

AI OS Controller 6.1

User Manual

Windows · Linux · macOS
Google Gemini · Anthropic Claude · Ollama (local)

Manual for the AiOsController.zip package
October 2026

Contents

- 1. Introduction**
- 2. Installation and first start**
- 3. The main window**
- 4. Asking questions**
- 5. Command approval — staying in control**
- 6. Answers and the chat area**
- 7. Diagnostics**
- 8. Privacy and data masking**
- 9. Local AI with Ollama**
- 10. Settings**
- 11. Menu reference**
- 12. Text version (aioscli)**
- 13. Start parameters**
- 14. Files and storage**
- 15. Troubleshooting**
- 16. Safety summary**

1. Introduction

AI OS Controller is a desktop assistant that lets an artificial intelligence diagnose and manage your computer for you. You ask a question in plain language — "*Why is my computer slow?*", "*How much free disk space do I have?*" — and the AI investigates by running shell commands on your machine, reads the results and explains what it found.

The most important rule: **the AI can only run commands you allow**. Every command is shown to you first, and you decide whether it runs, whether you want to change it, or whether it should not run at all.

1.1 What it can do

- Answer questions about your computer's hardware, software, network, disks, logs and performance.
- Run built-in **diagnostic checks** and let the AI analyse the results.
- Work with three kinds of AI:
 - **Google Gemini** (cloud, needs an API key)
 - **Anthropic Claude** (cloud, needs an API key)
 - **Ollama** — a local AI model such as *qwen3* running on your own computer (free, offline, private)
- **Mask sensitive data** (IP addresses, user names, passwords, serial numbers ...) before anything is sent to a cloud AI.
- Show answers **nicely formatted**, with *Copy* and *Run* buttons on code blocks.
- Keep a complete **log** of every conversation and every command output.

1.2 Supported systems

Operating system	Shells used
Windows 10 / 11	Windows PowerShell 5.1, PowerShell 7 (<code>pwsh</code>), <code>cmd</code>
Linux (incl. WSL and containers)	<code>bash</code> , <code>zsh</code> , <code>sh</code>
macOS	<code>zsh</code> , <code>bash</code>

The application detects your real operating system, version, CPU, memory, shell, administrator rights and package manager automatically, so the AI always knows what kind of machine it is working on.

1.3 Two versions

Version	Program file	Use it when
Desktop app (graphical window)	<code>AiOsController.exe</code>	You work on a normal PC or laptop with a screen. This manual mainly describes this version.
Text version <code>aioscli</code>	<code>aioscli.exe</code>	You work on a server, over SSH, or in a terminal window. See chapter 12.

Both versions are in the same folder and share the same engine, settings and log files.

2. Installation and first start

You have downloaded `AiOsController.zip`; this manual came with it. This chapter explains how to unpack the package, what you need to install first, and how to start the program for the first time. AI OS Controller has **no installer**: you unzip the folder and run the program from there.

2.1 What is in the package

After unzipping you have one folder, `AiOsController`, containing:

File / folder	What it is
<code>AiOsController.exe</code>	The desktop app — this is the file you start
<code>aioscli.exe</code>	The text version for terminal use (chapter 12)
<code>AiOsController.dll</code> , <code>aioscli.dll</code> , <code>AiOsController.Core.dll</code>	The program itself (also used to start it on Linux/macOS)
<code>Avalonia.*.dll</code> , <code>SkiaSharp.dll</code> , <code>HarfBuzzSharp.dll</code> , ...	Libraries for the graphical interface
<code>runtimes\</code>	Native graphics libraries for Windows, Linux and macOS
<code>*.deps.json</code> , <code>*.runtimeconfig.json</code>	Start-up information for .NET
<code>AI-OS-Controller-User-Manual.pdf</code> / <code>.md</code>	This manual

Important: keep all these files **together in the same folder**. Do not move `AiOsController.exe` out of it on its own — create a shortcut instead (see 2.4).

2.2 Requirement: the .NET 9 Runtime

The package does not include .NET itself, so your computer needs the **.NET 9 Runtime** (version 9.0 or a newer 9.x update). Many computers already have it.

Check whether it is installed — open a terminal (on Windows: *PowerShell* or *Command Prompt*) and type:

```
dotnet --list-runtimes
```

If the output contains a line starting with `Microsoft.NETCore.App 9.`, you are ready. If you get an error, or only other versions are listed, install it:

1. Go to <https://dotnet.microsoft.com/download/dotnet/9.0>.
2. Under **.NET Runtime 9.0.x**, download the installer for your system — on a normal Windows PC: **Windows x64**. (The *.NET Desktop Runtime* also works, because it includes the .NET Runtime. You do **not** need the SDK.)
3. Run the installer and follow the steps. No restart is normally needed.

If you start AI OS Controller without .NET 9, Windows shows a message such as "*You must install or update .NET to run this application*", often with a download button. Follow it, or use the link above.

2.3 Unpacking the zip (Windows)

1. **Unblock the zip first (recommended).** Right-click `AiOsController.zip` ► **Properties**. If you see a **Security: "This file came from another computer..."** line with an **Unblock** checkbox, tick it and click **OK**. This prevents extra security warnings for each file later.
2. Right-click the zip ► **Extract All...** and choose a location you control, for example:
 - `C:\Users\<you>\Apps\AiOsController` , or
 - `D:\Tools\AiOsController` , or
 - a USB stick (see *portable mode* in chapter 14).

Avoid `C:\Program Files` : the program may want to write files next to itself (portable mode), which is not allowed there without administrator rights.

3. Open the extracted `AiOsController` folder.

Do not start the program from **inside** the zip window (double-clicking the exe while browsing the zip). Windows then only extracts that one file to a temporary folder and the program will not start correctly. Always use *Extract All* first.

2.4 Starting the program

Windows

1. Double-click `AiOsController.exe` .
2. The first time, Windows SmartScreen may show "**Windows protected your PC**", because the program is not signed by a known publisher. Click **More info ► Run anyway**. You only need to do this once.
3. Optional — make it easy to start next time: right-click `AiOsController.exe` ► **Show more options ► Send to ► Desktop (create shortcut)**, or right-click ► **Pin to Start**.

Linux and macOS

The package was built for Windows (`.exe` files), but the program itself is cross-platform and contains the Linux and macOS graphics libraries. With the **.NET 9 Runtime** for your system installed (see 2.2), open a terminal in the extracted folder and run:

```
dotnet AiOsController.dll      # desktop app
dotnet aioscli.dll             # text version
```

2.5 First start: getting an API key

When you start the app for the first time and no API key is set for the selected AI, the **Settings** window opens automatically.

1. Choose which AI you want to use:
 - **Gemini** — get a key at Google AI Studio: <https://aistudio.google.com/apikey>
 - **Claude** — get a key at the Claude Console: <https://platform.claude.com/settings/keys>
 - **Ollama** — no key needed; see chapter 9.
2. Paste the key into the **API key** field of that provider (the *Get a key...* button opens the right web page).
3. Leave **Remember API keys** ticked if you don't want to enter the key again.
4. Click **Save**.

Tip: If the environment variable `GEMINI_API_KEY`, `GOOGLE_API_KEY`, `ANTHROPIC_API_KEY` or `OLLAMA_API_KEY` is set on your computer, it takes priority over the key typed in Settings. The Settings window shows a note when this is the case.

If you only want to use a **local AI** with Ollama, you don't need any key: choose **Ollama (local)** and follow chapter 9.

2.6 The welcome message

After start-up the chat area shows what the app detected, for example:

```
Detected system: Windows 11 Pro
Shell: Windows PowerShell 5.1 · running as normal user
AI: Google Gemini · gemini-flash-latest
Privacy: masked for cloud AI
```

The window title also shows the detected operating system.

3. The main window

The main window consists of six parts, from top to bottom:

Part	Purpose
Menu bar	File, Edit, AI, Privacy, Diagnostics and Help menus
Toolbar	Picklists for AI , Model and Commands (approval mode), the ↻ refresh button and ⚙ Settings
Chat area	Your questions, the AI's answers, the commands that were run and their output
Quick-prompt row	Quick prompts... picklist, the <i>Run input as a command myself</i> checkbox and a character counter
Input box	Where you type your question, with the Send and ■ Stop buttons
Status bar	Busy indicator, status text, privacy shield 🛡, token usage and system summary

3.1 Toolbar

- **AI** — choose *Gemini*, *Claude* or *Ollama (local)*. You can switch at any moment; the conversation simply continues with the new AI.

- **Model** — the model of the selected AI. Choose **Other model...** to type any model name yourself.
-  (or **F5**) — fetches the list of models your API key (or Ollama installation) can actually use.
- **Commands** — the command approval mode (see chapter 5).
-  **Settings** — opens the Settings window (chapter 10).

3.2 Status bar

- A moving **progress bar** appears while the AI is working.
- The **status text** shows what is happening: *Ready*, *Working...*, *Waiting for your approval...*, *Fetching model list...*, etc.
- The  **privacy indicator** shows whether data masking is active. **Click it** to see which values have been hidden in this chat.
- **Tokens** shows how many tokens were sent and received (relevant for cloud AI costs).
- The last item summarises the system and whether you run as normal user or as Administrator/root.

4. Asking questions

4.1 Typing a question

1. Click in the input box at the bottom.
2. Type your question. There is **no length limit** — you can paste whole error messages or log excerpts.
3. Press **Enter** (or click **Send**).

Key	Action
Enter	Send the message
Shift+Enter	New line in the input box
Ctrl+Enter	Always sends (also when <i>Enter sends</i> is switched off)
Ctrl+Up / Ctrl+Down	Recall the previous / next message you sent

The behaviour of Enter can be changed under **Settings ► Appearance ► Enter sends the message**. When switched off, Enter makes a new line and only Ctrl+Enter sends.

4.2 What happens next

The AI works in a loop:

1. It reads your question and decides which command would give useful information.
2. It **asks to run that command** — you approve, edit or refuse it (chapter 5).
3. The command runs and the output appears in the chat (folded under an **Output** header with status, duration and size).
4. The output is sent back to the AI, which may ask for more commands.
5. When it has enough information, it writes its final answer.

By default the AI may run up to **25 commands** for one message (configurable in Settings).

4.3 Quick prompts

Click **Quick prompts...** above the input box to pick a ready-made question. It is placed in the input box so you can edit it before sending. Available prompts include:

- Why is my computer slow right now? Investigate and explain.
- Check free disk space and tell me what uses the most space.
- Are there errors in the system logs from today? Summarise them and tell me which matter.
- Which programs are listening on network ports?
- Check my internet connection and DNS, and tell me if anything is wrong.
- Are there updates pending for this system?
- Check the health of my disks / SSD.
- What starts automatically at boot or login?
- Give me an overview of this machine's hardware.
- Which processes use the most memory, and is that normal?

Extra prompts appear depending on your system — for example *Windows Defender status* and *devices with driver problems* on Windows, *failed systemd services* and *firewall status* on Linux, and *battery health* on laptops / macOS.

4.4 Running a command yourself

Tick **Run input as a command myself (no AI)** and type a command instead of a question. It runs directly in the shell; the output is added to the chat so the AI can use it in later questions. The same is available via **Diagnostics ► Run a command myself...**

4.5 Stopping

Press **Esc** or click **■ Stop** to stop the AI immediately. A command that is still running is killed, including all processes it started.

5. Command approval — staying in control

5.1 The three modes

Choose the mode in the **Commands** picklist on the toolbar, in **AI ► Command approval**, or in **Settings ► Commands & safety**.

Mode	What happens
Ask before every command (safest) — <i>default</i>	Every command is shown to you before it runs.
Auto-run safe commands, ask for risky ones	Read-only commands run straight away. Risky commands still ask for confirmation.
Chat only — never run commands	The AI never runs anything. It explains and gives you the commands to run yourself.

5.2 The approval window

When the AI wants to run a command, the window **"Run this command?"** opens.

- The **command** is shown in an editable box. **You can change it** before running it.
- If the command looks risky, a **red warning box** explains why (for example *"deletes files or folders"*).
- **Run command** — runs the (possibly edited) command.
- **Don't run** — refuses. You may first type a **comment for the AI** in the field above the buttons, e.g. *"only read, don't change anything"*. The comment is sent only when you click *Don't run*, so the AI understands what you want instead.
- **Copy** — copies the command to the clipboard.

5.3 What counts as "risky"

Commands matching one of the following categories are always treated as risky and always require confirmation, also in auto mode:

Windows	Linux / macOS
Deleting files or folders (<code>Remove-Item</code> , <code>del</code> , <code>rd</code>)	Running with elevated privileges (<code>sudo</code> , <code>su</code> , <code>doas</code> , <code>pkexec</code>)
Formatting or wiping disks (<code>Format-Volume</code> , <code>diskpart</code> , <code>Clear-Disk</code>)	Deleting files (<code>rm</code> , <code>shred</code> , <code>find ... -delete</code>)
Modifying the registry (<code>reg add/delete</code> , <code>Set-ItemProperty</code>)	Formatting or writing disks (<code>mkfs</code> , <code>fdisk</code> , <code>parted</code> , <code>dd</code>)
Shutting down or restarting	Shutting down or restarting (<code>reboot</code> , <code>shutdown</code> , <code>systemctl poweroff</code>)
Changing boot configuration (<code>bcdedit</code>)	Stopping or changing services (<code>systemctl stop/disable/restart</code> , <code>launchctl unload</code>)
Changing the execution policy	Killing processes (<code>kill</code> , <code>killall</code> , <code>killall</code>)
Disabling / uninstalling / removing anything	Changing permissions or ownership (<code>chmod</code> , <code>chown</code>)
Changing permissions (<code>icacls</code> , <code>takeown</code> , <code>fsutil</code>)	Moving / renaming files (<code>mv</code>)
Changing user accounts (<code>net user</code> , <code>New-LocalUser</code>)	Installing, upgrading or removing software (<code>apt</code> , <code>dnf</code> , <code>pacman</code> , <code>brew</code> , <code>pip</code> , <code>npm</code> ...)
Changing or stopping services (<code>sc</code> , <code>Stop-Service</code>)	Changing user accounts (<code>useradd</code> , <code>passwd</code>)
Killing processes (<code>Stop-Process</code> , <code>taskkill</code>)	Changing network or firewall (<code>iptables</code> , <code>ufw</code> , <code>firewall-cmd</code>)
Changing network / firewall (<code>netsh</code> , <code>firewall cmdlets</code>)	Overwriting system files (<code>sed -i</code> , <code>truncate</code> , writes to <code>/etc</code>)
Installing software (<code>winget</code> , <code>choco</code>)	Downloading and executing a script (<code>curl ... sh</code>)
Downloading and executing code (<code>Invoke-Expression</code> , <code>iex</code>)	Changing macOS system settings (<code>defaults write</code> , <code>nvrnm</code> , <code>csrutil</code>)

Warning: The risk check is a **heuristic safety net, not a sandbox**. It cannot recognise every dangerous command. **Always read a command before you approve it** — especially when using a small local model.

5.4 How commands are run

- Commands run **non-interactively**: they cannot wait for keyboard input, pagers are switched off, and Linux package managers run in non-interactive mode.
- Each command has a **timeout** (default 120 seconds). A command that runs too long is stopped and marked *TIMED OUT*.
- Only the first part of the output (default 15 000 characters) is sent to the AI. The **full output** is always written to the log file.
- Commands always run on the computer where **this app** runs, with the rights of the user who started the app.

5.5 Administrator rights

Some checks need administrator (Windows) or root (Linux/macOS) rights.

- **Windows:** use **File ► Restart as Administrator**. Windows asks for confirmation; the current chat is closed but its log is kept.
- **Linux/macOS:** the app knows when it is not running as root and tells the AI that `sudo` cannot ask for a password. Start the app with the necessary rights if you need them.

6. Answers and the chat area

6.1 Formatted answers

AI answers are shown formatted (Markdown): headings, **bold** and *italic* text, `inline code`, bullet and numbered lists (also nested), ☒ task lists, quotes, tables and clickable links (only `http / https`).

Code blocks in an answer have two buttons:

- **Copy** — copies the code to the clipboard.
- **► Run...** — runs the code as a command. **This always opens the approval window first**, whatever approval mode is selected.

To see plain text instead, switch off **Settings ► Appearance ► Show AI answers formatted**.

6.2 Command output in the chat

Every command that ran appears in the chat with an **Output** section showing:

- the status (success, error code, *TIMED OUT*),
- the duration in seconds,
- the number of characters of output.

Refused commands are shown with 🚫 **Refused**, including your comment if you gave one.

6.3 Copying and exporting

Action	Where
Copy the last answer	Edit ► Copy last answer
Copy the whole chat as Markdown	Edit ► Copy whole chat (Markdown)
Clear the input box	Edit ► Clear input box
Save the chat as a Markdown file	File ► Export chat as Markdown... (Ctrl+S)
Save a version safe to share (sensitive values masked)	Privacy ► Export chat masked (safe to share)...
Copy the masked chat	Privacy ► Copy chat masked

6.4 Starting a new chat

File ► New chat (Ctrl+N) clears the conversation and starts a new log file. Stop the current answer first (Esc) if the AI is still working.

7. Diagnostics

The **Diagnostics** menu contains ready-made check scripts for your operating system. When you choose one, the script runs (after your approval), and its output is sent to the AI for analysis.

7.1 Windows (PowerShell)

Check	What it looks at
General system overview	OS, CPU, memory modules, disks, network adapters, top 10 processes
Hardware / USB / input devices	Devices with problems, mouse/HID and USB devices, related driver events
Event log errors & warnings	Recent errors and warnings from the System and Application logs
Network & connectivity	IP configuration, DNS, gateway and internet test, listening ports
DPC / interrupt latency	DPC and interrupt time, processor power events, audio/HID drivers
GPU & display	Video controllers and drivers, GPU usage, display driver events (TDR), power plan
Windows Update & integrity status	Recent hotfixes, pending reboot, Microsoft Defender status
Power & USB power management	Power plans, device power management, battery (read-only)
Basic repair (restart HID services & Explorer)	Restarts input services and Windows Explorer — changes the system , asks for extra confirmation

7.2 Linux

Check	What it looks at
General system overview	OS, kernel, uptime, CPU, memory, disks, top processes, failed services

Check	What it looks at
Hardware / USB / input devices	PCI & USB devices, input devices, kernel messages about USB/HID, sensors
System log errors & warnings	Journal (or syslog) errors and warnings since last boot, failed services
Network & connectivity	Interfaces, routes, DNS, ping test, DNS lookup, listening ports
Performance snapshot	Load & pressure, vmstat, swap, biggest memory users, I/O statistics
GPU & display	GPU hardware and drivers, NVIDIA status, kernel modules, display session, OpenGL
Updates & packages	Reboot-required flag, upgradable packages (apt, dnf, pacman, zypper, flatpak)

Tools that are not installed are simply skipped.

7.3 macOS

Check	What it looks at
General system overview	macOS version, hardware (serial number hidden), memory, disks, top processes
System log errors	Errors/faults from the unified log in the last hour, recent crash reports
Network & connectivity	Interfaces, routes, DNS, ping test, listening ports
Hardware / USB / battery	USB devices, power and battery health, thermal state

Note: With `cmd.exe`, or with PowerShell on Linux/macOS, there are no built-in diagnostics. The menu says so — you can still ask the AI anything.

The Diagnostics menu also contains **Run a command myself...** to run your own command (its output is shared with the AI).

8. Privacy and data masking

Command output often contains things you don't want to send over the internet. Before anything is sent to a **cloud AI** (Gemini, Claude or an Ollama *cloud* model), AI OS Controller replaces sensitive values by **placeholders**.

8.1 What is masked

What	Example	Placeholder
Passwords, API keys, tokens, private keys, Wi-Fi keys, credentials in URLs	<code>password=hunter2</code> , <code>sk-...</code> , <code>ghp_...</code>	<code>[SECRET_1]</code>
IPv4 addresses (private vs. public is kept)	<code>192.168.1.37</code> / <code>84.241.192.15</code>	<code>[LAN_IP_1]</code> / <code>[PUBLIC_IP_1]</code>

What	Example	Placeholder
IPv6 and MAC addresses	fe80::a1b2:..., 3C-52-82-1A-2B-4F	[IPV6_1] , [MAC_1]
Wi-Fi network names	SSID name : "Home-Net"	[SSID_1]
User names (also in paths), computer name, domain	C:\Users\Jeroen , DESKTOP-K7Q2M9X	[USER_1] , [HOST_1]
E-mail addresses, Windows SIDs	jan@example.nl , S-1-5-21-...	[EMAIL_1] , [SID_1]
Serial numbers, hardware UUIDs, product keys	SerialNumber : WD-WX12...	[SERIAL_1]
Your own words (name, company, address ...)	set in Privacy settings	[CUSTOM_1]

8.2 How it works

- The **same value always gets the same placeholder** within a chat, so the AI can still reason ("*[LAN_IP_2] is your gateway*").
- When the AI uses a placeholder in a command (`ping [LAN_IP_2]`), the app puts the **real value back locally** before you see and approve it.
- Answers are shown to you with the real values.
- The real values never leave your computer. The log file shows exactly what was sent (masked).
- Harmless values stay visible: `127.0.0.1` , netmasks, well-known DNS servers (8.8.8.8, 1.1.1.1), version numbers, times, and command names like `Format-Table` .

8.3 Masking modes

Set the mode in **Privacy ► Mask sensitive data** or **Settings ► Privacy**:

Mode	Meaning
Off — send everything as is	No masking at all
Only for cloud AI (Gemini, Claude) — <i>recommended, default</i>	Masks for cloud AI; local Ollama sees everything, because nothing leaves your PC
Always — also for local Ollama	Masks for every AI

8.4 Privacy tools

Menu item	What it does
Privacy ► Show hidden values of this chat... (or click ♥ in the status bar)	Lists each placeholder and the real value it stands for
Privacy ► Preview: what would be sent for the input box...	Shows how the text currently in the input box would be masked
Privacy ► Show system prompt as sent...	Shows the system prompt exactly as the AI receives it
Privacy ► Export chat masked (safe to share)...	Saves a Markdown file that is safe to post on a forum or send to a helpdesk

Menu item	What it does
Privacy ► Copy chat masked	Copies the masked chat to the clipboard
Privacy ► Privacy settings...	Opens the Privacy tab of Settings

8.5 Fine-tuning

In **Settings ► Privacy** you can:

- switch each category on or off (secrets; network; identity; hardware),
- add words to **Always hide** (one per line) — your name, company name, street, project names,
- add values to **Never hide** (one per line) — e.g. a public server IP the AI may see,
- paste text in the preview box and click **Preview masking** to test the result.

Limitation: Masking is pattern based. It catches the common formats, but cannot guarantee that everything personal is found — for example a serial number in a table column without a label. When in doubt, add the value to *Always hide*.

9. Local AI with Ollama

With **Ollama** the AI runs on your own computer: free, works offline, and no data leaves your machine.

9.1 Step 1 — Install Ollama

OS	How
Windows	Download and run the installer from https://ollama.com/download . Ollama runs in the background (llama icon in the system tray).
macOS	Download the app from https://ollama.com/download , move it to Applications and start it.
Linux	<code>curl -fsSL https://ollama.com/install.sh sh</code> (installs a systemd service <code>ollama</code>).

9.2 Step 2 — Download a model

This manual uses **qwen3** as the example model. It is a good all-round choice for AI OS Controller because it supports **tool calling** (it can ask to run commands in a structured way, just like Gemini and Claude) and it comes in several sizes. Open a terminal and run:

```
ollama pull qwen3      # = qwen3:8b, about 5.2 GB download
ollama list           # shows installed models, e.g. "qwen3:latest"
```

Choose a size that fits your computer:

Model	Download	Good for
qwen3:4b	2.5 GB	Laptops and PCs with little memory or no graphics card
qwen3 (= qwen3:8b)	5.2 GB	Recommended starting point for most PCs

Model	Download	Good for
qwen3:14b	9.3 GB	Better answers, needs a graphics card with plenty of video memory

If you pick another size, use that exact name (for example `qwen3:4b`) everywhere this manual says `qwen3`.

9.3 Step 3 — Check that Ollama is running

- Open <http://localhost:11434> in a browser. It should say "**Ollama is running**", or
- run `curl http://localhost:11434/api/tags` in a terminal.
- Not running? Windows/macOS: start the Ollama app. Linux: `sudo systemctl start ollama` (or `ollama serve`).

When Ollama and this app run on the same computer **nothing else is needed**: no API key and no extra configuration.

9.4 Step 4 — Connect the app

1. In the toolbar choose **AI → Ollama (local)**.
2. Open ⚙ **Settings ▶ AI providers ▶ Ollama** and check:

Field	Recommended value
Server URL	<code>http://localhost:11434</code>
Model	<code>qwen3</code> — click Check connection first; the list is then filled with your installed models
Context window	<code>16384</code> (qwen3 supports up to <code>40960</code> ; more uses more memory)
Running commands	<code>Automatic</code>
Keep model loaded	<code>15m</code> (<code>-1</code> = always in memory: faster, but uses RAM/VRAM; <code>0</code> = unload immediately)
API key	leave empty

Note: out of the box the app's Ollama **Model** field says `phi4` (an older default). Change it to `qwen3`, or to the model you downloaded, otherwise you'll get "*Model 'phi4' is not installed in Ollama*".

3. Click **Check connection**. The result lists your installed models (e.g. `qwen3:latest`) and tells you whether the selected model supports tools — for qwen3 it does, so commands use **native tool calling**.
4. Click **Save** and ask a question such as "*How much free disk space do I have?*"

9.5 How commands work with local models

Gemini, Claude and many local models — including **qwen3** — support **tool calling**: the model asks for a command in a structured way, and the app shows you the approval window. Some older or smaller models don't. The setting **Running commands** decides how this is handled:

Option	Meaning
Automatic (recommended)	The app asks Ollama what the model can do and picks the right method. For qwen3 this is native tool calling
Native tool calling	For models with tool support, e.g. qwen3 , llama3.1 , gpt-oss:20b , mistral-small3.2
Text protocol	Fallback for models without tool support. The model writes the command in a <code>```run</code> block; the app takes the first block, shows you the approval window, runs it and sends the real output back

If a model unexpectedly reports that it does not support tools, the app switches to the text protocol by itself. With qwen3 you normally never need the text protocol.

Thinking mode. qwen3 is a "thinking" model: it reasons for a while before it answers, which makes answers better but slower. Recent Ollama versions keep that reasoning out of the answer you see. If answers contain a block of text between `<think>` tags, update Ollama. For faster, shorter answers you can add `/no_think` under **Settings ► Prompt & language ► Extra instructions**; qwen3 then skips the thinking step.

Tip: Local models make more mistakes than Gemini or Claude, and smaller sizes more than larger ones. **Always read a command before you approve it.**

9.6 Context window

Ollama's default context is small (2048–4096 tokens). The system prompt plus a few command outputs quickly exceed that, and the model then silently "forgets" the start of the conversation. The app therefore sends the **Context window** setting with every request.

For qwen3, **16384** is a good balance. You can go up to **40960** if your computer has memory to spare, or down to **8192** if it has little. A value of **0** uses the Ollama default (not recommended).

Rough hardware guide for qwen3 (8b) with a 16k context: about 8 GB of video memory for full GPU speed. With less, Ollama runs part of the model on the CPU — it works, but slower. On a PC without a suitable graphics card, qwen3:4b is the better choice.

9.7 Ollama on another computer (optional)

Only needed when Ollama runs on a **different** machine (for example a PC with a big GPU). On the Ollama computer:

- Let Ollama listen on the network by setting `OLLAMA_HOST=0.0.0.0:11434` and restarting Ollama.
 - Windows:** `setx OLLAMA_HOST "0.0.0.0:11434"` in PowerShell, then restart Ollama.
 - macOS:** `launchctl setenv OLLAMA_HOST "0.0.0.0:11434"`, then restart the Ollama app.
 - Linux (systemd):** `sudo systemctl edit ollama.service`, add `Environment="OLLAMA_HOST=0.0.0.0:11434"` under `[Service]`, then `sudo systemctl daemon-reload && sudo systemctl restart ollama`.
- Open port **11434** in the firewall **for your local network only**.
- In the app set **Server URL** to `http://<ip-of-ollama-computer>:11434` and click **Check connection**.

Security warning: Ollama has **no login**. Never forward port 11434 on your router to the internet. Use it only on your home network or over a VPN. If you put a reverse proxy with a token in front of it, enter that token in the **API key** field.

Commands always run on the computer where **this app** runs, not on the Ollama server.

Models whose name ends in `-cloud` run on ollama.com, **not** locally; the app treats them as cloud AI for privacy masking.

10. Settings

Open with ⚙ **Settings**, **AI ► Settings...** or **Ctrl+,**. Click **Save** to apply or **Cancel** to discard changes.

10.1 AI providers

Section	Fields
Google Gemini	API key (<i>Get a key...</i>), Model (default <code>gemini-flash-latest</code>), Base URL, API version (default <code>v1beta</code>)
Anthropic Claude	API key (<i>Get a key...</i>), Model (default <code>claude-sonnet-5-5</code>), Base URL, API version header (default <code>2023-06-01</code>)
Ollama	Server URL + <i>Check connection</i> , Model + <i>Other...</i> , Context window, Running commands, Keep model loaded, API key (optional). <i>Download Ollama...</i> opens the download page.
General	<i>Show API keys</i> , <i>Remember API keys</i> , <i>Max output tokens per answer</i> (default 8192)

API keys are stored **encrypted with your Windows account** on Windows; on Linux/macOS the settings file is readable only by you.

10.2 Commands & safety

Setting	Default	Meaning
When the AI wants to run a command	Ask before every command	The approval mode (chapter 5)
Shell	auto	<code>auto</code> , <code>powershell</code> , <code>pwsh</code> , <code>cmd</code> , <code>bash</code> , <code>zsh</code> or <code>sh</code>
Command timeout (seconds)	120	A command running longer is stopped
Max characters of output sent to AI	15 000	The rest is only written to the log
Max command rounds per message	25	Maximum number of commands the AI may run for one question

10.3 Privacy

Masking mode, categories, *Always hide* / *Never hide* lists and the masking preview — see chapter 8.

10.4 Prompt & language

- **Answer language** — `auto` (answers in the language you write in), English, Dutch, German, French, Spanish, Italian, Portuguese or Polish.
- **Send detected system details to the AI** — OS, version, CPU, RAM, host name, user ... Switch off if you prefer not to share these (with masking on, names are still replaced by placeholders).
- **Extra instructions for the AI** — free text added to the system prompt, e.g. *"I am a beginner, explain every step."* or *"Never restart services without asking."*
- **Preview full system prompt...** — shows the complete instructions the AI receives.

10.5 Appearance

- **Theme** — `system`, `light` or `dark`.
- **Chat font size** — 10 to 28 (default 14).
- **Show AI answers formatted** — Markdown rendering on/off.
- **Enter sends the message** — when off, Ctrl+Enter sends.
- The bottom of the tab shows where the settings and log files are stored.

11. Menu reference

File

Item	Shortcut	Description
New chat	Ctrl+N	Clear the conversation, start a new log file
Export chat as Markdown...	Ctrl+S	Save the conversation to a <code>.md</code> file
Open logs folder		Opens the folder with all log files
Open current log file		Opens the log of this chat
Open settings folder		Opens the folder containing <code>settings.json</code>
Restart as Administrator		Windows: restart with admin rights
Exit	Alt+F4	Close the app

Edit

Copy last answer · Copy whole chat (Markdown) · Clear input box.

AI

Item	Shortcut	Description
Provider		Gemini, Claude or Ollama
Command approval		Ask / Auto-run safe / Chat only
Refresh model list	F5	Fetch available models
Use another model...		Type any model name

Item	Shortcut	Description
Test connection		Sends a short test message and shows the answer
Show system prompt		Shows the instructions sent to the AI
Settings...	Ctrl+,	Opens Settings

Privacy

Mask sensitive data · Show hidden values of this chat... · Preview: what would be sent for the input box... · Show system prompt as sent... · Export chat masked... · Copy chat masked · Privacy settings...

Diagnostics

The diagnostic checks for your system (chapter 7) and *Run a command myself...*

Help

Item	Shortcut	Description
System information		What was detected about this computer, plus settings/log locations
Tips & keyboard shortcuts	F1	Quick reference
About		Version and runtime information

Keyboard shortcuts — summary

Key	Action
Enter / Shift+Enter	Send / new line (switchable in Settings)
Ctrl+Enter	Always send
Ctrl+Up / Ctrl+Down	Previous / next sent message
Esc	Stop the AI and kill a running command
Ctrl+N	New chat
Ctrl+S	Export chat as Markdown
Ctrl+,	Settings
F5	Refresh model list
F1	Tips & keyboard shortcuts

12. Text version (aioscli)

For servers, SSH sessions and machines without a desktop, use the text version. It is in the same folder as the desktop app.

Windows: open the `AiOsController` folder in File Explorer, right-click an empty spot ► **Open in Terminal** (or type `powershell` in the address bar and press Enter), then run:

```
.\aioscli.exe # start with saved settings
.\aioscli.exe --provider=ollama --model=qwen3 # start with a local model
```

Double-clicking `aioscli.exe` also works; it opens in its own console window.

Linux / macOS: in a terminal in the extracted folder:

```
dotnet aioscli.dll
dotnet aioscli.dll --provider=ollama --model=qwen3
```

Type a question after `You:` and press Enter. For multi-line input, end a line with a backslash (`\`) to continue on the next line, or type `""` on its own line to start and stop a multi-line block.

Ctrl+C stops the current answer or command without closing the program.

When the AI wants to run a command you are asked:

```
Run it? [y]es / [n]o / [e]dit / or type a reason to refuse:
```

12.1 Commands

Command	Description
<code>/provider</code> <code>[gemini claude ollama]</code>	Show or switch AI provider
<code>/model</code> <code>[name]</code>	Show or set the model
<code>/models</code>	List models available for your key
<code>/key</code>	Enter or replace the API key of the current provider (input hidden)
<code>/test</code>	Test the connection
<code>/approve</code> <code>[ask auto chat]</code>	ask = confirm every command, auto = only risky ones, chat = never run
<code>/diag</code> <code>[id]</code>	List built-in diagnostics, or run one (e.g. <code>/diag network</code>)
<code>/run</code> <code><command></code>	Run a command yourself (output is shared with the AI)
<code>/sysinfo</code>	Show what was detected about this system
<code>/prompt</code>	Show the system prompt sent to the AI
<code>/usage</code>	Token usage of this chat
<code>/new</code>	New chat (clears history, new log file)
<code>/export</code> <code>[file]</code>	Save this chat as Markdown
<code>/export-masked</code> <code>[file]</code>	Save with sensitive values replaced (safe to share)
<code>/mask</code> <code>[off cloud always]</code>	Data masking mode
<code>/masked</code>	Show which values are hidden and their placeholders
<code>/preview</code> <code><text></code>	Show how a text would be masked

Command	Description
/exit	Quit

13. Start parameters

Both the desktop app and `aioscli` accept these parameters. Example (Windows, in a terminal in the program folder):

```
.\AiOsController.exe --provider=claude --approval=chat
```

You can also add parameters to a desktop shortcut: right-click the shortcut ► **Properties** and append them to the **Target** field, after the closing quote of the path.

Parameter	Meaning
<code>--provider=gemini claude ollama</code>	AI to use
<code>--model=NAME</code>	Model name
<code>--key=APIKEY</code>	API key for this session
<code>--approval=ask auto chat</code>	Command approval mode
<code>--shell=auto powershell pwsh cmd bash zsh sh</code>	Shell to use
<code>--gemini-url=URL / --claude-url=URL</code>	Alternative API address
<code>--api-version=V</code>	API version
<code>--data-dir=PATH</code>	Folder for settings and logs
<code>--no-save</code>	Don't save settings changes

14. Files and storage

What	Location
Settings (Windows)	%APPDATA%\AiOsController\settings.json
Settings (Linux/macOS)	~/.config/AiOsController/settings.json
Logs	logs/chat_<date>.log in the same folder — one file per chat

- **Portable mode:** put a `settings.json` next to `AiOsController.exe` and that file is used instead (handy on a USB stick). The logs are then also kept in that folder. An empty file containing just `{}` is enough to start in portable mode.
- **Logs** contain every request, every response and the **full** output of every command (the AI only receives the first part). What was sent to a cloud AI appears in masked form.
- **Upgrading from v5 (Gemini OS Controller):** an old `settings.json` next to the exe is recognised; the Gemini key and model are taken over.

14.1 Updating to a newer version

1. Close AI OS Controller.

2. Unzip the new package and replace the contents of your `AiOsController` folder with it (or extract it to a new folder and update your shortcut).
3. Your settings, API keys and logs are kept, because they are stored in your user profile — not in the program folder. **In portable mode**, copy your `settings.json` (and the `logs` folder if you want it) into the new folder.

14.2 Removing the program

There is no uninstaller, because nothing was installed:

1. Close the app and delete the `AiOsController` folder (and any shortcuts you made).
2. To also remove your settings, saved API keys and logs, delete the settings folder: `%APPDATA%\AiOsController` on Windows (paste it into the File Explorer address bar), or `~/.config/AiOsController` on Linux/macOS.
3. The .NET 9 Runtime can stay installed (other programs may use it), or be removed via **Settings ► Apps** on Windows.

15. Troubleshooting

Starting the program

Problem	Solution
"You must install or update .NET to run this application"	Install the .NET 9 Runtime (x64) — see 2.2
"Windows protected your PC" (SmartScreen)	Click More info ► Run anyway . To avoid it for every file, unblock the zip before extracting (2.3)
Nothing happens, or an error about missing DLL files	The program was started from inside the zip, or the exe was moved out of its folder. Extract the whole zip with <i>Extract All</i> and start <code>AiOsController.exe</code> inside the extracted folder
Antivirus quarantines or blocks the program	The app runs shell commands by design, which some security software flags. Restore the file and add an exception for the <code>AiOsController</code> folder only if you trust where you got the package
Settings are not saved	The folder is read-only (e.g. inside <code>C:\Program Files</code>) while in portable mode. Move the folder to a location in your user profile
Linux: window does not open	Make sure a desktop session is running (X11/Wayland). On a server without a desktop, use <code>dotnet aioscli.dll</code>

General

Problem	Solution
"No API key set"	Open ⚙ Settings and paste the key for the selected AI, or switch to Ollama
Connection test fails	Check the key, your internet connection and the Base URL. Use AI ► Test connection

Problem	Solution
Model not found	Press F5 /  to fetch the models your key can use, then pick one
A command needs admin rights	Windows: File ► Restart as Administrator . Linux/macOS: start the app with the required rights
A command times out	Raise <i>Command timeout</i> in Settings ► Commands & safety
The AI stops after many commands	Raise <i>Max command rounds per message</i>
The AI misses part of the output	Raise <i>Max characters of output sent to AI</i> (the full output is always in the log)
The AI answers in the wrong language	Set Settings ► Prompt & language ► Answer language
You want to know exactly what was sent	File ► Open current log file , or Privacy ► Show system prompt as sent

Ollama

Message / symptom	Solution
<i>Cannot reach Ollama at http://localhost:11434</i>	Ollama is not running: start the app, <code>ollama serve</code> , or <code>sudo systemctl start ollama</code> . Check the URL
<i>Model 'x' is not installed in Ollama</i>	<code>ollama pull x</code> , or choose an installed model (Check connection / )
First answer takes very long	The model is being loaded into memory, and qwen3 thinks before answering. Raise <i>Keep model loaded</i> (e.g. <code>1h</code> or <code>-1</code>)
The AI forgets the question or earlier output	Context window too small: set it to 16384 (or higher, up to 40960 for qwen3) and/or lower <i>Max characters of output</i>
Answer ends with "cut off"	Raise <i>Max output tokens</i> or the context window
The model describes a command but nothing runs	Check that Running commands is set to <i>Automatic</i> and that you use a tool-capable model such as <code>qwen3</code> . With a model without tool support the command must be in a <code>```run</code> block; ask "run it"
Very slow	Not enough video memory: use a smaller model (<code>qwen3:4b</code>) or a smaller context window, or add <code>/no_think</code> to <i>Extra instructions</i>
<i>Model 'phi4' is not installed in Ollama</i>	The app's default Ollama model is <code>phi4</code> . Set Model to <code>qwen3</code> in Settings (9.4)
Answers contain <code><think></code> ... text	Update Ollama to a recent version

16. Safety summary

1. **Read every command before you approve it.** The AI can make mistakes — small local models more often.
2. Use **Ask before every command** when you are unsure; use **Chat only** if you only want advice.

3. Risky commands always ask for confirmation, but the risk check is a **safety net, not a sandbox**.
4. Keep **data masking** on for cloud AI, and add personal words to *Always hide*.
5. Use **Export chat masked** before sharing a conversation with others.
6. Never expose an Ollama server to the internet.
7. Everything is logged — use the log file to review what happened.