Nyx AI Threat Model

This document explains what Nyx AI protects, what it does not protect, and where the real trust boundaries are. It is written for users, reviewers, store reviewers, and security researchers.

It is intentionally a **public trust document**, not a raw security audit dump. It names limitations honestly, but avoids step-by-step abuse instructions and demonstration commands. Internal audit reports and detailed validation notes should stay private/unbundled until any affected issue is fixed and disclosure is appropriate.

Current review: **1.5.493 (2026-08-20).** The last substantive revision was **1.5.491**, which fixed cloud-proxy locality classification, makes Scheduler egress wording apply to every model task, and aligns the public execution and consent boundaries with the code. It adds no destination or tool authority. The current boundaries and residual risks are stated below; detailed reproduction, detection and validation mechanics remain private.

Historical fixes do not describe current authority. In particular, a recorded sandbox grant that cannot be verified as revoked remains an unresolved local residue and is retried or reported rather than silently counted as removed. Missing protected journal state is not reused as authority for a later sandbox identity. See §2.2 and the residual-risk section for the user-relevant limits.

1. Trust Model

Nyx AI is a **single-user, local-first AI coding desktop app**. The user, their machine, and their workspace configuration are trusted. Nyx AI is designed to reduce risk from:

- the AI model, including model-issued tool calls;
- malicious or misleading files the model may read or generate;
- remote content fetched by user-triggered tools;
- third-party extensions or integrations the user chooses to install.

Nyx AI is **not** designed to defend against a hostile operating system, a hostile logged-in user, malware already running as the same user, or a local administrator.

The important product promises are:

- **No telemetry or analytics.** Nyx AI does not transmit usage metrics, crash reports, install IDs, prompts, files, audit logs, or project contents to NYX LIMITED. Microsoft Store and Windows may separately process acquisition, usage, and health diagnostics under Microsoft's terms, and NYX LIMITED may receive the Partner Center reports Microsoft makes available; the Privacy Policy describes that separate platform processing.
- **Local-first, with explicit cloud opt-in.** Model work stays local only when its resolved endpoint is loopback and the selected Ollama model is not cloud-proxied. Once the user configures or selects a cloud provider, saves a remote Ollama endpoint, or selects a cloud-proxied Ollama model, some model roles may leave the device. The prefer-local routing setting biases

auto-routed roles toward a suitable local model where one is reachable; it is a preference with cloud fallback, not an egress block.

- **Explicit egress.** Network calls are limited to documented features: model providers, web tools, user-installed integrations, the update check (on by default, opt-out), first-use tool downloads, and similar user-visible flows.
- **User control.** Risky model-issued actions go through approval or hard-blocking. Ordinary model-issued high-risk and critical shell commands require native confirmation or are refused in every mode, including Bypass. Two bounded trusted flows do not prompt for each command: a Build flow the user starts may run screened scaffold or verification commands selected by Nyx AI, and enabled Auto-verify may run detected or pinned project verification wrapper commands after edits in a workspace the user trusted, or in any workspace while a Sandbox execution level is live. Auto-verify refuses wrappers classified blocked, high-risk or critical; project scripts invoked by an allowed wrapper run as trusted repository code and their bodies are not semantically screened by Nyx AI. Build commands retain hard blocks, screening and the user blocklist. Critical actions and designated sensitive categories require a fresh explicit one-shot decision and cannot be silently approved in Bypass. The default mode prompts for other risky actions; Auto-accept and the explicit time-limited Bypass mode skip only the prompts described in the app and Terms.
- **No sandbox is 100%.** The sandbox is risk reduction, not a promise that arbitrary code can never do harm.

1.1 Prompt Injection

Files, attachments, project memory, tool results, web pages, integrations, and image/vision descriptions can contain text that tries to impersonate the user or Nyx AI. Nyx AI marks these sources as untrusted data across the normal prompt and session lifecycle. Workspace guidance is deliberately shown to the model to shape project conventions, but is accompanied by a warning that it cannot override the user's instructions, Nyx AI's safety rules, or approvals.

Files that control app behaviour receive additional brokered write protection. The model-facing file tools refuse those writes, and direct command forms are screened and may be refused or require approval. The app's own user interfaces use separate channels for authorised settings and rules changes. These are not an operating-system read or write boundary for every process: an allowed shell process can still access workspace content with the authority of its execution tier, and Standard execution is not a filesystem jail. Review diffs and approval requests; do not treat a workspace guidance file as trusted merely because the app loaded it.

Recovery separates authority from analysis: Nyx AI's bounded direction is a deterministic harness message, while helper-model diagnosis is accepted only as a small validated JSON shape and reaches the primary model as untrusted data. Installed MCP integrations may invoke only tools currently advertised by the approved connected server; the membership is rechecked immediately before RPC.

These instruction-level measures are best-effort: a model can still be influenced by malicious content or misunderstand provenance. Prompt injection is not “solved”, and these measures are not the security boundary. Workspace path validation, mode/capability gates, egress guards, hard command

blocks, and user/native approvals limit what a misled model can actually do. Review important actions and do not approve a request merely because fetched or generated content says that it is safe.

2. Security Layers

Nyx AI uses several layers. They are strongest when considered together.

2.1 Always-On Floor

These controls apply to model-issued, Nyx AI-brokered file and command operations regardless of the selected execution backend. They are not a filesystem jail for an arbitrary child process: user-installed MCP servers, hooks, plugins, language servers, project dev servers, and other trusted extensions have the separate boundaries described in section 3.

Layer	Purpose
Brokered workspace path containment	Nyx AI's file/path tools confine model-requested file operations to the active workspace unless the user explicitly allows another root. This broker boundary remains active whether execution isolation is on or off.
Model-command screening	Known-dangerous model-issued shell command families and suspicious shell patterns are blocked before execution. You can exempt a command you trust by name, in **Settings → Safety**; that allow-list relaxes only the dangerous-**name** screen, only for a single command with no chaining or redirection, and never relaxes the path screen or the encoded/obfuscated-command guards. That allow-list, and the screen it can relax, apply only while execution isolation is on; with isolation off that screen does not run at all, so the allow-list has no effect, and raw shell processes are covered by the narrower built-in refusals described in the paragraph below this table — which still include hard refusals for encoded and obfuscated command forms. Python source instead follows its separate risk, approval, native-confirmation-when-uncontained, environment-scrubbing, and selected-isolation path.
Risk classification	Higher-risk model tool calls require approval. Some commands are always refused.

Layer	Purpose
Secret scrubbing	Known secret patterns are scrubbed by default before cloud model egress; broader personal-data scrubbing is available through stronger sanitizer presets and specific supported flows such as workspace-root redaction. This matching is pattern-based and therefore best-effort: it reduces accidental exposure and is not a guarantee that every secret or personal identifier is caught.
Approval prompts	Risky actions go through Nyx AI's approval policy, which may prompt, require native confirmation, auto-allow only under the user's selected mode/rule, or hard-block.
Audit logging	Sandbox and approval events can be recorded locally for review.

When extra execution isolation is off, raw shell processes start in the active workspace; Python is staged and starts in a Nyx AI temporary directory. Both run with the user's normal filesystem and network permissions. They do not receive raw-command path screening, Job-Object limits, sandbox network controls, or an AppContainer boundary. Built-in hard shell-command refusals still apply to shell commands; risk classification, secret-environment scrubbing, and the applicable approval policy still apply to shell and Python. The brokered file/path-tool boundary is unchanged.

The size of that workspace boundary is the user's choice, so the choice itself is screened before it is accepted. A drive or share root, the Windows directory, and Program Files subtrees are refused as workspace roots with no override. A profile-class folder — the user profile itself, the all-users profile folder, Desktop, Documents, Downloads, Pictures, Music, Videos, `%APPDATA%` / `%LOCALAPPDATA%`, ProgramData, Public, and the OneDrive root — is opened only after a main-process native confirmation that names the folder and states that the model gets read and write over everything inside it. Since 1.5.420 that screen also covers those personal folders when Windows Known Folder Move has relocated them under OneDrive (`<OneDrive>\Documents`, `\Desktop`, `\Downloads`, `\Pictures`, `\Music`, `\Videos`); that is the default offer on a new Windows install, and those paths previously matched nothing and opened with no confirmation at all. The confirmation is put at every open, including the automatic restore of the last workspace, because agreement to a sweeping root is deliberately not stored. Matching for these profile-class folders is by exact path equality, so a project folder *inside* one of them is an ordinary workspace and is unaffected; the refusals above match subtrees as well as the folder itself. This screen is a warning about scope, not a containment control: whichever root is opened, the brokered boundary above applies to it.

Model-proposed persistent verification commands are a separate registration seam: Nyx AI displays the complete normalized list in a native dialog once, refuses any high/critical/blocked command, and

writes nothing when consent or workspace authority changes. Once registered, later verification runs remain silent; the user may also edit the trusted workspace's `.nyx/verify.json` directly.

The always-on floor reduces common mistakes and model misuse. It is not a complete OS isolation boundary by itself.

2.2 Locked Execution: AppContainer

("Locked execution" is this document's internal name for the tier shown in the app's Safety settings as the Sandbox levels — "Sandbox + internet" / "Sandbox · no internet". They are the same AppContainer mechanism, with and without network capability.)

Where supported, Nyx AI launches model-issued shell/Python work inside a Windows AppContainer. This is the strongest execution tier:

- the OS enforces a per-file **write boundary** over the workspace, the sandbox's scratch/dependency folders, and explicitly allowed non-system paths (the allowed-path control refuses Windows and configured Program Files subtrees); toolchain/interpreter, package-cache and common Windows/Program Files locations may retain the read/execute access needed to run, and some of those directories sit inside your own user profile, so this is not a claim that every location outside the workspace is unreadable; none of these automatic grants covers a broad profile root — your settings, saved keys and documents — which a contained process reaches only if you deliberately make one your workspace or add one as an allowed path (§2.1);
- optional no-network mode removes outbound network capability for the contained process;
- runtime-built paths and common path tricks are blocked by the OS boundary, not only by JavaScript checks.

Before Nyx AI applies a sandbox filesystem access grant, it durably records the grant in an encrypted write-ahead journal protected by operating-system-backed secret storage; there is no plaintext fallback. A revocation is treated as complete only when a fresh read of the target's actual permission records proves the grant is gone — success is never reported from intent.

Process identity and filesystem authority are deliberately separate: the identity under which a contained process runs is not the principal written to workspace, scratch, allowed-root and toolchain permissions, and each workspace isolation identity is unique to that workspace. No broad "all application packages" grant is used, and no permission for it is placed on the app's own runtime files. The precise Windows principal types and permission-entry layout are implementation detail and are held in the private security record.

If a target is offline, inaccessible or ambiguous, or the check otherwise cannot prove removal, the outstanding record stays in the journal and the next cleanup retries it. Nothing is reported as revoked that was not proved removed.

If the journal or secure storage cannot be read or durably written, explicit Locked execution refuses to grant access. Automatic mode may select the documented Standard tier only where that would not

mask a blocked, partially applied, or unproven sandbox permission state; those remain fail-closed rather than being reinterpreted as ordinary compatibility failure.

If protected journal state is irreparably unreadable, Nyx AI quarantines the opaque bytes and creates a fresh isolated sandbox identity rather than reusing the old one, so a permission whose record can no longer be read cannot authorize a later Nyx AI sandbox. Stale permissions and profile metadata may remain on disk as local residue (Residual 13).

A GUI launch the user approved through the native OS consent additionally grants the contained workspace identity temporary, least-privilege access to the interactive desktop so the approved program can paint its window. Higher-risk desktop capabilities such as clipboard access, screen reading, and input capture are deliberately excluded from that grant, and the filesystem and network jail is unchanged. The grant is reference-counted and revoked when the last approved GUI process exits and on a proof-required workspace/policy cleanup. Ordinary app quit does not run a blocking desktop-ACL operation while holding the single-instance lock: it retains the write-ahead marker, and a crash/quit-stale grant is revoked fail-closed before the same identity runs again. A consented GUI launch under Locked never silently downgrades to the Standard tier — it refuses instead. (On machines whose execution tier already resolved to Standard, consented GUIs run under Standard's Job UI limits by design.)

A residual risk worth stating plainly: while an approved GUI is alive, another command contained in the same workspace can share some of that temporary desktop access without a consent of its own. The window is bounded by the approved GUI's lifetime, and the filesystem and network boundary is unchanged by it. Narrowing this further is tracked hardening work, so it is documented here rather than claimed as prevented.

Locked execution is the tier to rely on when a real filesystem jail matters; the "Sandbox · no internet" level is the tier to rely on when an OS network boundary also matters. If the user explicitly selects either Sandbox level, Nyx AI fails closed if it cannot start that tier.

2.3 Standard Execution: Job Object Fallback

When AppContainer cannot be used in Automatic mode, Nyx AI can fall back to a Windows Job Object tier. That fallback is never used to mask a blocked, partially applied, or unproven sandbox permission state — those remain fail-closed (§2.2). This tier provides useful containment:

- process lifetime control;
- child process cleanup;
- process and memory limits;
- command and path screening before launch.

However, the Standard tier is **not a kernel filesystem jail** and is **not a complete network jail**. It is defense in depth for compatibility cases. If the Job assignment or resource-limit setup fails, Nyx AI may continue under the always-on floor and surface the reduced boundary in the UI. That status reporting is diagnostic rather than tamper-resistant: on this tier the code Nyx AI runs for the model

runs as you and could interfere with it. Treat it as a diagnostic, not as proof of containment; explicit Locked execution is the fail-closed tier. One case is deliberately excluded from that fallback: when Locked execution is barred because a previous access revocation has not been proved, and the Job tier cannot be started either, the command is refused rather than run with no containment at all. Users who need stronger isolation should use Locked execution and confirm it is active.

During startup, before Automatic has an active backend, screened commands may run on the always-on floor without an OS jail. The active-tier indicator is therefore the authoritative status; users who require fail-closed OS isolation should explicitly select a Sandbox level.

The Standard backend's Job Object containment also has process-start limitations, so its resource/lifetime containment is not an absolute process-tree boundary. Closing that implementation limitation is tracked work. It does not affect Locked execution's AppContainer boundary, which is applied at process creation.

2.4 Optional Hardening

Advanced settings can route some user-configured components, such as MCP servers or hooks, through extra containment. These settings can reduce damage from buggy or malicious extensions, but may break legitimate developer tools. They are off by default where compatibility cost is high.

3. What Runs Outside The Sandbox

Some things intentionally run with normal user permissions. This is important and should not be hidden.

The "Preview port sweep" row is retained as a historical disclosure for 1.5.458–1.5.459. From 1.5.460 there is no launch-reachable sweep of untracked port owners; only Nyx AI-owned, in-memory preview children are stopped.

Component	Why it is outside or partially outside	Practical risk
Electron main process	It is the trusted app core and owns filesystem, IPC, preview, providers, and sandbox setup.	If the main process is compromised, it has normal user permissions.
Renderer process	It is isolated from Node by context isolation and no direct Node integration, but it is not the same as a fully sandboxed browser tab.	Renderer dependency risk should be reviewed carefully.

Component	Why it is outside or partially outside	Practical risk
MCP servers	User-installed tool servers often need filesystem or network access.	Install only trusted servers; they can often act with the user's permissions. With the opt-in "Sandbox MCP servers" setting, stdio servers are spawned in the active isolation backend: Locked execution provides the stronger containment, while under the Standard (Job Object) tier containment is best-effort — if Windows refuses the Job, the server keeps running with normal-user filesystem/network authority, and Nyx AI marks the connection "Containment degraded" on its server card and in the audit log. Install-time native consent, advertised-tool membership, Chat allowlisting, secret scrubbing, and audit still apply; the separate opt-in Low-integrity drop fails closed (a server that cannot be verifiably dropped is not started, and never reconnects at normal integrity). Settings changes that alter or revoke the isolation posture restart or stop the affected contained servers rather than letting a running server silently keep a stale posture, and the switches that weaken isolation never loosen an already-running server's containment.
Plugins	Plugins are workspace manifest bundles. Their command files are model-visible Markdown loaded by Nyx AI, not native plugin code; any MCP servers or hooks they define use the separate process surfaces in the adjacent rows.	Trust is mainly at install/enable time. A commands-only plugin starts no child process, while plugin-defined MCP/hook children have the user's permissions and are outside containment by default unless the matching opt-in setting applies.

Component	Why it is outside or partially outside	Practical risk
Hooks	Hooks are user-authored commands triggered by lifecycle events.	Treat hooks like local scripts the user chose to run.
Language servers (LSP)	Language servers power code intelligence and read your code; they run with normal user permissions, not inside the model shell sandbox. Some language servers and workspace toolchains can execute repository-controlled code when they start or analyse a project, using the user's permissions.	Language servers start only for a trusted workspace (see below). The highest-risk project-code-execution features additionally require a separate off-by-default setting whose enablement takes a native OS confirmation the AI cannot bypass. That setting is not a universal gate: some code-intelligence components can still execute repository-controlled code under workspace trust alone, and no claim is made here that they cannot; closing that gap is tracked. Only trust — and enable project code execution for — repositories whose build code you would run yourself.
Scheduled tasks	User-created recurring work may run later without another chat turn.	A bad task can repeat until disabled.
Historical only (1.5.458–1.5.459): preview port sweep (Windows)	Those releases could inspect and terminate an untracked operating-system process holding a requested preview port. The implementation and its authority are retired in 1.5.460.	Current preview startup stops only children tracked as owned by this Nyx AI process, or selects a free port. It does not enumerate or terminate an unrelated port owner. This row is retained only so the historical release record remains intelligible.

Component	Why it is outside or partially outside	Practical risk
Preview/render surfaces	Previews and document renders may load user or generated content. Puppeteer is spawned by the trusted main process, outside the AppContainer/Job command-execution sandbox.	Chromium's own OS sandbox is attempted for every eligible browser first. Only after all sandboxed candidates fail may one exact executable be retried without that browser sandbox, and only after a fail-closed main-owned native confirmation for the current app session. The prompt states that preview/network guards are not a replacement for Chromium's sandbox. Fatal or timed-out Edge is never eligible.
System tool installers	User-approved installs can change the machine permanently.	Nyx AI must show clear consent before these actions.

The model's use of these surfaces should be gated by UI prompts, install prompts, scoped permissions, or explicit user configuration.

What "trusted workspace" means. Several rows above are gated on workspace trust. Trust is granted per workspace root by an explicit user decision — a native OS confirmation dialog (also reachable from Settings → Safety) whose wording states exactly what trust enables. The decision is stored in Nyx AI's own profile directory, never inside the workspace, so a repository can never mark itself trusted. Until a workspace is trusted, Nyx AI will not run that workspace's own executable configuration — its hooks, MCP servers, or plugins — and will not start language servers for it (or download their binaries).

Trust also enables one thing the list above does not name, and it is stated here because a reader would not infer it: **automatic verification.** After a turn that edits files completes, Nyx AI runs the project's own typecheck/test/lint commands. When no `.nyx/verify.json` pins that list, the commands are auto-detected from the `package.json` scripts. Those are repository scripts, and only the wrapper command (for example `npm test`) is risk-screened — the script body it executes is not. Outside Locked execution they therefore run project-authored code with your normal user permissions and without a further prompt, and the AI model can author that script body in the same session. This runs automatically only for a trusted workspace; an untrusted workspace either skips verification entirely or, where Locked execution is active, runs it inside the AppContainer. It can be turned off in Settings, or pinned to a fixed command list in `.nyx/verify.json`. The model cannot write that file directly — file-tool writes to it are refused — but it can *propose* a list through the registration seam described in §2.1, which shows you the complete normalized commands in a native dialog once and refuses any rated high, critical or unsafe. A pinned list is therefore model-extendable with your explicit approval, not a boundary the model is outside of. Trust can be revoked at any time

from Settings → Safety; revoking needs no confirmation and takes effect immediately, shutting down running language servers and disconnecting the workspace's MCP servers.

4. Local Data And Secrets

Nyx AI stores most product data locally.

Data	Storage model	Notes
API keys and provider credentials	Provider keys plus secret-shaped MCP environment/header/argument values and MCP OAuth tokens are stored through Electron safeStorage where supported. Legacy plaintext MCP records migrate when a real OS keychain is available; a fresh MCP argument secret refuses the atomic disk save if it cannot be protected.	Secrets must be decrypted in memory when used. Malware running as the same user may still be able to access them; an unavailable keychain can make a newly entered key session-only. For MCP servers specifically, a secret that cannot be re-protected keeps its previously stored value — so a rotated credential can revert at the next launch — and a stored value that cannot be decrypted in this session is omitted rather than used; in both cases the server runs with the wrong or missing secret rather than the application refusing to start. **1.5.427** — a removal is now reported as a distinct state from a key that cannot be decrypted, because the two used to look identical and they mean opposite things. If you remove a key and the settings file cannot be written (a full disk, a locked file, an antivirus hold), the encrypted value is still on disk: Settings says so, keeps offering Remove, and the next successful save in that run completes the deletion. If the app is closed before any save succeeds, the disk remains authoritative and the key is present again at next launch — so a removal reported as failed has to be retried, not assumed. A key that is live but could not be written is always described as live, even while an earlier removal is still owed. A username/password embedded in an OpenAI, Anthropic, Ollama Cloud, or cloud-embeddings base URL is rejected at save (the URL — and any key submitted with it — is not stored); Ollama Local URLs likewise reject

Data	Storage model	Notes
		embedded credentials. **1.5.423** — saving a provider key whose destination is neither a loopback address on this machine nor that provider's own default address requires a native OS confirmation that names the exact address which will receive the key. This closes an ordering gap: the earlier confirmation covered only a base-URL change that moved an already-saved key, so setting the address first and the key afterwards was not confirmed at all. Declining discards the key and keeps the rest of the save. **Name the same-user case plainly:** under **Standard (Job Object)** execution, code the model runs is itself a process running as you, and Standard execution does not isolate model-run code from same-user application data — including Nyx AI's stored provider credentials, whose at-rest protection is bound to your Windows account rather than to Nyx AI. Command/path screening is not a filesystem isolation boundary once a process is running. **Locked (AppContainer) execution provides the filesystem boundary** needed to protect Nyx AI's stored provider credentials from contained project code. That is not a claim that nothing outside the workspace is readable — as §2.2 says, the contained process keeps the narrow read-and-execute access needed to run toolchains — but none of that access covers your settings, your saved keys, or your documents. If you keep cloud API

Data	Storage model	Notes
		keys in Nyx AI, that is the practical reason to choose Locked.
Chat/session history and scheduled-run status	Local user-data/workspace files. Recognised secrets are scrubbed before session messages, attachment labels/document names, free-text plans/todos/tool summaries, and scheduler output reach their persistence/IPC funnels. Structural path/branch/tool identifiers remain exact so persistence does not break workspace continuity.	User-readable local data; pattern-based scrubbing is best-effort, so do not paste secrets into chats or task output deliberately.
Workspace metadata and indexes	Local workspace/user-data files.	Local-first, but the user can choose cloud-backed model or embedding flows.

Data	Storage model	Notes
Sandbox audit log	Local hash-chained log plus a key-authenticated head, encrypted where the operating system's protected storage is available. The separate JSONL activity audit is plaintext: secret-scrubbed on write and re-scrubbed on query/export for legacy-row safety — that scrubbing is pattern-based and best-effort, as for sessions — and it records the raw text of commands the model issued. **Historical note (packaged builds through 1.5.461):** a bundling defect meant the activity audit was held in memory only — its on-disk file was never created in any packaged build, so the plaintext action history (file operations, shell command text, dependency installs) did not survive an app exit and no on-disk activity log from those builds exists; the hash-chained log above was unaffected. From **1.5.462** the activity audit persists as `audit.jsonl` in an `audit` folder under the app's user-data directory, rotating at 5 MB into dated archives in the same folder. There is no retention policy beyond that rotation — rotated archives are never deleted automatically; the Settings audit opt-out stops new entries, and **Delete all my data** removes the whole audit folder.	Record edits and, after the safeStorage-backed v2 migration, rewritten/removed suffix metadata are detected; a signal found while logging is opted out is written before the next re-enable marker. It is not notarised or indestructible, and a same-user attacker/administrator who can remove every log/key/head artifact is outside the boundary. The tamper-evidence claims in this row cover the hash-chained log only; the plaintext activity audit is an ordinary editable local file.
Crash and non-fatal-error records	Local user-data files in a `crash` folder, rotated and capped. Error text and its context object are secret-scrubbed before writing.	Never uploaded — Nyx AI sends no crash reports. The folder is included in **Delete all my data**.

Data	Storage model	Notes
AI content reports and email drafts	Prepared locally for user review. AI-content reports contain only the sanitized reported response, sanitized reason, app version, and model name; general bug/security/contact drafts contain only the template and text the user chooses to send.	Nyx AI does not upload reports automatically. The user chooses whether to send the draft from their own email app or copy the report text.

The audit log is useful for self-review, debugging, and friendly audits. It is not notarised, not uploaded, and not designed as adversarial court evidence.

Delete-all is authorised by the main-owned native confirmation, not by the renderer, an argument, or a writable intent file. After confirmation, Nyx AI blocks new writers and keeps retrying sandbox access-revocation until it is proven complete. The wipe itself is performed by a dedicated minimal helper started directly by the confirmed application; the helper acts only after verifying that it was started by the confirmed Nyx AI instance for exactly this request, and if that verification, the application lock, the shutdown of captured processes, or the access-revocation above cannot be established, it stops without deleting anything. Nyx AI identifies its own live processes using read-only system queries only; a process it cannot inspect but which is reachable from Nyx AI's own ancestry is an explicit blocker, never an omitted or guessed identity, and no model or workspace input reaches those queries. Honest limits remain: process identity is checked against repeated point-in-time snapshots, which narrows but cannot fully close the operating system's own race windows around process exit, and a fully detached descendant that Windows can no longer attribute is never chased by guessing. The sweep deletes only Nyx AI user-data/temp locations, Nyx AI's own `Documents\Nyxion\Scheduled` output folder, the app-managed language-server cache under the user profile, attributable Nyx AI sandbox profiles, and the literal `.nyx` and `.nyxdeps` children of validated current/recorded workspace roots, never the workspace/project root itself, and reports leftovers rather than overclaiming. Files you placed inside those app-owned folders yourself are removed with them.

Chat mode blocks model-directed workspace and arbitrary-path writes; it is not a promise of zero local storage. Chat-originated turns keep the Chat tier even if the UI switches modes mid-turn. Code/shell/Python and arbitrary-path tools are blocked, document writers produce chat artifacts, memory is forced to the global user-data store, and workspace lifecycle hooks do not fire. Nyx AI can still write its own local app data: chat transcripts (including preview/artifact payloads), optional global memories, settings, usage and audit data. Document rendering may use short-lived temporary files with best-effort cleanup. Scheduled Chat tasks also write durable output to `Documents\Nyxion\Scheduled` and may add downloadable artifact payloads to chat history; these writes occur under the user-created schedule, without a fresh click at fire time. "Save to workspace…" and "Open externally" otherwise write only after an explicit user action. Code mode / Nyx AI Lab is

deliberately different: there the model edits files in the workspace you open — that is the purpose of that mode.

Chat mode is a capability allowlist, not process containment. It permits only MCP servers explicitly classified as chat-safe (the built-in Time server is the known default); all other MCP tool, resource, and prompt access is blocked in both renderer and main. A permitted MCP server still has the transport/process privileges disclosed in §3.

The chat HTML preview is delivered to its sandboxed iframe by a small in-memory, loopback-only service (1.5.349). Each preview is capability-scoped: the service exposes no filesystem path, answers only requests presenting that preview's own unguessable identifier, enforces bounded memory use, and serves a restrictive content-security policy that blocks remote subresources, connections and forms. It is never LAN-shared, and main-process navigation policy prevents the inline preview from navigating itself to another document. The serving step holds HTML in RAM and creates no preview temp file; the same payload can still be persisted as part of chat history. The service is torn down at quit and exposes only the model preview HTML already shown on screen.

5. Network Egress

Nyx AI sends no telemetry, analytics, crash reports, install IDs, or background usage reports. Network traffic exists only for documented features. If a future change adds a new network path, this table must be updated before release.

Automatic startup restoration, provider probes, remote Ollama catalog discovery, and the automatic update timer remain disabled until the current Terms version and consent-text hash have been durably accepted. The renderer and main process use the same acceptance predicate; workspace-open IPC also fails closed while consent is pending.

Feature	Destination	Data sent	Trigger
Chat/model call	The model provider endpoint the user configured or selected.	Prompt, conversation context, selected files/snippets, and attachments included in the request.	User sends a message or starts a model run. Explicit primary and scheduled choices carry provider identity; an unqualified id advertised by more than one configured provider is refused rather than resolved by provider registration order, a provider-qualified choice resolves only its exact owner, and a pinned scheduled task never falls through to role routing. **1.5.418** — when a provider refuses a request because you are over its per-minute allowance and states how long to wait, that one request may be re-sent once after the wait — to the same host, with the same key and the same content, and only when the wait the provider itself named is short. Nothing else is sent, and nothing new is stored. **1.5.416** — a redirect is followed only within the same origin. This previously applied only to the "Test connection" check; it now applies to the ordinary chat and streaming requests that carry the key on every turn, for every cloud provider (Ollama Cloud, OpenAI, Anthropic and any endpoint you point those slots at). Two

Feature	Destination	Data sent	Trigger
			separate exposures are closed by it: a key carried in a provider-specific header is not dropped automatically when a response redirects elsewhere, and a redirect that preserves the request body would otherwise replay your prompt, conversation and any attached snippets to a server you never named. Scope is the cloud model providers and cloud embeddings; MCP server connections follow their own redirect handling and are not covered here. The Local Ollama provider sends no credential, so no key is at risk on it — but when its base URL is pointed at a non-loopback host (see the Local Ollama row), its requests carry prompt content and are **not** redirect-guarded; that residual is listed in §7.

Feature	Destination	Data sent	Trigger
Model-run commands (shell/Python)	Wherever the approved command connects.	Whatever the command sends.	A model-issued command allowed by the approval policy runs with outbound network **enabled by default** in the sandbox. The no-network Locked profile removes outbound capability at the AppContainer OS level; Standard/Job network blocking is partial only (see §6). Command shapes that move local data outbound — file uploads to remote hosts and cloud storage, source-control pushes to a literal remote, and commands that re-point a source-control remote — are rated high, which requires the native confirmation in every mode including Bypass. Screening is pattern-based over known command families and is not a guarantee against every spelling; the detection patterns themselves are held in the private security record rather than published here.

Feature	Destination	Data sent	Trigger
Background helper / chat-title calls	The model provider for the helper's role. Small helper calls prefer a local model when one is reachable; heavier planner/coder/reviewer/ reasoning roles follow the user's model routing and may use a configured cloud provider. The prefer-local routing setting biases those heavier roles toward a suitable local model where one is reachable; it is a preference with cloud fallback, not an egress block.	A short summary, file excerpt, the first user+assistant messages, or the raw prompt — run through the privacy sanitizer first. (Before transport to any non-local model, all textual request components — the current message, history, compacted summaries, memory snippets, file context, and tool results — pass through the configured privacy sanitizer; images are not text-sanitized, and web-tool and MCP traffic follow separate paths.) The default sanitizer targets recognised credentials/secrets; ordinary personal information remains unless the user selects a stronger preset, and all sanitization is best-effort.	Automatically, to support file summaries, refactor/pre-edit checks, memory reflection, prompt classification, and naming a new chat.

Feature	Destination	Data sent	Trigger
Vision handoff	The vision provider and model resolved by the app's vision settings.	Image attachment plus prompt/context needed to describe it.	User attaches or asks about an image and the active model needs vision help — including automatic preview/screenshot descriptions and the analyze_image tool. All routes classify "local" by the RESOLVED destination (an Ollama model on a remote-host URL or a cloud-proxied model counts as cloud) and honor the "keep local-model images on this device" toggle: blocked → the image stays on-device and the model is told it has NOT seen it; allowed local→cloud → a visible notice. The handoff setting stores provider plus model. If privacy settings, the saved provider, or an unambiguous destination cannot be resolved, every handoff path fails closed and keeps the image on-device. When a screenshot description would go to a cloud vision model after the session has read sensitive content, a main-owned native confirmation is required before pixels leave the device.

Feature	Destination	Data sent	Trigger
Local Ollama	Product-owned loopback Ollama by default; alternatively the custom endpoint the user saves or `OLLAMA_HOST` when the user explicitly selects the Environment source. Ambient endpoint variables alone do not redirect a fresh/reset profile.	Prompt/embedding text; model-catalog metadata requests (`GET /api/tags` and selected `/api/show`); model-management requests.	After current Terms acceptance: when restoring or refreshing the model catalog, after the endpoint source or custom URL changes, when Connections opens or Refresh is clicked, after a Pull, and when a selected model is used. **1.5.427** adds automatic readiness and reconciliation triggers for existing catalogue reads (`GET /api/tags`). Before every non-slash message, a bounded readiness probe checks configured providers and Ollama reachability. When reachable local Ollama is the sole reason that check would pass, a second time- and size-bounded count-bearing read distinguishes an empty library from a usable provider. Banner reconciliation also runs after a settings change, when the app window regains focus, and when a command naming Ollama completes in the built-in terminal; after a confirmed ready transition it may refresh the cached model catalogue before hiding the notice. These checks exist so the "no model available" notice cannot outlive its evidence when installing a model or stopping the daemon

Feature	Destination	Data sent	Trigger
			changes no setting. Because the configured endpoint may be a host other than this computer, the automatic requests go to that machine without a further prompt; the destination is unchanged and no prompt, code or credential is included. Cloud or custom-provider users avoid the second count-bearing read, not the ordinary readiness check. Catalogue and optional model-metadata reads are bounded and excess replies fail closed. A settings-driven loopback→remote effective-endpoint transition and an unsaved remote Test require native confirmation; URLs with embedded credentials are rejected at save, and probe IPC returns only `scheme://host[:port]`. Remote endpoints are treated as cloud for egress scrubbing/labels and get no local-only privileges.
Ollama model pull	The Ollama daemon's configured model source/registry.	Model name and the daemon's download request metadata.	User triggers a model download/pull through Nyx AI.

Feature	Destination	Data sent	Trigger
Cloud embeddings	User-configured cloud embedding provider, or — when no dedicated embeddings provider is configured, or when the configured one fails — a **custom** Ollama Cloud base URL; the hosted ollama.com endpoint is skipped because it serves no embedding model. Product/environment/custom endpoint authority is resolved at each call, matching chat/model discovery; stale saved custom URLs are ignored after the source changes.	Code chunks, file/symbol metadata, and search text needed for embeddings — redacted through the same egress scrub as chat before sending.	User selects Cloud embeddings, or Auto mode uses a configured cloud embedding endpoint while the active chat model is cloud. Locality is enforced by resolved address: a Local-only index REFUSES to embed against a non-loopback "local" URL (no egress; keyword search continues), and a cloud-opted index sends a non-loopback "local" endpoint only the same redacted text as any cloud endpoint. **1.5.416** — as with the chat call, a redirect on a key-bearing embedding request is followed only within the same origin, so neither the key nor the code batch can be carried to a server the user never named.
Inline autocomplete	Local Ollama endpoint (loopback ENFORCED).	Code prefix/suffix around the cursor.	User enables autocomplete and types. If the Ollama Local URL resolves to a non-loopback address, autocomplete disables itself rather than sending code there (the consent text's local-only promise is enforced, not assumed).

Feature	Destination	Data sent	Trigger
Web fetch/search/URL screenshot	Public URLs or search providers reached by the guarded web tools — currently DuckDuckGo, Bing, Startpage, Ecosia, Wikipedia, Google News and Wikimedia Commons, plus any URL you or the model supplies.	URL, query, and fetched page request data. Browser-rendered search providers may set cookies in an ephemeral managed profile; Nyx AI does not click their consent buttons.	User/model triggers a web tool within the app's guardrails — **including Build mode, which web-searches your Build topic to ground the design** (disclosed in the Build dialog), and including a model-requested screenshot of a remote URL, which renders that page in the managed browser under these same request guards. HTTP search redirects and observable browser main/subrequests receive DNS-aware public-address guards; rendered search aborts observed non-HTTP traffic and state-changing methods and disables known service-worker, socket, WebRTC, and worker APIs. This interception is defense in depth, not an OS network jail: browser-managed or out-of-band channels may sit outside the page-request handler. Turn off under **Settings → Safety → "Allow the assistant to access the web"** (default on; narrower than the **Sandbox · no internet** tier, which also cuts sandbox egress). DNS failures and empty answers fail closed and each request/hop is checked, but a narrow

Feature	Destination	Data sent	Trigger
			DNS-rebinding race remains between address validation and connection establishment — a documented architectural residual.
Document remote images	URLs referenced by a document or generated document request; when presentation auto-images are enabled, Wikimedia search also receives title-derived queries.	HTTP request for the referenced image, or a slide title/search phrase sent to Wikimedia before selecting an image.	User opens/generates content that references remote images. Respects four controls, any of which forces a placeholder (the document still builds): the doc-specific **"Auto-fetch images in generated documents"** toggle (default on; Settings → Safety), the broader **"Allow the assistant to access the web"** master, the **Sandbox · no internet** tier, and — inside an unattended scheduled run only — the run's local-provenance latch, which suppresses remote image fetching for the rest of that run once the run has read workspace data (see §8.1). A model-passed `autoFetchImages:false` can also skip fetching for one request, but can never override the user setting.
MCP or plugin catalog/marketplace	Configured marketplace/catalog URL, and the plugin marketplace source.	Catalog HTTP request.	User fetches a catalog or marketplace source.

Feature	Destination	Data sent	Trigger
Extension child processes (stdio MCP servers, hooks, plugin-defined children, language servers)	Whatever the installed component contacts, plus the package registry its allow-listed runner (`npx` , `uvx` , `uv` , `docker`) fetches from on first run.	Whatever that component sends. Unless the optional MCP/hooks containment is enabled, Nyx AI does not mediate its traffic.	You connect or enable the component in a trusted workspace. An already-enabled MCP server can also reconnect — and so fetch its runner package — automatically when its trusted workspace is restored. **Unless the optional MCP/hooks containment is enabled (§2.4) — in which case a contained child under Locked receives no network capability at all — this traffic is not covered by Block Network** (see §6): it does not otherwise pass through the sandboxed-command firewall rules.
Plugin repository install	The Git repository URL selected from a marketplace source (HTTPS or SSH).	A `git clone` request, repository URL, and ordinary Git/SSH transport metadata; credentials or SSH-agent material may be used by Git according to the user's own configuration.	User approves the high-risk install and its native confirmation. The repository is cloned only after that approval.
Remote MCP/OAuth	The installed remote MCP server and its auth endpoints.	MCP payloads and OAuth flow data; a configured bootstrap command can make its own dependency/network requests.	User installs/connects a remote MCP server. An enabled server can also connect or reconnect automatically when its trusted workspace is restored; its bootstrap behaviour follows that installed component's configuration.

Feature	Destination	Data sent	Trigger
Scheduled tasks	Depends on the task.	Whatever the user-configured task does.	User-created schedule fires. Auto tasks use role routing; a pinned task must retain an exact provider/model pair and fails visibly if that pair is absent or ambiguous. Legacy id-only pins require user re-selection before running. Once an unattended run reads workspace data (file read, glob, edit or patch), a main-process provenance latch refuses that run's remaining web search/fetch calls and suppresses remote document images for the rest of the run; the run's own calls to the configured model provider are **not** affected and still carry whatever context the task assembled (see §8.1).

Feature	Destination	Data sent	Trigger
Managed browser download	Google's Chrome-for-Testing HTTPS object for the exact Puppeteer-tested revision/platform/archive.	Download request for that Chrome Headless Shell archive; ordinary connection metadata. Download and installed sizes vary by build.	A preview / screenshot / build-QA feature needs it and no usable installed/cached candidate remains, or the user explicitly requests it in Settings. **Main-owned native consent is required before any first fetch for that tested revision.** A grant is stored with the exact revision; a legacy grant or different tested revision asks again. "Not now" fails the calling feature and can be reversed at Settings → Safety → "Download browser engine". Every redirect is bounded and must retain HTTPS, Google's exact origin, and the exact revision/platform/archive pathname. The first fetch is trust-on-first-use: its integrity rests on that pinned HTTPS origin, not on an independently NYX-pinned upstream checksum. Cached copies then receive best-effort local integrity checking, and cache maintenance verifies it is operating only on app-managed browser files — failing closed rather than deleting anything whose ownership it cannot prove. A legacy browser cache inside an older packaged resources

Feature	Destination	Data sent	Trigger
			directory is trusted as an installed application resource and is only existence/layout checked. When used, the engine loads/renders requested content and may execute page scripts under separate browser/network guards, with the browser's own OS sandbox attempted first (§3); those guards reduce risk but are not a guarantee. Google/upstream terms and embedded notices apply.
First-use language-server download	The HTTPS artifact host in the pinned LSP manifest; redirects stay on that host except GitHub releases may use the exact allowlisted official GitHub asset hosts.	Download request for the language-server binary. Every hop is HTTPS and DNS-guarded; the completed artifact must match the manifest SHA-256 before extraction/use.	An enabled compatible language server is prewarmed when a trusted workspace opens, or a language feature first needs it, and no cached copy exists.
Python analysis package cache (`execute_python` , `analyze_csv`)	The configured Python package index (normally PyPI).	Requests for allow-listed package names and compatible binary wheels. Wheels and their dependencies are imported/executed by Python.	A high-risk approved Python/CSV action first needs an uncached package set. The approval text names the download. Successful sets are reused from a Nyx AI user-data cache; failed installs are not cached. Either Assistant web access OFF or the broader no-network setting blocks a missing-cache download. `analyze_csv` requests pandas, numpy, matplotlib, and seaborn.

Feature	Destination	Data sent	Trigger
Rust GNU toolchain repair	`rustup` / Rust distribution hosts for `stable-x86_64-pc-windows-gnu`.	Toolchain download request and standard `rustup` metadata.	User approves native OS consent that explicitly names the internet download and out-of-sandbox execution. The repair refuses before consent/spawn while the no-network setting is active.
Dependency install (`install_dependency`)	Official package registries (crates.io, registry.npmjs.org, PyPI, the Go module proxy, Maven Central, NuGet, RubyGems, Packagist), together with the companion download, index and checksum hosts those same ecosystems use — including Gradle's plugin portal and wrapper-distribution host, and the Yarn registry mirror, which are themselves package sources rather than mere companion hosts. The allow-list names each one, and admits no host outside official registry infrastructure.	Package-manager requests for the named packages/versions; lifecycle/build scripts are disabled during the fetch where the package manager supports it.	User approves a native OS consent for that specific install. Rust installs under the Locked tier are fetched by a trusted fixed-argv helper routed through the app's registry-only loopback proxy (the sandbox itself is kernel-blocked from loopback, so in-jail proxy scoping is impossible — measured 2026-07-03); other Locked-tier installs fetch directly from inside the sandbox, and non-Locked tiers apply the loopback proxy by env var (advisory). The result reports the actual posture.

Feature	Destination	Data sent	Trigger
Cloud-key probe	Ollama Cloud, OpenAI, Anthropic, or the configured cloud base URL.	A minimal chat-scope check to confirm the saved key works for inference: a model-list request, then a minimal throwaway chat call to the same host + key. A model the endpoint declines to serve, or that does not answer in time, may be retried against another model the endpoint itself advertises — so a valid key is not rejected because of which model was chosen for the test — within a bounded request count and a bounded shared time budget. A 400 never verifies access; only an actual 2xx does. All attempts go to the same host + key and stop at the first success or at a response showing that retrying will not help — the key being refused, the endpoint throttling, or a server-side failure. A redirect is followed only within the same origin, so no response can move the key to a server other than the one the user named. No project code, files, or prompt content in any of these calls.	Ollama Cloud: after current Terms acceptance, on app startup when a cloud key is configured and after relevant save/change. OpenAI/Anthropic: only when the user clicks **Test connection** (or during onboarding key entry) — not on startup.

Feature	Destination	Data sent	Trigger
Model-catalog discovery (`/v1/models`)	The configured provider's base URL — Ollama Cloud, or an OpenAI-/Anthropic-compatible endpoint you pointed a provider slot at.	A `GET /v1/models` list request (key in the auth header); receives the provider's model catalog. No project code, files, or prompt content.	On startup once the model list / router is first used with a saved key (cached ~5 min). Ollama Cloud discovers whenever a key is set; the OpenAI/Anthropic slots discover only when pointed at a non-default host — the default `api.openai.com` / `api.anthropic.com` use built-in lists and make no such call. **1.5.416** — as with the chat call, a redirect on this request is followed only within the same origin, for every cloud provider slot.
Google Drive via MCP/integration	Google APIs.	Data handled by the user-installed Google Drive MCP server or integration.	User installs, configures, and triggers that integration.
Update check	An update host we operate, served via Cloudflare R2 (`<feed>/latest.yml`).	Request IP + app version (in the User-Agent), request time, and ordinary connection metadata; the request contains no Nyx AI-generated install ID, device ID, usage event, prompt, file content, or chat history. Responses are size- and time-bounded.	After current Terms acceptance, on by default: shortly after launch + at most about once a day across restarts; also when the user clicks "Check for updates". Saving the opt-out in Settings → Advanced → Updates removes future timers and aborts an in-flight automatic check; the manual button remains available.

Feature	Destination	Data sent	Trigger
Update download	The same update host (Cloudflare R2); explicit artifact URLs and every redirect destination on the in-app path must remain on that origin.	The in-app path receives a full signed installer; differential/blockmap download is disabled. The download is size- and time-bounded and must exactly match the manifest's declared size; SHA-512 and the full NYX Limited Authenticode publisher are then verified, and the signed installer's version metadata must equal the stable manifest version before staging or raising the anti-rollback floor.	Only after a user requests Download and accepts a main-process native confirmation bound to that exact version. Installation requires a second native confirmation for the already-verified version. A manual installer/release link opens in the user's browser and is outside this in-app verification path; the user must verify its published signature/checksum before running it.
LAN preview sharing	Local LAN listener.	Preview content served to a token-bearing local device. Nyx AI refuses to serve paths it recognises as sensitive — credential directories, key material, files named like `.env*`, and the app's own per-workspace `.nyx` folder, apart from the rendered preview output this feature exists to serve. That recognition is name- and pattern-based and therefore best-effort, as elsewhere in this document: it is not a guarantee that every sensitive file under the shared root is withheld. Share only a tree you would show to anyone on your network who holds the link.	The global LAN-sharing permission may be enabled by default, but Nyx AI opens no LAN listener until the user explicitly approves an individual share; each share uses a new token and expires.

Feature	Destination	Data sent	Trigger
Preview / document-render subresources	Public hosts referenced by a project dev-server preview or render content, subject to surface-specific guards.	Requests for referenced images, scripts, styles, or fonts where permitted.	User opens or generates a preview/render. Chat-inline and built-in static previews send a CSP that blocks remote subresources, connections, and forms; document renders honor Block Network and reject loopback/LAN/metadata destinations. **From 1.5.480 the automation browser the model drives — the one behind preview screenshots, page reads and script evaluation — also honors Block Network, aborting every non-loopback request while it is closed.** A third-party project dev server's own response policy may still permit public remote subresources in the renderer's preview pane, which is the surface a person looks at rather than one the model reads back. These remain separate rendering surfaces, not the model shell sandbox.

Honest summary: most network activity happens only when the user acts. Some enabled features run in the background, and first-use tool downloads go to documented fixed hosts. None of these paths is telemetry.

6. What "Block Network" Covers

(The corresponding Safety-settings control is labelled "Allow internet access for AI-run code" — turning it off is the "Block Network" state described here.)

Network blocking applies to the model's own execution and supported network tools. In Locked execution with no-network enabled, outbound network access from the contained model-run process is blocked at the OS capability level.

In Standard execution, Block Network is a compatibility fallback, not a complete OS network jail. It can block or proxy only the supported command families Nyx AI can screen; tools such as Node, Python, or PowerShell may still reach the network. Its effect also depends on how Windows has classified the network you are currently connected to, and on some connections it does not apply at all. Standard-tier network blocking must therefore never be relied on as an OS network jail. Use the no-network Locked profile when network isolation is the boundary that matters.

Live changes take effect immediately: toggling Block Network — or resetting settings — while the sandbox stays enabled reconciles the Standard-tier firewall rule and the status badge at once, rather than only at the next workspace open. In-flight commands keep the network capability they were launched with; the change applies to newly launched commands.

This does **not** mean that every other app feature is silent. The following may still have network behaviour if the user enabled or triggered them:

- MCP servers, plugin-defined MCP/hook child processes, hooks, and language servers running outside the model shell sandbox (unless optional MCP/hooks containment is enabled); scheduled task orchestration and model/provider traffic also run outside it, but scheduled shell actions route through the same shell broker and active execution backend as interactive commands;
- the renderer's own preview pane loading user or generated content — the model-driven automation browser behind preview screenshots, page reads and script evaluation IS covered from 1.5.480, see §5;
- update checks, marketplace fetches, cloud providers, or first-use downloads triggered by the user or enabled feature.

For a practical air gap, use local models, disable cloud keys and networked integrations, disable scheduled/network tasks, and avoid networked preview content.

7. Residual Risks

These are known limitations. They are documented so users understand the real boundary.

1. **The audit logs are local-only.** The hash-chained sandbox audit log can show evidence of record/head modification, but it has no external anti-rollback anchor. Someone with sufficient machine access can delete the log, protected key and head together. The separate plaintext activity audit described in §4 carries no tamper-evidence at all: it is an ordinary local file that any process running as you can edit or delete.
2. **Public releases and updates are signed; local development/test builds may not be.** The public release workflow Authenticode-signs artifacts and verifies the expected NYX LIMITED

publisher identity. Internal, development, or test packages can be unsigned, trigger SmartScreen, and provide weaker provenance.

3. **Updates require user confirmation to download and install.** Nyx AI checks for updates automatically (on by default, opt-out) and notifies the user, but the download and install each require a main-process native confirmation. Users who disable the check or decline updates stay on their build.
4. **Secrets exist in memory while used.** OS keychain storage protects secrets at rest, not against malware running as the same user.
5. **Renderer compromise remains serious.** Context isolation reduces exposure, but a fully compromised renderer can reach the application capabilities that are exposed to the renderer. UI-only limits and renderer-mediated approval controls therefore must not be treated as security boundaries against a compromised renderer: they constrain a misled *model*, which cannot skip the interface's own dispatch, not an attacker running inside the interface. Main-process and OS-enforced controls — native confirmations, main-side path/control-file guards, and Locked execution's OS boundary around project commands — provide the stronger boundary; Standard execution does not provide that filesystem jail. Renderer dependencies and any content loaded into UI surfaces still need security review.
6. **Extensions are a trust decision.** MCP servers, plugins, hooks, and scheduled tasks can have broad local effects if the user installs or configures malicious ones.
7. **Standard execution is weaker than Locked execution.** The Job Object tier is useful defense in depth, but not a kernel filesystem or universal network jail.
8. **Automatic fallback changes the strength of the boundary.** In Automatic mode, Nyx AI may fall back to the Standard tier if Locked execution cannot start. Users relying on the stronger boundary should explicitly select Locked execution and confirm the active tier.
9. **Command screening is not semantic code review.** A command line can launch a file whose contents need separate review. Approval prompts and Locked execution are the more important controls for staged code.
10. **Preview and document rendering are not the model shell sandbox.** Puppeteer browsers are separate main-process rendering surfaces, outside AppContainer and the Job Object used for model-run commands. A custom browser path requires main-owned native confirmation before Nyx AI inspects or launches it, runs with normal user authority, and is re-confirmed if its verified identity changes. Chromium's own OS sandbox is attempted first; any compatibility retry without it requires a separate native decision and otherwise fails closed. Verification reduces replacement risk but is not a kernel launch binding, so a narrow check-to-launch race and adjacent browser resources remain residual risks.
11. **Document bombs are mitigated, not impossible.** File readers apply size and structure limits, but deliberately hostile documents may still cause a recoverable app crash.

12. **No sandbox is perfect.** A novel OS, Electron, browser, dependency, or implementation vulnerability could bypass intended controls.
13. **Allowed-path removal and crash recovery are fail-closed.** Removing a path from Allowed paths (or resetting Safety settings) requires every journalled sandbox access grant to be physically revoked and verified before the settings change or confirmed data deletion can report success; Save is refused while a live contained process remains, and Automatic mode cannot downgrade a command blocked by this barrier to a weaker tier or to no containment — it is refused. The honest limits, briefly:
    - An abnormal exit can leave a filesystem permission behind temporarily; the next clean session revokes it. That is *recovery*, not prevention.
    - A normal quit does not perform a physical permission sweep. Outstanding grants are re-proved at the next proof-required cleanup — an allowed-paths save or reset, or a confirmed **Delete all my data** — rather than at ordinary startup. Until then the permission remains on disk, and it will also survive an uninstall.
    - An inaccessible or offline path is never counted as proof of revocation, so it can prevent verified cleanup; the flows above stay blocked until cleanup is proved.
    - A journalled grant whose recorded object Windows reports as simply not there does not stop Locked execution: nothing is present for a contained process to open, so the tier starts and the record is retained for a later retry rather than discarded. A probe that fails for any other reason — a permission denial, an I/O error, a device that is not ready — counts as present and still refuses. The residual is the narrow case where the object survives elsewhere, such as a drive letter removed and later reattached, while the entry at stake is a recursive Modify grant.
    - Confirmed **Delete all my data** first stops running Locked commands and quiesces sandboxed integrations so no child retains the authority being revoked. If a child cannot stop or later cleanup cannot be proved, no local app-data deletion starts. An integration already stopped for that safety check may deliberately remain disconnected after a partial/unproved cleanup; reconnect it or restart Nyx AI before retrying. A failure before authority cleanup, or after cleanup fully converges, resumes integrations that were proved safe to resume.
    - Filesystem identity reporting is not reliable on every volume or redirected storage service. A moved or replaced object can therefore leave permission residue that Nyx AI cannot prove or remove automatically. Uncertainty is reported, fresh sandbox authority is not granted from it, and manual cleanup may be required.
    - If protected journal state is lost beyond repair, unknown historical permission residue and profile metadata may remain on disk (Nyx AI does not scan drives for them), and confirmed deletion cannot prove removal of records it no longer has.
    - Such residue does not become authority for a fresh sandbox: a new isolated identity is selected and an older identity is not reused. Protective deny-entry residue may also remain; it is not an access grant.

Win32 access-control behaviour still requires the on-metal Windows matrix in §8 before a release may claim operational validation.

14. **A release can ship with a known dependency advisory for which no safe upstream fix exists.** The limitation is that such an advisory is bounded and documented rather than eliminated. The release process runs dependency-audit and software-composition checks, and a release must not claim that a specific scanner ran unless its evidence records that run. Where no safe fixed version of an affected package exists, reachability and practical exposure are assessed and documented separately, and the exposure is bounded where possible — including excluding unused affected code from the shipped package, with a packaging gate that enforces the exclusion. Transitive fixes are actively pinned when a safe upgrade exists; the shipped software bill of materials records the exact dependency set.
15. **A trusted Cargo dependency-prefetch helper has a narrow race.** A narrow race remains in this trusted dependency-prefetch path when repository state changes concurrently; the helper's environment and transport are restricted, and validation refusals fall back to an in-jail build. Avoid concurrent project modification during dependency preparation for untrusted repositories, and prefer the structured dependency tool. Additional kernel-enforced hardening is tracked, and detailed reproduction mechanics remain in the private audit until the residual is fully closed.
16. **Workspace control metadata created by contained code can affect a later host-side developer action.** AppContainer applies targeted write denies, and structured host-side Git is pinned to the selected repository root, receives a scrubbed environment, suppresses workspace hooks until the repository is trusted, and requires additional integrity checks before any automatic commit; any ambiguity leaves the changes for manual review. Compatibility needed for ordinary in-jail Git still means this is not a complete immutable-control-files guarantee. Trust only projects whose repository configuration you would run, and review changes before manual host Git operations. A stronger kernel rule that preserves normal Git operation is tracked.
17. **The free-tier daily usage reset relies on the device's local clock.** Offline software cannot reliably distinguish a legitimate long shutdown from deliberate clock manipulation without contacting a trusted time source. The implementation includes a local anti-abuse ratchet, but this remains a product limit rather than a security boundary; exact thresholds and bypass mechanics are intentionally not published in this bundled trust document.
18. **A remote "Local Ollama" endpoint, and MCP server connections, do not have the same-origin redirect guard.** Since 1.5.416 every cloud model-provider and cloud embedding request refuses a redirect that would move it to another origin, so neither the key nor the request body can be carried to a server the user never named. Two paths are outside that: the Local Ollama provider, which sends no credential but does send prompt content and may be pointed at a non-loopback host (a change that already requires native confirmation); and MCP servers, which re-issue their own requests across redirect hops while enforcing HTTPS and address checks but not same-origin. In both cases the destination is one the user configured; the residual is that a hostile or compromised configured endpoint could redirect content onward.

19. **The personal-folder warning is best-effort path recognition, not a containment boundary.** A redirected personal folder that Windows does not report through the locations Nyx AI checks, or the same folder reached through an alternate path representation, may open as an ordinary project without that extra warning. The brokered workspace boundary still applies to the root that was opened; this residual affects the warning only, not the Locked jail.
20. **Filesystem grants written by earlier versions can remain as attributable or unverifiable residue after upgrade.** A document-render failure reported in 1.5.460–1.5.469 was traced to a Nyx AI defect in how Locked execution wrote filesystem permissions, and has since been fixed and verified by controlled before/after testing. The diagnostic detail and the specific permission entries involved are held in the private security record.

    Legacy grants are repaired only when Nyx AI can positively attribute them and prove the relevant authority is gone. Required workspace authority stays fail-closed while that cannot be proved; unrelated or unattributable entries are not removed. The app surfaces repair progress without treating display state as proof that the boundary is ready.

    This conservative repair may leave older unknown local residue that requires support or manual cleanup. It does not broaden permissions, weaken the selected execution tier, or claim cleanup succeeded without evidence. Document rendering also fails honestly when its own required runtime boundary cannot be established.

8. Verification And Release Practice

Before each public release we verify:

- the active execution tier is shown honestly in the UI;
- Locked execution still confines filesystem and network behaviour on real Windows hardware;
- Standard execution still applies process lifetime/resource controls when the Job is successfully created/assigned, and visibly reports degradation when it is not;
- hard-blocked and approval-gated commands still behave as expected;
- the network egress table matches the implementation;
- generated PDFs, including their embedded cross-document links, and in-app Legal/Consent text match the Markdown docs;
- the corresponding-source and relinking archives promised by the bundled notices are hash-verified and published to the update host as part of the release upload, before the update manifest points at the new build;
- new integrations add their own trust prompts and this document is updated before release.

Detailed validation scripts, issue reproductions, and raw security audit notes should live in internal, unbundled audit records. Public release notes can summarise fixed issues after patches ship.

8.1 Additional boundary outcomes

- Scheduled code/research work can use the web before it touches workspace data; after workspace-derived content is read or changed, external web tools and remote document images are disabled for the rest of that unattended run.
- Renderer-supplied recent/restore paths are not workspace authority. New roots require the native folder picker, app-owned data cannot become a workspace or allowed root, and launch authority is checked again immediately before a child starts.
- Repository-controlled execution outside Locked requires the documented native trust decision, and that decision is workspace-wide for the session: one Allow covers later repository-controlled commands in the same workspace — including ones the model issues afterwards — until Nyx AI is closed. The covered class is wider than builds and tests: it reaches working-tree Git operations and project-local binaries that project code, including the model, can create after the click. It is held in memory only, never written to disk, and only an actual Allow is remembered. Commands classified high or critical keep their own per-command native confirmation, which runs before and independently of that grant. Unattended work that cannot obtain a required decision is refused. Automatic may use the screened always-on floor while its backend resolves; an explicitly selected Sandbox level remains fail-closed and never silently falls back.
- Locked children receive their lifecycle and filesystem protections before they run; protected workspace-control paths remain denied until cleanup is proved.
- Preview and MCP lifecycle boundaries preserve workspace ownership, origin and secret cleanup across asynchronous changes; incomplete cleanup is not treated as success.
- Release staging is bound to a clean source revision and exact lockfile. This is not isolation from malicious dependencies running as the build user, so public signing still requires a controlled release host and separately protected credentials.

DNS address validation still requires a future connection-time pinning proxy for Chromium browsing; the documented rebinding residual remains until the actual socket destination is bound to the vetted address.

9. When To Update This Document

Update this document before release when any change:

- adds a new network path or background timer;
- changes model-provider, embedding, web, preview, or update behaviour;
- changes sandbox backend selection, fallback, or UI labels;
- adds a new extension, plugin, MCP, hook, or scheduler surface;
- changes where secrets, logs, sessions, or indexes are stored;
- changes approval prompts or hard-blocking behaviour;
- strengthens or weakens any "local-first", "no telemetry", or sandbox claim.

If the implementation changes and this document does not, the release review is incomplete.

Published by **NYX LIMITED**, a company registered in England and Wales (company number 17261427), registered office 70 Clarkehouse Road, Sheffield, England, S10 2LJ. ICO registration **ZC167877**. Contact: customerservice@nyxlimited.com.