Path A or Path B). Setting up Claude Desktop assumes the MCP server already exists.

The MCP server only works **on your own computer** and only inside the **Claude Desktop application** — not in the claude.ai web version, and not on other users' computers.

Recommended: if you ran the `install-claude-mcp.ps1` script under Path A (Chapter 3), Steps 4.1—4.2 are already done — the script detects, patches, and backs up the configuration file automatically. Go straight to Step 4.3.

Step 4.1 — Configuration file location

Claude Desktop's configuration file `claude_desktop_config.json` is in one of four places on Windows, **depending on the version and how it was installed**. The Path A script finds the right file itself. For the manual path (Path B), find it with the steps below.

Variant	Location
MSIX (claude.ai/download), data moved (Claude Desktop 3.1623 and later)	`%LOCALAPPDATA%\Claude-Data\claude_pzs8sxrjxfjjc\`
The same MSIX before the data move	`%LOCALAPPDATA%\Packages\Claude_pzs8sxrjxfjjc\LocalCache\Roaming\Claude\`
Microsoft Store build (not verified)	`%LOCALAPPDATA%\Claude-Data\anthropicpbc.claude_fnn82j28hfe8t\`
Earlier `.exe` installer	`%APPDATA%\Claude\`

Newer Claude Desktop versions have moved their data from the old package folder to the `Claude-Data` folder. The old folder keeps only a `.user-data-left…` marker (on some machines also a stale `claude_desktop_config.json`). **A file that Claude no longer reads can be edited without any error, but the change never reaches the app**, and the MCP server silently fails to load. Always use the new folder when it exists.

Show which of these files exist on your machine (regular PowerShell):

```
"$env:LOCALAPPDATA\Claude-
Data\claude_pzs8sxrjxfjjc\claude_desktop_config.json",
"$env:LOCALAPPDATA\Packages\Claude_pzs8sxrjxfjjc\LocalCache\Roaming\Claude\c
laude_desktop_config.json",
"$env:LOCALAPPDATA\Claude-
Data\anthropicpbc.claude_fnn82j28hfe8t\claude_desktop_config.json",
"$env:APPDATA\Claude\claude_desktop_config.json" | Where-Object { Test-Path
$_ }
```

Use the **first** file in the list (the order is the same as in the table: newer before older). Exceptions:

- If the list contains a `Packages\…` file and the same folder also contains a `.user-data-left…` file, that file is stale — do not use it.
- If the folder `%LOCALAPPDATA%\Claude-Data\claude_pzs8sxrjxfjjc` exists but has no `claude_desktop_config.json`: open Claude Desktop → Settings → Developer → Edit Config (this creates the file) and run the command again.
- If the list is empty: run Claude Desktop at least once and run the command again.

Step 4.2 — Configuration file

Create a file named `claude_desktop_config.json` in the (correct, as identified in Step 4.1) folder.



When creating the file with Notepad: File → Save As → file type **All Files (*.*)** → name: `claude_desktop_config.json`

File contents:

```
{
  "mcpServers": {
    "directo": {
      "command": "node",
      "args": [
        "C:\\Users\\YOURNAME\\directo-mcp\\directo-mcp.js"
      ]
    }
  }
}
```



Replace `YOURNAME` with your **actual** profile folder name — not your assumed username. On domain-joined (AD) computers, the actual profile folder may differ from your sign-in name (e.g. include a domain suffix). Check the exact value:

```
$env:USERPROFILE
```

Use exactly what the command returns in the JSON (with double backslashes). Example for illustration — if `$env:USERPROFILE` returns `C:\Users\john.user.COMPANY`, the path in the JSON should be

`C:\\Users\\john.user.COMPANY\\directo-mcp\\directo-mcp.js`. Whether or not a domain suffix (`.COMPANY`) is present depends on your own computer's setup — always use your own command's actual output, not this example.

If the file already contains other content (Claude Desktop's own settings, e.g. a `preferences` block) — **do not overwrite it**. `mcpServers` must be at the **root level** of the file, at the same level as `preferences`, not inside it. If you're unsure about correct brace placement, use the Path A script instead, which edits the file as a structure (`ConvertFrom-Json`/`ConvertTo-Json`), not as raw text — this completely eliminates brace-placement errors.

The API key is **not** added to the configuration file — the server reads it automatically from Credential Manager (Path A script, or Step B7).

Step 4.3 — Restarting Claude Desktop

1. Right-click the Claude icon in the **system tray** (near the clock, in the taskbar) → **Quit** / **Exit**
2. Simply closing the window is not enough — Claude Desktop keeps running in the background
3. If Quit isn't available — open Task Manager → find the Claude process → End Task
4. Open Claude Desktop again and wait a few seconds

Step 4.4 — Testing

In a Claude Desktop conversation, type first:

> Which Directo interfaces are enabled on my API key?

Claude uses the `get_api_access` tool and shows the list of interfaces. This query returns no Directo business data, so it is the safest first test. Then try a data query, for example:

> Which warehouses do I have in Directo?

Claude uses a suitable tool (here `get_stocks`) and returns Directo data. Above the answer you see a gray label „Used directo integration“. If the interface is not enabled on your key, Claude tells you which interface must be enabled in Directo.

Claude Desktop asks for permission the first time. The first time a new tool is used, a confirmation question appears in the Claude Desktop window (for example Allow once / Allow always). Choose `Allow always` if you trust the tool. If there is no answer and a „No result received“ message appears after several minutes, check whether a confirmation window is open somewhere.

Make sure the answer is data from **your database**: ask for something you know from your own

Directo (for example a well-known customer or warehouse) and compare.

Checking the logs if something doesn't work:

In current Claude Desktop versions the logs are in `%LOCALAPPDATA%\Claude\logs`:

```
# Current Claude Desktop versions:
Get-Content "$env:LOCALAPPDATA\Claude\logs\mcp-server-directo.log" -Tail 30

# Earlier .exe installer:
Get-Content "$env:APPDATA\Claude\logs\mcp-server-directo.log" -Tail 30
```

Official Claude Desktop MCP documentation: modelcontextprotocol.io - Connect to local MCP servers



The official MCP documentation still gives the old paths (`%APPDATA%\Claude\…`) for Windows. Step 4.1 of this guide describes where Claude Desktop currently keeps its files.

5. Setting up Gemini CLI



First complete Chapter 3 (Step 3.1 + either Path A or Path B). Setting up Gemini CLI assumes the MCP server already exists.



The MCP server only works **on your own computer**, and only inside the terminal-based Gemini CLI application — not the Gemini web version, and not on other users' computers.



⚠ Before continuing, make sure you have valid authentication: a paid Gemini API key, a Gemini Code Assist Standard / Enterprise license, or a free API key that works with Gemini CLI. Signing in with a Google account (including Google AI Pro/Ultra) no longer works (see Step 2.3).



Recommended: if you ran the `install-gemini-mcp.ps1` script under Path A (Chapter 3), Steps 5.1 and 5.3—5.4 are already done. Only go through Step 5.2 (Google API key — the script doesn't do this, since it's separate Google, not Directo, authentication) and Steps 5.5—5.6.

Step 5.1 — Installing Gemini CLI



This step does not need admin rights — provided npm's global install location points

into your own profile. If not yet configured, do this once (regular PowerShell, not as administrator):

```
mkdir "$env:APPDATA\npm-global" -Force
npm config set prefix "$env:APPDATA\npm-global"
[Environment]::SetEnvironmentVariable("Path", $env:Path +
";$env:APPDATA\npm-global", "User")
```

Close PowerShell completely and open a **new** window (PATH only refreshes in a new session), then install Gemini CLI:

```
npm install -g @google/gemini-cli
```

Verify:

```
gemini --version
```

⚠ If the command still asks for admin rights, the likely cause is that npm's prefix was already set (e.g. by another piece of software) to point at `C:\Program Files\…`. Check the current setting with `npm config get prefix` — after the step above it should show `%APPDATA%\npm-global`, not a Program Files folder.

Step 5.2 — Setting up the Google API key

Go to https://aistudio.google.com/apikey, create an API key and **enable billing** for the key's project (Set up billing). Without billing the key runs on the free tier, which may not work with Gemini CLI.

Save the key as a persistent environment variable:

```
[System.Environment]::SetEnvironmentVariable("GEMINI_API_KEY",
"YOUR_GOOGLE_API_KEY", "User")
```

Close PowerShell and reopen it. Verify:

```
$env:GEMINI_API_KEY
```

Step 5.3 — Configuration file location

The Gemini CLI configuration is located on Windows at:

```
$env:USERPROFILE\.gemini\settings.json
```

Check whether the folder exists:

```
Test-Path "$env:USERPROFILE\.gemini"
```

If the answer is `False`, create the folder:

```
New-Item -ItemType Directory -Force -Path "$env:USERPROFILE\.gemini"
```

Step 5.4 — Adding the MCP server

Create or edit the file `$env:USERPROFILE\.gemini\settings.json`:

```
{
  "mcpServers": {
    "directo": {
      "command": "node",
      "args": [
        "C:\\Users\\YOURNAME\\directo-mcp\\directo-mcp.js"
      ],
      "env": {
        "DIRECTO_API_KEY": "$DIRECTO_API_KEY"
      }
    }
  }
}
```



Replace `YOURNAME` with your **actual** profile folder name — check the exact value with `$env:USERPROFILE` (see the warning under Step 4.2 on this same topic).



A note on the API key logic: `directo-mcp.js` reads the key from Credential Manager first (Path A script, or Step B7) — this works **exactly the same way for Gemini as for Claude Desktop**, since both run the same server file. The env block here is a **fallback**, not the only path — if Credential Manager is already set up, this block isn't strictly necessary, though it doesn't hurt either. If you do want to use it, save the value as an environment variable:

```
[System.Environment]::SetEnvironmentVariable("DIRECTO_API_KEY", "YOUR_DIRECTO_API_KEY", "User")
```

Gemini CLI supports environment-variable expansion in `env` values using `$VAR_NAME` syntax — that's why the JSON contains „`$DIRECTO_API_KEY`“, not the actual key.

Step 5.5 — Trusting the folder (Trusted Folders)

Gemini CLI has a security feature called **Trusted Folders**, which by default doesn't allow connecting MCP servers from an untrusted folder. If this feature is enabled, the folder needs to be trusted.

The simplest way — trust the server directly in `settings.json`, by adding `„trust“: true` to the `directo` block:

```
{
  "mcpServers": {
    "directo": {
      "command": "node",
      "args": [
        "C:\\Users\\YOURNAME\\directo-mcp\\directo-mcp.js"
      ],
      "env": {
        "DIRECTO_API_KEY": "$DIRECTO_API_KEY"
      },
      "trust": true
    }
  }
}
```



`„trust“: true` skips the tool-call confirmation dialogs. Only use this for servers you fully manage yourself — this Directo server is such a case.

Alternative — if Trusted Folders is enabled, Gemini CLI shows a trust dialog the first time it opens this folder. Choose **Trust folder** there. The choice is saved to `~/.gemini/trustedFolders.json` and is only asked once per folder. Use the `/permissions` command inside the CLI to change the trust level later.

Step 5.6 — Testing

Start Gemini CLI:

```
gemini
```

Check the MCP connection:

```
/mcp list
```

Should show:

```
 directo - Ready (41 tools)
```

Test query:

> Which Directo interfaces are enabled on my API key?

Official Gemini CLI MCP documentation: geminicli.com/docs/tools/mcp-server

6. Troubleshooting

Common errors (either client)

Error message	Cause	Fix
`DIRECTO_API_KEY is missing`	Credential Manager save failed, or the server runs under a different Windows account	Redo Step B7 (or Path A: `.\set-directo-apikey.ps1`) under the same account Claude runs under
`Unable to convert Credential object…` / `null-valued expression`	The key was saved with `cmdkey`, which is not always compatible with how the CredentialManager module reads it	Delete it with `Remove-StoredCredential -Target DIRECTO_API_KEY` and save it again with `New-StoredCredential` (Step B7) or with the Path A script
`Cannot find module`	npm install failed, OR directo-mcp.js is in the wrong folder/has the wrong name	Redo Step B6; check `Test-Path „$env:USERPROFILE\directo-mcp\directo-mcp.js“`
`SyntaxError`	directo-mcp.js file is corrupted or incomplete	Download the file again, see Step B4 (or Path A: A1)
`The „…“ interface is not enabled for this API key`	This interface is not enabled on your API key	Ask your Directo administrator to enable the interface on the key (API Direct). Ask „Which Directo interfaces are enabled on my API key?“ to see what the key allows
`Filter „…“ cannot be used … not selected for output`	This field is not selected for output on the key	Use another filter, or have the field added to the key's outputs
`Unknown filter name`	Wrong filter name (names are case-sensitive)	Ask which filters the interface allows: „Which filters does the … interface allow?“
`Filter „…“ has an invalid value syntax`	The filter value has the wrong format (a date must be ISO 8601, for example 2026-01-01T00:00:00)	Claude usually corrects it itself; if needed, ask it to use the right format
`The API key is missing or invalid (401)`	The Directo API key is wrong, expired or deleted	Check the key in Directo admin and save it again (`.\set-directo-apikey.ps1`)
`Access denied: the IP address of this computer is not allowed`	The key has an IP restriction and your computer's address is not allowed	Ask your Directo administrator to allow the IP
`Rate limit exceeded (429)`	Too many requests	Wait a moment and try again (recommended limit: up to 60 requests per minute)
`Directo reported a server error (5xx)`	A temporary problem on Directo's side	Try again later; if it persists, contact Directo support
`running scripts is disabled`	PowerShell execution policy	`Set-ExecutionPolicy RemoteSigned -Scope CurrentUser` (Step A1 or B1)
`is not digitally signed` / `UnauthorizedAccess`	Windows Mark-of-the-Web (file downloaded via a browser)	`Get-ChildItem <folder> -Recurse | Unblock-File` (see Step A1)

Error message	Cause	Fix
`npm not recognized`	Node.js is not on PATH	Close and reopen PowerShell



The server's error messages are in English. Claude explains them to you in your own language.

Problem	Fix
No hammer icon visible	Check the log file (see below)
MCP server doesn't start	Check that the config file's name doesn't have a `.txt` extension
Config folder is missing	Run Claude Desktop once — the folders are created automatically
Config edited, but nothing changes	Make sure you're editing the RIGHT file — see Step 4.1 (four possible locations; in current Claude Desktop the file is in `Claude-Data`, the old package folder is empty or stale)
No answer (after several minutes: „No result received“)	Claude Desktop is waiting for you to confirm the tool use: check whether a confirmation window is open in Claude Desktop, and choose Allow once / Allow always
MCP disappeared after a Claude Desktop update	Claude moved its data to the new folder (Step 4.1). Check that `claude_desktop_config.json` in the new folder contains the `directo-mcp.js` line (`Select-String -Path „<file>“ -Pattern „directo-mcp.js“`). If not, run the Path A script again.

Log file (current Claude Desktop versions; the second line is for the earlier .exe installer):

```
Get-Content "$env:LOCALAPPDATA\Claude\logs\mcp-server-directo.log" -Tail 30
Get-Content "$env:APPDATA\Claude\logs\mcp-server-directo.log" -Tail 30
```

Gemini CLI-specific errors

Problem	Fix
`/mcp list` shows Disconnected	Add `„trust“: true` to the server config (Step 5.4), or trust the folder via the dialog / the `/permissions` command
`GEMINI_API_KEY` error	Redo Step 5.2, open a new PowerShell window
Rate limit / 429 error	The key is on the free tier or has reached its limit — enable billing for the key's project in Google AI Studio (see Step 5.2)
Gemini CLI won't sign in with a Google account	Google-account sign-in (incl. Google AI Pro/Ultra) ended on June 18, 2026: use a paid Gemini API key (see Step 2.3)

Gemini CLI diagnostics:

```
/mcp list
```

7. MCP tools

Both clients (Claude Desktop and Gemini CLI) use the same 41 tools (40 Directo API interfaces and the access-overview tool `get_api_access`):

Group	Tools
Sales	get_invoices, get_orders, get_sales_quotations, get_sales_contracts, get_receipts
Customers	get_customers, get_suppliers, get_contacts
Items / stock	get_items, get_itemclasses, get_stocklevels_sn, get_stocklevels, get_allocations, get_priceformulas, get_stocks
Purchasing	get_purchase_orders, get_purchase_invoices, get_purchase_payments, get_stock_receipts
Accounting	get_transactions, get_accounts, get_financial_budgets, get_nettings
Logistics	get_deliveries, get_movements
People / HR	get_users, get_absences, get_wage_calculations, get_expenses, get_approvals
Assets	get_assets, get_customer_assets
Other	get_projects, get_objects, get_resources, get_events, get_recipes, get_replacementlogs, get_usagelogs
Sync	get_deleted
Access overview	get_api_access

Good to know

- **A tool only works if its interface is enabled on your API key.** Your Directo administrator chooses which interfaces and which fields the key allows. If an interface is missing, the error message says which interface must be enabled in Directo.
- **Access overview:** ask „Which Directo interfaces are enabled on my API key?“. The result is saved for 1 day (file `discovery-cache.json` in the `directo-mcp` folder). If you change the key's configuration in Directo, tell Claude „Refresh the Directo access info“ (or delete that file and restart Claude Desktop). Directo's overview can omit some enabled interfaces (for example `stocklevels`), so try the tool: the real error message tells you whether access is missing.
- **Stock levels:** prefer the `get_stocklevels_sn` tool. If that interface is not enabled on the key, use `get_stocklevels`.
- **Filters:** every data tool has the usual filters (for example date, customer, status) plus a free-form filter that lets the AI use any field enabled on the key (including row fields). Only fields selected for output on the key can be filtered.
- **Result size:** the start of every answer shows the number of matches and the size. If it says „LARGE“ or „TOO LARGE“, narrow the query (ideally up to 1 month at a time). A very large answer may not fit in the AI's memory and totals may end up incomplete.
- **Sensitive data:** the people, payroll, expense and usage-log tools (`get_wage_calculations`, `get_absences`, `get_expenses`, `get_approvals`, `get_usagelogs`) work if the matching interfaces are enabled on your key. The key owner decides, and is responsible for, whether these are opened to the AI.

8. Security

Measure	Explanation
Credential Manager (both clients)	The API key is encrypted and visible only to your Windows account — Claude and Gemini read the same entry
Environment variable (Gemini fallback)	If Credential Manager is set up, the `env` block isn't strictly necessary — see the note under Step 5.4
Local server	The MCP server only runs on your computer, it is not publicly reachable
Data to the AI provider	Query results are sent to the AI provider's cloud so the model can answer them. The API key never leaves your computer.

Measure	Explanation
Minimal permissions	Restrict the Directo API key's permissions to only the interfaces and fields you need (including people and payroll data)
Access-overview cache	The file `discovery-cache.json` (in the `directo-mcp` folder) contains only interface and field names, not data, and is tied to your key
OneDrive/Dropbox	Exclude the `directo-mcp` folder from cloud sync
Admin rights	Only for installing Node.js (Step 3.1) — a shared prerequisite for both paths — everything else, including the CredentialManager module, runs under `-Scope CurrentUser` without admin rights (see 2.4)

9. Updating and versions

Which version am I running?

The MCP server version is in the server's start-up line in the log (`Directo MCP server v… started (N tools)`):

```
Select-String -Path "$env:LOCALAPPDATA\Claude\logs\mcp-server-directo.log" -Pattern "Directo MCP server v" | Select-Object -Last 1
```

The scripts' version is visible on their first line when you run them (for example `v1.5`).

How to update

1. Download a **fresh** ZIP (see Step A1), extract it and open PowerShell in the extracted folder.
2. Copy the new `directo-mcp.js` over the old one (command below).
3. Close Claude Desktop **completely** (Step 4.3) or Gemini CLI and open it again.
4. Check the version in the log (see above).

Copy command:

```
Copy-Item .\directo-mcp.js "$env:USERPROFILE\directo-mcp\directo-mcp.js" -Force
```



The new `directo-mcp.js` needs no new dependencies, so `npm install` is not needed. Running the install scripts again does **not** update the server file: the script copies `directo-mcp.js` only if it is not in the folder yet.

10. References

Claude Desktop:

- Download Claude Desktop
- Claude Desktop MCP setup

- Guide: local MCP servers on Claude Desktop
- Claude Desktop data folders on Windows (Anthropic)
- Official MCP documentation

Gemini CLI:

- Gemini CLI documentation
- Gemini CLI MCP setup
- Gemini CLI quotas and pricing
- Gemini CLI transition to Antigravity CLI (announcement)
- Google AI Studio API keys
- Gemini Code Assist documentation

Directo:

- Directo API documentation

General:

- Download Node.js
- CredentialManager PowerShell module

Quick-install scripts (see Path A, Chapter 3):

- `directo-mcp.js` — MCP server (41 tools)
- `set-directo-apikey.ps1` — API key management (initial setup + rotation)
- `install-claude-mcp.ps1` — automatic Claude Desktop install
- `install-gemini-mcp.ps1` — automatic Gemini CLI install

From:
https://wiki.directo.ee/ - **Directo Help**

Permanent link:
https://wiki.directo.ee/en/ask_directo

Last update: **2026/11/27 11:12**