Nexo Memory user guide

Nexo Memory keeps live, visual and shared memory for working with external AIs without starting from zero.

Welcome

Live, shared and readable memory so the user and AI see the same project.

Nexo Memory in one sentence

Nexo Memory organizes a live project memory: shared, readable and reviewable by the user and by the AIs you choose.

The key value is not only restarting chats. The key value is that the user and the AI look at exactly the same memory: the same cards, lines, rules and current state.

Your data is yours. The memory lives in your Drive or GitHub, and Nexo Memory acts as the local panel to view, organize and validate it.

You can connect that same memory to different AIs. That way the project does not depend on one chat or one AI.

Start by selecting an existing project or creating a new one when you want to test the full flow.

- If you are new to Nexo Memory, open First steps to see what you can do.
- If you already know Nexo Memory, use Projects to select an available project or create a new one.
- Nexo Memory does not replace ChatGPT, Claude, Gemini or Codex: it gives them a common readable memory so they can work better.

First steps

Your first project: create the memory, choose where it is stored and run the first AI work cycle.

Create the first project

A project is the container where Nexo Memory stores live memory: context, tasks, decisions, AI instructions and pending proposals to review.

1. Open Nexo Memory in the side panel.
2. Press Projects.
3. Press Create new project.
4. Write the project name.
5. Write a short description with objective, scope and context.
6. Validate Drive or GitHub as the project memory.

- Recommended name: short, recognizable and stable.
- Recommended description: what is being built, for whom, current state and main objective.

Choose Drive or GitHub

Choose Drive or GitHub while creating the project. That choice creates the single canonical memory: every read, synchronization and proposal uses that provider and its exact path.

Drive is usually better for getting started, personal documentation or non-technical projects. GitHub is usually better for code, versioning and work with Codex.

- Drive: simpler for non-technical users.
- GitHub: better for code projects and version control.
- In new projects, validating Drive checks permissions. Nexo Memory does not create the project folder until you press Create project and memory.
- GitHub can store the token encrypted in Drive with an internal password so you can use the same project from another computer.
- If you later change provider, use Memory migration. Nexo Memory copies and verifies the project before changing its canonical memory.
- Drive always retains the control profile: the global project index, the current canonical-provider pointer and, when GitHub is used, the encrypted token vault. This does not force canonical memory to remain on Drive.

Views and presets

Nexo Memory lets you edit the project and card view without changing the project memory.

New projects start with the Nexo Memory visual preset. Existing projects keep their saved styles when you update the extension.

- Custom presets can be saved with a name and reused.
- Changing a preset affects the view, not the lines or stable memory.

- Base Nexo Memory presets are not overwritten: if you modify them, Nexo Memory asks you to save a named custom preset.
- If you are already editing a custom preset, Nexo Memory asks whether to update that preset or create a new copy.

Guided first flow

Use this walkthrough the first time. It does not try to explain all of Nexo Memory: it only shows the complete work cycle.

The idea is simple: Nexo Memory stores memory, the AI converses and you review what comes back before accepting it.

Once the project has been created:

1. Open an existing card or create a card with a clear definition. Examples: Roadmap, Ideas, Tasks to validate, Prompts, Mockups, Templates, Archive, Decisions, Bugs, Rules, Deliverables or References.
2. Add one simple line with a real task, idea, decision or fact. To edit it, click it and press Edit.
3. Press Open chat and paste it into the AI you prefer. Nexo Memory prepares instructions so the three of you can work together.
4. Work in the external AI.
5. When you finish, use Close chat to request a useful summary and possible proposals. Nexo Memory does not read the chat automatically.
6. Review Pending events. Accept only what is clear and useful; reject or delete anything doubtful, duplicated or noisy.
7. As you keep working, repeat the process or ask the AI to update Nexo Memory.

- Export project = manual copy to save, review or deliver outside Nexo Memory.
- Open chat = instructions and context to start working with an AI.
- Close chat = way to bring back a summary, decisions and reviewable proposals.
- Line = visible row.
- Ficha = the full content of that line, with description, objective, relations, AI instructions and linked files.
- Pending events = review inbox; nothing enters stable memory until you accept it.

Minimum checks

Before continuing to work with an AI, confirm that memory works and that the correct project is open.

- You see the correct project name.
- The status line does not show active errors.
- Canonical memory is available and the AI has direct access or can request AutoBridge.
- The main cards load correctly.

Tutorial

See what each Nexo Memory area does and how to move through the panel.

Sidebar

Visual in Nexo Memory: Visual representation of the Nexo Memory sidebar with its main shortcuts.

The sidebar is the side bar used to enter the main project functions. From here you can enter Projects, search information, synchronize memory, open Docs, AIs and Nexo Memory, adjust settings and prepare instructions for an AI.

Projects lets you create a new project or select an existing one. When creating a new project, Nexo Memory asks for title, short description and memory location. That memory can live in Google Drive or GitHub depending on the project use.

Search helps find concrete lines without going through every tab. Refresh checks the connected memory and brings changes, external documents or pending events that appeared outside Nexo Memory.

- Projects: create or select the active project.
- Search: find lines stored in cards, documents, tasks or events.
- Refresh: synchronize Nexo Memory with Drive or GitHub.
- Docs, AIs and Nexo Memory: direct shortcuts to the main areas.
- Settings: open project, view, memory, logs, guide and trash settings.
- Open chat and Close chat: connect Nexo Memory work with an external AI.

Tabs

Visual in Nexo Memory: Tab menu with card, edit, export and archive actions.

Nexo Memory organizes information in three levels. Tabs are large work areas. Cards organize information inside each tab. Lines are the concrete data you store inside a card.

The tab menu manages a whole tab. From there you can add a card, edit the tab layout, export its content or archive it.

Use tabs when you need to separate large project areas: strategy, development, documentation, deliveries, research or any block with its own purpose.

- Add card: creates a new card inside the selected tab.
- Edit tab: changes the one-column or two-column view.
- Export: generates a copy of that tab to save, share or pass to an AI.
- Archive tab: removes the tab from the main workspace and sends it to Trash.

Cards

Visual in Nexo Memory: Card menu with insert line, edit card, view, columns, export and archive actions.

A card can store ideas, tasks, documents, rules, decisions, processes or work state. What matters is that each card has a clear definition so the user and AIs know what belongs there.

The card menu controls how one card is used, displayed and organized. Insert line creates a new line inside the card. Edit card changes title, description, card type selected from the available types and instructions for AIs.

- Useful card examples: Roadmap, Ideas, Tasks to validate, Prompts, Mockups, Templates, Archive, Decisions, Bugs, Rules, Deliverables or References.
- Create a card only if it will have a clear definition and related lines.
- Edit view changes colors, icons, borders, lines, buttons and visual style. It also lets you save custom visual presets, apply the style to other cards or reset values.
- Edit columns decides which columns are visible, in which order they appear and how important information is read.
- Export prepares that card in useful formats for AI, manual copy or external work.
- Archive sends the card to Trash without making a final destructive deletion immediately.
- Expand, collapse and adjust height help adapt the workspace to the real size of each card.

Lines and fichas

A line is the visible row in a card. The ficha is the full detail of that line: description, objective, notes, AI instructions, hierarchy, relations and linked files.

When you click a line, Nexo Memory opens a small menu with three actions: Edit, Copy MD and Download package.

Edit opens the full ficha. Copy MD copies the ficha as Markdown for pasting into any AI. Download package creates a compressed folder with ficha.md and the readable attachments Nexo Memory can download.

- Use the ficha so a line is not just a title: add enough context for another person or AI to understand it.
- Copy MD is useful when you want an AI to work on one concrete line without exporting the whole project.
- Download package is better when the line has documents, images, PDFs, spreadsheets or other attachments the AI should review.
- If an attachment cannot be downloaded from Drive or GitHub, Nexo Memory leaves its link as a reference inside ficha.md.
- Children, relations and linked files keep context attached to the line instead of scattered through the project.

Docs

Docs gathers the project's live documentation. It is the place for rules, summaries, external documents and history.

General project rules stores stable information any user or AI must respect: behavior, criteria, important decisions and explanations needed to continue well.

Chat summaries stores conversation closures. It prevents losing decisions, tasks or context when a chat ends or gets saturated.

- External documents shows files found in the project folder in Drive or GitHub that have not entered Nexo Memory as lines.
- Change history records changes made by the user, an AI, external synchronization or a change detected in Drive/GitHub.
- Docs should contain useful information for understanding and continuing the project, not temporary noise.

AIs

AIs coordinates how different artificial intelligences work inside the project. It is not a technical area: it is where you define roles, learnings and assignments between AIs.

External AI behavior lets you indicate what each AI should do. For example, one AI can review ideas, another can summarize and another can implement.

AI self-learning stores useful learnings about the user, the project or the way of working. It should be reserved for information that helps in future sessions.

- Tasks sent to AIs works as an internal inbox for assignments or messages intended for another AI.
- Completed tasks keeps the history of tasks already processed.
- The practical difference is simple: sent tasks are pending or in progress; completed tasks were already read, processed or completed.

Nexo Memory

Nexo Memory is the operational center of the project. It controls the current work, the next step, the immediate plan and the proposals an AI wants to turn into memory.

Current AI work has three protected lines: Dónde estamos, Dónde vamos and Plan y bloqueos. It is the live operational engine of the project: current state, next objective and how to move forward with blockers or decisions.

Pending events is one of the key areas. An AI can propose creating, moving or updating information, but the user reviews and decides before it enters live memory.

- A pending event should show requested action, destination, confidence, recommendation, reason, origin and proposed content.
- Accept applies the proposal and turns it into Nexo Memory memory.
- Reject does not apply the proposal, but leaves a record so AIs know that idea was discarded.
- Delete removes the proposal without the same visible rejection history.
- The green dot on the Nexo Memory sidebar icon tells you there are pending events to review.

Settings

Settings adjusts the project, the view, connected memory and general Nexo Memory tools.

Project information changes name, description and tutorial. Language changes the main interface texts when available. Edit view changes the general workspace appearance without changing memory.

Memory shows only the current canonical provider and the option to migrate it to Google Drive or GitHub. Migration copies the complete memory and internal attachments, preserves external links, and changes the canonical provider only after verifying the copy.

The previous location remains as an inactive recoverable copy, but it is not shown as active memory and is never used to hydrate the project. The global index and encrypted GitHub vault remain on Drive.

- Zoom increases or reduces the visual size of the workspace.
- Sign out disconnects the current session, but does not delete memory stored in Drive or GitHub.
- Logs help review synchronizations, changes or errors.
- Pending changes shows the local save queue, its state and retry/cancel actions without exposing full content.
- Guide opens this help.
- Contact is used to request support with useful information.
- Export project generates a manual copy of the project; Open chat prepares instructions for an AI.
- Trash contains archived items. To delete physical files, review Drive or GitHub directly.

Open chat

Open chat prepares project context to start a conversation with an AI. Nexo Memory generates text with the project, memory, cards, important lines, linked documents, rules and expected way of working.

There is no connection mode to select. The AI tries direct access first only when it can open the canonical provider and exact project path. Otherwise it uses AutoBridge when Nexo Memory is open and requests permission in context if needed.

Open chat prevents starting from zero. It turns Nexo Memory memory into a clear entry point for the right AI.

- Direct access and AutoBridge complement each other; they are not exclusive modes.
- With AutoBridge, Nexo Memory answers verified reads and captures Pending events from an authorized AI website.
- If neither direct access nor AutoBridge is available, the AI must say it cannot save and explain that one must be enabled.
- Manual export/import is a recovery tool and is not proof that memory was saved.
- Export project generates an information copy; Open chat prepares working instructions for an AI.

Close chat

Close chat ends a conversation without losing what matters. Use it when the chat is long, a session has finished, decisions were made or ideas appeared that should move into Nexo Memory.

When closing a chat, Nexo Memory prepares a summary of what was discussed, decisions, pending tasks, new ideas and possible useful memory changes.

Close chat can also generate pending events. The user reviews those proposals and decides whether to accept, reject or delete them.

- Open chat gives context to the AI.
- Close chat brings back what was learned into Nexo Memory.
- Pending events prevent a long conversation from becoming unreviewed changes.

Understand Nexo Memory

Understand what Nexo Memory stores, where memory lives and how changes are reviewed.

What Nexo Memory is

Nexo Memory is a local memory and organization layer for projects worked on with external AIs.

Its purpose is to prevent every chat from starting from zero. Memory is structured into cards and lines so the user can see it naturally, organically and in order.

The AI converses. Nexo Memory remembers. The user decides.

Shared memory between AIs

The same project can be used with different AIs. Nexo Memory keeps one shared memory source in Drive or GitHub so ChatGPT, Claude, Gemini, Codex or another AI can work from the same context when given access or an offline package.

That memory is not a chat transcript. It is a structured working state.

- The user can visually review what the AI proposes.
- The AI should read the project memory before proposing changes.
- Nexo Memory keeps the review boundary in Pending events.

Where memory lives

The canonical memory lives outside the local panel: in Drive or GitHub. Nexo Memory keeps a local cache to work quickly, but that cache must be replaceable.

If a project is deleted or missing in external memory, Nexo Memory should not trust an old local cache as if it were the real project.

- Drive is useful for personal or document-heavy work.
- GitHub is useful for code, version control and Codex workflows.

- ProjectIndex helps detect external files, but it does not replace Memory Schema V2 resources.

Pending events

Pending events is the reviewable entry point for changes. An AI can propose, but the user decides.

The goal is not to import everything. The goal is to preserve only information that improves the project's durable memory.

- Accept: applies a proposal.
- Reject: records that the proposal was not accepted.
- Delete: removes a proposal from view.
- Accept all applies proposals in that card, but does not force child proposals if they must remain pending.

Documents and attachments

A document can be linked to a line. If the file is local, Nexo Memory can absorb and store it in the external memory location for that card. If the file already lives in Drive or GitHub, Nexo Memory can keep a real link.

StartHere, ProjectIndex and project summaries should show that an attachment exists, but they should not copy the full content of large documents or PDFs.

- Attached files belong to the card context of the line.
- External documents detected outside Nexo Memory appear in Docs > External documents.
- The user can later decide whether an external document should become a real Nexo Memory line.

Connect AI

Prepare a chat with context and recover what matters when you finish.

Start with context

Use Open chat when starting a conversation with an external AI. Nexo Memory prepares instructions and context so the AI understands the project.

The instructions use one flow: direct access to the canonical provider and exact path when it is truly available; otherwise, AutoBridge with contextual permission.

- Direct access is valid only when the AI can read the canonical provider and exact project path.
- AutoBridge lets Nexo Memory answer verified reads and capture Pending events.
- Every AI write is proposed as a Pending event and read back to verify its eventId.
- Manual export/import provides a recovery package to paste or share, but does not confirm a save.

Close with useful memory

Use Close chat when a conversation has produced decisions, tasks, useful context or proposals that should not be lost.

A good close is a complete but condensed executive summary: enough to know what was discussed, decided, discarded and left pending without rereading the whole chat.

- Ask for a real and complete executive summary when the chat has important context.
- Ask for decisions, pending tasks, discarded ideas and next steps.
- Ask the AI to propose events only for durable memory changes.
- Avoid many tiny duplicated lines.
- Avoid literal transcripts, one-line walls and two-sentence closings.

AutoBridge and authorized AIs

When an AI has no direct access and detects Nexo Memory open, AutoBridge requests the required permission inside the working context. One confirmation stores authorization for that AI's main origin.

Project > Connections > AIs only lists saved authorizations and lets you revoke them; it has no connection selector and does not require manually connecting each AI in advance.

- There is no manual URL field and no mode selector.
- Permission is requested only when needed and requires user confirmation in the browser.
- While canonical memory is synchronizing, the status bar says AutoBridge is disabled and its observer does not process messages.
- When memory is synchronized, AutoBridge resumes automatically and checks the AI's latest completed message for a pending request.
- When Nexo Memory saves new Pending events, AutoBridge confirms their eventId values in the chat with a readable message. Confirmation does not mean they were accepted or applied: they still await your review.
- Nexo Memory ignores paths and parameters: a specific ChatGPT conversation counts as chatgpt.com.

- Authorization remains available for future AutoBridge use on that origin.
- To remove an AI, use Revoke connection on its authorized AI row.

Copy or download context

Nexo Memory can prepare context for copying or downloading as a recovery or manual transfer tool.

The manual package does not replace direct access or AutoBridge and cannot be treated as a saved event until Nexo Memory imports and verifies it.

- It works with any AI that accepts pasted context or files.
- The AI must clearly say when it cannot read real memory.
- It must not invent paths, IDs or states.

After the chat

Return to Nexo Memory and review what the AI proposed. Good memory is not the longest memory, but the memory that lets the project continue with less confusion.

- Accept only proposals with a clear destination.
- Reject vague, duplicated or unverified proposals.
- Delete test noise that does not add useful history.

Prompts for AI work

Example text for asking an AI to work with Nexo Memory context.

How to use them

These texts are support templates. You can copy them, adapt them or use them as a reference when asking an AI to work.

The AI should interpret your intent with the project internal context, technical guide and existing cards.

It can suggest a better way to organize the work, but it must not invent data, IDs, approvals or unverified states.

- The user's order words are concepts: the AI should understand the intent, not wait for exact commands.
- If the AI has real Drive/GitHub access, it should work through Pending events.

- If it does not have real access, it should return a clear package for the user to bring into Nexo Memory.
- If it thinks a new card or tab is needed, it should ask in the chat with name, location and reason.

Useful prompts

Use these examples when you want to guide an AI without writing a long specification from scratch.

Organize what comes next

```
Review the available Nexo Memory context and tell me what should be done now. Interpret my words as concepts, not literal commands. If a suitable card or tab is missing, present the complete structure in chat with name, location, exact card type and reason. Only after my explicit approval emit create_tab or create_card with chatApproval:true and chatApprovalSummary; Nexo Memory still requires my manual acceptance of the event.
```

Create or update memory

```
Extract only durable information from this chat. If something already exists in Nexo Memory, propose update_item instead of duplicating it. If the destination and exact IDs are available, leave the event ready. If any ID or destination is missing, use resolutionStatus=needs_target_resolution. Ask the user only if that missing information blocks the immediate work.
```

Close chat with useful memory

```
Close this chat for Nexo Memory: create a complete but condensed executive summary for Docs > Chat summaries. Write it for human reading, with clear paragraphs, line breaks and useful sections. It must let someone know what was discussed without rereading the full chat. Include the important topics: context, work performed, decisions, accepted changes, pending proposals, open questions, discarded ideas, errors, validations, current state and next steps. Do not turn it into a transcript, one long block or a two-line closing.
```

Review structure without creating it

```
Evaluate whether the current Nexo Memory structure is suitable for storing what we discussed. If it is not suitable, present a new card or tab in chat with name, location, exact card type, recommended fields and reason. Wait for my explicit approval, then emit create_tab or create_card with chatApproval:true and chatApprovalSummary for final review in Pending events.
```

Update AI work state

```
If you have real access to Nexo Memory memory, update through Pending events the Current AI work card in the Nexo Memory tab: Dónde estamos with status, focus and latest progress; Dónde vamos with next objective, exact next step and expected result; Plan y bloqueos with immediate plan, blockers and needed decisions. In each update_item use a short clear title that summarizes the current content of that row, not just the protected label; the title cannot be empty, undefined or generic. Send description as the complete final text, preserve identity through targetItemId/fixedLineLabel and do not use it as history.
```

Security

Review which permissions Nexo Memory uses, where your data stays and what to do if something fails.

Your data

Project memory belongs to the user. Nexo Memory does not need its own central database to store your projects.

If you choose Drive, files live in your Drive. If you choose GitHub, they live in your repository.

- Do not store passwords or tokens in cards.
- Do not share logs before checking that they contain no sensitive data.
- Do not paste private keys or OAuth codes into AI chats.

Permissions

Nexo Memory requests permissions to work as a Chrome side panel, store local settings and connect to Drive or GitHub when the user chooses.

Permissions should match the current flow. If you see old or unnecessary permissions, update the extension.

- identity: allows authentication with services such as Google.
- storage: stores local settings and working cache.
- sidePanel: shows Nexo Memory as a side panel.
- scripting: lets Nexo Memory activate AutoBridge only in the AI website authorized by the user.
- tabs: lets Nexo Memory detect the active AI and show the contextual AutoBridge request when needed.
- Drive/GitHub: used only for external memory and project synchronization.

Account and access

The Nexo Memory account is validated with the verified Google email used by the extension. There is no Nexo Memory UID or code to copy from the website.

Nexo Memory asks Supabase whether that email is active. The Bubble website manages sign-up, payments, cancellations and free accounts; Supabase keeps only the account state needed to authorize the extension.

- The automatic check runs once on each new local calendar day.
- Validate account forces an immediate check without waiting for the next day.
- When the account is active, Free use disappears and Active account since shows the date signed by Supabase.
- When the account is not active, Nexo Memory reads the real Drive-root creation date to calculate the free period; account validation does not create folders or projects.
- If validation fails, or the account is inactive after the free period, memory stays readable but writes are blocked.

Sign out and clear cache

Sign out removes local credentials or connection data saved by Nexo Memory in that browser. This matters on shared computers.

Clear cache removes local extension data. It should not delete persistent memory stored in Drive or GitHub.

- After clearing cache, return to Projects and select the available project when Nexo Memory detects it.
- To remove Google access completely, review your Google account security settings.
- For GitHub, delete or rotate the token from GitHub.
- If the GitHub token was encrypted in Drive, Nexo Memory can ask for the internal password when selecting that project on another computer. Once unlocked, it is stored locally in that browser until you sign out of GitHub in Nexo Memory or clear the cache.

Trash and projects

Archiving a project removes it from the active view and sends it to Trash. It does not automatically delete external memory from Drive or GitHub.

Emptying Trash removes local data for archived projects and clears associated local references such as sync cache, zoom, card heights and local credentials.

- If you archive the active project, Nexo Memory remains with no project open.
- In Trash you can restore an archived project or delete its local copy from this computer.

- Emptying Trash should not delete real Drive or GitHub files.
- If you also want to delete external memory, do it consciously in Drive or GitHub.

Common problems

When something fails, first identify whether the problem is in Nexo Memory, Chrome, Drive, GitHub or the external AI.

- If a project does not appear, check account, provider and permissions.
- If a GitHub project appears but does not open, check token or internal Nexo Memory password.
- If Chrome still uses an old version, reload the extension from dist.
- If Pending events do not appear, check that the files are in the correct folder.
- If you see unverified local cache, refresh memory before trusting that data.
- If an AI cannot read memory, enable direct access or AutoBridge; use a manual package only for recovery and do not treat it as a saved event.

Technical docs

Review advanced details about files, events, synchronization and support.

Technical scope

This tab does not replace the user guide. It explains how memory is structured, how proposals enter and which limits AIs must respect.

Technical documentation must be verifiable against the real product. It must not describe retired functions as active.

Functional architecture

Nexo Memory works as a local frontend for external persistent memory. Drive or GitHub contain the memory to preserve; Nexo Memory displays, edits, validates and synchronizes it.

The external AI is not a required internal part of Nexo Memory. It connects through instructions, context packages and pending events.

- Local Nexo Memory: UI, cache, validation and synchronization.
- Drive/GitHub: persistent project memory.
- Internal AI index: reading entry generated by Nexo Memory so the AI understands the project.
- Pending events: reviewable proposals before stable memory.

- Change history: record of applied tasks or detected changes.

Memory files and folders

Exact names can vary by provider, but the functional contract should remain stable.

- Internal AI index: live context Nexo Memory generates so the AI understands the project.
- memory/manifest.json and memory/*: structured V2 memory used to reconstruct tabs, cards, lines, relationships, history and idempotency.
- nexus-memory.json: Memory V1 removed from the live flow; it is not used as fallback.
- ProjectIndex.json and ProjectIndex.md: lightweight project navigation map with structure, main routes and detected loose external files.
- Nexo Memory/Eventos pendientes/: input folder for event_*.json and event_*.md.
- Nexo Memory/<tab>/<card>/: folders where Nexo Memory can store attachments linked to concrete lines.
- event_*.json: structured event.
- event_*.md: readable Markdown event that Nexo Memory should interpret if it preserves the structure.
- Logs: operational diagnostics, not documentary memory.

Event contract

External AIs must not write stable memory directly. They must propose reviewable changes.

Nexo Memory should only apply allowed actions with a resolvable destination.

- Allowed actions: create_item, update_item, move_item, update_card, create_tab and create_card.
- Before create_tab/create_card, the AI must present the structure in chat and receive explicit approval; the event must include chatApproval:true and chatApprovalSummary, then still needs manual acceptance in Nexo Memory.
- If the exact destination is missing, it must mark needs_target_resolution.
- It must not invent IDs, parentId, paths or relationships.
- move_item must change structural relation or order, not rewrite titles and descriptions as movement text.

Direct access and AutoBridge

Access is not configured as exclusive modes. The AI uses direct access only if it can open canonical memory at the exact path; otherwise it tries AutoBridge.

Open chat and Close chat must prevent an AI from claiming it read or saved files it cannot open or verify.

- Direct: the AI reads real canonical memory before acting and writes only verifiable Pending events.
- AutoBridge: Nexo Memory serves verified reads and captures Pending events from an authorized AI.
- While canonical memory synchronizes, AutoBridge stays disabled; when synchronization finishes, it resumes and checks the latest pending message.
- Without either channel, the AI must say it cannot save, avoid inventing content and explain how to enable one.
- A manual package is for recovery or transfer, not confirmation that anything was saved.

Technical diagnostics

A useful technical report separates version, provider, failed action, visible message and logs.

Logs can be copied from Nexo Memory and sent to support after checking they contain no secrets.

- Nexo Memory version and dist/manifest.json.
- Provider: Drive or GitHub.
- Action: create project, select project, synchronize, import events, open chat or close chat.
- Exact message shown in Nexo Memory.
- Exported or copied operational logs.

Contact

Prepare the information needed to ask for help without sharing sensitive data.

Support contact

If you find an error, copy logs from Nexo Memory and send them with a short description of what you were trying to do.

Contact email: soporte@nexomemory.local

- Include Nexo Memory version.
- Indicate whether you use Drive or GitHub.
- Explain the exact step where the problem appears.
- Attach reviewed logs without tokens, keys or sensitive information.

Before sending logs

Logs are meant for diagnostics. Even so, review them before sharing.

Do not send tokens, passwords, OAuth codes or private content you do not want support to read.